CS747: Assignment 2

Shubham Anand Jain, 17D070010

October 21, 2020

1 MDP Solving by various algorithms

In this section, we look at different algorithms that we use for solving MDPs, both continuous and episodic. We also briefly explain the implementation of these algorithms.

1.1 Value Iteration

In the value iteration algorithm, we start with an initial large negative values vector V of values, and continuously iterate to get a better V until the difference between the V values in consecutive iterations is small enough (we define small enough as the L_1 norm being smaller than 10^{-10}). Note that the initial value being huge negative values is important - else this fails to work in negative reward scenarios, where the negative reward may be an improvement (We can change the values of tolerance or how negative the $v's$ are if we want the algorithm to work quickly).

1.2 Howard's Policy Iteration

Our implementation of Howard's Policy iteration is very similar in ideology to the code for Value Iteration. The differences are:

- Calculated of the V values - calculated after each policy change by the numpy library functions, solving the Bellman Equations
- While solving Bellman equations, we take special care not to make the matrix non-singular by looking at the end states and forcing the $V(s^T) = 0$ at such terminal states
- Exit condition of the loop - we just check that there is no change in the optimal policy at each step

1.3 Linear Programming

We use the Pulp library to write the Linear programming algorithm for this assignment. The LP formulation is the same as given in the class notes. The only additional comment is that, to speed up the LP solution, we used Python dicts. LP takes the most amount of time out of the three algorithms - for the MDP, it is quite fast, but to solve maze it takes a significant amount of time (maybe an hour for grid 100).

2 Maze Solving via MDPs

2.1 Modelling the Maze as MDP

The maze can be naturally modelled as an episodic MDP. We first re-iterate the conditions for an episodic MDP, and list some observations that we can make about the maze:

- Each cell that is not blocked by a path should be one state in the MDP
- The directions we can move in, correspond to the actions that we want to allow - thus, there should be 4 actions
- Each step that we take, represents some "cost". Thus, perhaps taking a step should cost "-1" value, and we want to maximize our final total - thus, we minimize number of steps
- In an episodic MDP, it is necessary that for each policy and state pair, there should be a non-zero probability for an agent to reach the terminal state. Thus, each of our actions needs to be stochastic in some sense - each action should allow the agent to go to any of the possible places, so that it may end up in the final state in some way.

Keeping all the above considerations in mind, I came up with the following algorithm to convert the Maze into an MDP:

- Only the cells not covered by a path are considered as states
- Actions, with 0.6 probability, move to North, South, East or West (depending on the action); and with the rest of the probability evenly distributed amongst the other options, goes to any other direction
- If an action causes the algorithm to hit a wall, instead the algorithm remains in the same state; thus, the transition is from the state to itself
- There is a reward of "-1" for every step that the algorithm takes before it reaches the end state. On reaching the end state, there is a huge positive reward, and thereafter the algorithm remains in this state with 0 reward
- We choose $\gamma = 1$, which ensures correctness, and signifies the fact that each step is counted equally

The above algorithm to convert Mazes to MDPs was verified to work on all the provided maze inputs, and all the MDP solving algorithms above converge on the solution as well.

2.2 Time Taken by various algorithms

For the MDP solving problem, the time taken by all the algorithms is small because of the small number of states (It takes 34.3 seconds summed across all algorithms and all testcases according to PlannerVerifyOutput). Here, we document time taken for each algorithm in the Maze -¿ MDP case, to provide a benchmark (and to signify which algorithm to run to check our encoder and decoder):
(Note that the below times are to solve all grids, from grid10 to grid100)

- Value Iteration: 21 minutes, 19 seconds
- Howard's Policy Iteration: 100 minutes, 15 seconds
- Linear Programming: 100 minutes, 47 seconds

An interesting point is that, previously I had the wrong encoding for the maze - however, my algorithms were blazingly fast on that encoding (about 1 minute on the 100*100 grid), and gave the correct outputs. Thus, I think that if the above times are too slow, then they are possibly because my encoding is sub-optimal in terms of number of constraints somehow; since the MDP times seem to be quite fast.