CS747 Assignment 2

Manas Vashistha
17D070064

October 23, 2020

Task 1

Value Iteration

- Value function for each state is initialised as $V^0(s) = \max\limits_{a \in A} \left(\sum\limits_{s' \in S} T(s,a,s')R(s,a,s') \right)$.
- The updated value function V^{t+1} is computed using the previous value function V^t.
- Stopping condition: $\begin{cases} \|V^{t+1}(s) - V^t(s)\|_\infty < 10^{-8} & \gamma \in \{0,1\} \\ \|V^{t+1}(s) - V^t(s)\|_\infty < 10^{-6}\left(\dfrac{1}{\gamma} - 1\right) & otherwise \end{cases}$.
- $Q(s,a)$ is evaluated using $V^*(s)$ obtained after it converges as per above conditions.
- The optimal action for a state s, $\pi^*(s)$ is equal to the argument a for which the value of $Q(s,a)$ (calculated from previous step) is maximised $\pi^*(s) = argmax_{a \in A} Q(s,a)$. Ties are broken by considering the lowest argument.

Linear Programming

- The problem is solved using the command PULP_CBC_CMD.solve() and we get the optimal Value function V^*.
- The optimal policy is calculated using a similar approach as in Value Iteration.

Howard's Policy Iteration

- The initial policy $\pi^0(s)$ is a random choice of actions $a \in A$ for each state.
- The value function $V^t(s)$ is computed using $\pi^t(s)$ by solving the bellman's equations.
- For solving, the Bellman's equations are first converted to the form $Ax = B$ $(x = V^{\pi^t}(s))$ and then $V^{\pi^t}(s)$ is computed using $\mathrm{x = np.linalg.solve(A,\ B)}$. For the case where A might be a singular matrix, $\mathrm{x = np.linalg.lstsq(A,\ B)[0]}$ is used.
- After computing $V^{\pi^t}(s)$, we evaluate $Q^{\pi^t}(s,a)$ $\forall a \in A$ and select the argument which maximizes action value function $Q^{\pi^t}(s,a)$, as the action under new policy for each state s i.e. $\pi^{t+1}(s) = argmax_{a \in A} Q^{\pi^t}(s,a)$.
- The steps above are repeated until $\pi^{t+1} = \pi^t$. On stopping, π^t is the optimal policy and the optimal value function V^* is obtained using this optimal policy.

Task 2

Maze to MDP

- Algorithm used for solving MazeMDP is Value Iteration.
- The maze is assumed to consist of n_T cells in total, out of which n_1 cells have value 1 (wall tiles) and others have value either 0, 2, or 3.
- Each non-wall tile is considered as a separate state of MDP. $\implies$ Total $n_T - n_1$ states.
- A transition in the MDP is representated as $[s_t, ac, s_{t+1}, r, 1]$. Here s_t & s_{t+1} are the states at time step *t* and *t+1* respectively i.e. before and after taking the action *ac* from s_t. r represents the reward. The probability of reaching s_{t+1} from s_t on taking an action *ac* is 1 as all the state transitions are deterministic.
- Action from a state can be from $\{0, 1, 2, 3\}$ which correspond to $\{N,\ E,\ S,\ W\}$ respectively.
- All the MDPs are episodic tasks and the terminal states are the end states of the maze from where no further transitions can be made.
- If a wall is encountered on taking an action, the MDP returns to its original state thus forming a self loop and receives a penalty for this.
- The rewards for a transition is $R(s, a, s') = \begin{cases} 100(n_T - n_1) & s' \in end\_states \\ -0.5 & otherwise \end{cases}$.
- $\gamma = 0.99$ for all the MDPs. This will be crucial for solving the MDPs where the shortest path is comparatively long and MDP requires to take into account many states.

Shortest Path

The path determined using the above approach is indeed the shortest path from the start state to the end state. This can be shown as a simple exercise as follows-

Let, the number of steps required to reach the end state from the starting state be k. Then we can write the value function of the starting state as-

$$V(s^0) = -0.5 - 0.5\gamma - 0.5\gamma^2 - 0.5\gamma^3 - \ldots - 0.5\gamma^{k-1} + \gamma^k(n_T - n_1)$$

Clearly as k is increased, $V(s^0)$ decreases as γ^k goes down and there are more negative terms in the expression. But for the policy to be optimal the value function needs to be maximized which can only happen if we find the minimum value of k. Hence k is the shortest number of steps required to reach end state from starting state.