CS747 - PROGRAMMING ASSIGNMENT- 2 REPORT

NAME - ARPIT DWIVEDI

Roll Number-200100032

Department of Mechanical Engineering

1 *Task 1*

For this task as per passed arguments the if and else condition will decide which algorithm to use.
Note: In case of random policy passed TA's has provided two formats one for Task one which contains only the policy while for task2 they had given corresponding states also hence the Policy check if statement contains two bifurcation for the same.

1.1 Value Iteration

It involves following steps to reach the optimal policy for given MDP:

- First data is extracted from the MDP file passed.i.e..Terminal states, NumStates, NumActions ,Trans probab, Reward, mdptype, Gamma
- Initial guess for the Value function for states which is array containing each element as 1 except for the states that are terminal, we will initialize them to 0 as they will not be going to update
- We will run the Bellman optimality operator on the initial guess and run the loop till the changes in the next Value states and previous one is less than ϵ..*i.e...* ($||V_{Bellman-on-Previous}$ - $V_{Previous}||_\infty$) <= e^{-10}
- That policy will be chosen for a state for which we have the maximum updated State value function
- Then we will return the obtained Optimal Values and policy corresponding to the iteration which we ran

1.2 Linear Programming

- We will use the standard way to solve the linear programming problem using pulp
- The main thing is in defining the constraints in which we need to take care whether the mdptype is episodic or continuous as for the episodic one we will put constraints as standard constraints we had in slides while when it comes for the episodic we need to fix the Value function for those to be zero so we will add these constraints for those states

$$0 <= V_{terminalstates} <= 0$$

- This will give us the Value function but we are still left to find out the policy for those optimal value function, for which i have defined Calculatepolicy function in which I had checked which is the policy over which if we evaluate the state value function using the optimal state values from Linear Solver, we get minimum error.

1.3 Howard's Policy Iteration

- We will first generate a random policy and then evaluate the state value function for that policy for the given mdp

1.3.1 Value Function for a given Policy

I have the values $V(s) \forall s \epsilon S$ so i can formulate the linear equation as:

$$\mathbf{Ax} = \mathbf{b}$$
$$where \mathbf{A} \epsilon \mathbb{R}^{s \times s} \text{ s is Number of States for given MDP}$$

Clearly on solving this linear equation we will have the Value function for the states. **Note:** Assume that the equation is determinate for the given MDPs

- Then we will calculate the set of improvable actions for the states by evaluating the value function for other actions
- We will consider only those policies for which the Value function calculated are greater than the previously calculated by at least e^{-8} as exact zero condition was giving erroneous results
- We will append the policy with maximum Value it to the list of improvable set of actions for all states.
- Now we will run the loop of updating the policy since we found that the Improvable actions set for all states are empty set.i.e..$IA_S = \phi$
- If the set is non-empty we will update the policy by choosing that policy we found in previous to previous step. We will repeat till we achieve the condition of previous step
- The final results of the iteration will give us the optimal policy and the value function for the policy

2 *Task 2*

2.1 Formulation of MDP

- I have considered Number of States for the MDP to be 1 more than the number of states provided from the list of all possible states generated from the script provided. This extra state I considered is the final winning state
- Number of actions would be 5 as only possible attempt A can do are:
  - 0- Defend
  - 1- Attempt a single run

  - 2- Attempt 2 runs
  - 4- Attempt a boundary (4 runs)
  - 6- Attempt a six

- The terminal states I had considered is the final winning state as my algorithm don't need the loosing state so not considered that as other terminal state

- Discount factor is taken to be 1 as this is an episodic task

- mdptype would be EPISODIC

- **TRANSITION PROBABILITIES**: Following are the obvious things we need to understand while considering the game

  - The transition probability(T) would be zero for if it causes the number of balls left to increase. Example: $T(1307, a, 14rr) = 0$ for an action a and runs left rr.
  - The transition probability(T) would zero if the runs to be scored increases on moving from one state to other
  - Now A can choose an action which could result him a consecutive state in terms of number of balls left in that case we can directly use the difference of runs between those state and using the action info we will get the info of transition probability. Example: $T(1306, 0, 1204) = P(2|0)$ where $P(r|a)$ means Probability of scoring 'r' runs given 'a' action policy chosen.
  - Now the transition probability to reach the next consecutive state, with respect of number of balls, if A had taken 1, 3 or difference of runs between two given states is greater than 6 runs will be zero as B had the next strike. Example: $T(1306, 0, 1205) = 0$ and also $T(1312, 0, 1202) = 0$
  - Now when A is at end of over we need to consider that for A to remain in strike it need to score one of these runs = [1,3]. So any other actions which result in any other possibilities will have 0 transition probability to next consecutive state.
  - I had bifurcated the assignment of probabilities in two parts:
    1) If the next reached state is winning state:
    I had further bifurcated it in two parts:
    * If the two states are consecutive in terms of 1 ball left –direct win by only one shot of A:: This is covered by function **'End Probab A'**
    * otherwise:
      · In that case either shot of A can directly lead to win or B will be involved in between

· So when B is involved depending on the run A scored and transferred the strike to B we will have the all possible cases accounted in function defined as **'End Expec probab'**

2) If the next reached state is not winning state:
I had further bifurcated it in two parts:

* If the two states are consecutive in terms of number of balls:
  · Then direct T(s,a,s) will be assigned using $P(r|a)$
* If two states are not consecutive then surely B got the strike:
  · Now since we had to win then B should not get out hence to reach that state it can either score no runs or 1 runs depending whether B is at end of over or mid.
  · Number of balls B will play between two state bb_1rr_1 to bb_2rr_2 would be $bb1 - bb2 - 1$. So first the strike will be given by A to B by scoring some r runs then rest of the runs will be scored by B to reach that state ***if possible*** with constraints of number of balls and runs B can actually attempt would be with probability $P(r|a) \times (\frac{1-q}{2})^{bb1-1-bb2}$
* ***if possible****
  · Expec Probab function in code consider all those possibilities in which B can actually help in transition otherwise it will return 0 expected probability
  · The cases consider the over change and balls left and the constraint that B can score atmost 1 run not more than that, and also the rule that on scoring 1 at end of over strike will be maintained by batsman

– The encoder file will return the formed MDP as per the idea used above and passed to planner to evaluate the optimal policy or value function as per need.

– **NOTE: THE CODE IS COMMENTED AT REQUIRED NECESSARY PLACES**

2.2 Plots Generated

- Clearly the plots are as per intuition that with optimal policy we will always have a greater probability of winning compared to a random policy
- With increasing weakness of B the winning probability decreases
- With increasing number of Balls for a fixed runs the winning probability increases
- With increasing number of Runs to score for a fixed number of balls the winning probability decreases

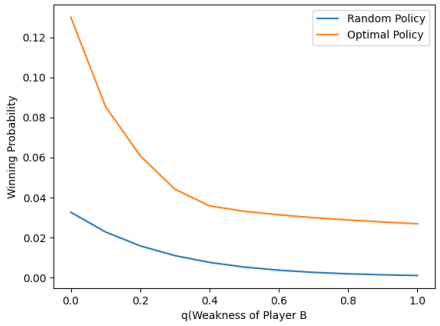

Figure 1: Winning Probability vs Weakness of B(q)

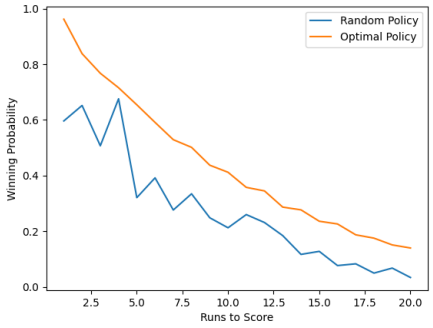

Figure 2: Winning Probability vs Runs to be Scored

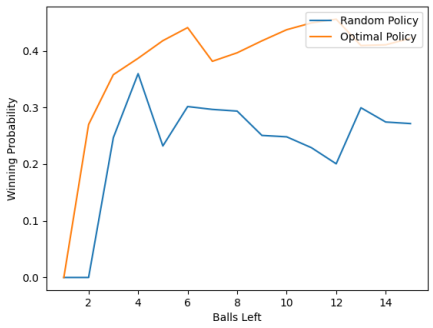

Figure 3: Winning Probability vs Balls Left