Assignment 2 Report

Nikhil Manjrekar

October 2022

Contents

1 Task 1

1.1 Policy Evaluation

Initialisation Implemented Linear Programming for this. Used the `LpVariable.dicts` function of the PuLP library in python to initialise total variables equal to number of states which represents the value of the corresponsing state.

Objective Function Used the objective function given as belows:

$$\min(\Sigma_{s \in S} V(s))$$

Constraints Added the following constraints equations to the solver:

$$V(s) \geq \Sigma_{s \in S} T(s, \pi(s), s1)(R(s, \pi(s), s1) + \gamma V(s1))$$

added the above constraints for all the states. Then found out the optimal policy achieved by finding the action which gives V(s) for each state. Also for the episodic MDP the Value of end states was equatted to zero.

1.2 Value Iteration

Initialisation A numpy array of size `num_states` was used to store the value for each state.

Implementation Applied the B^* on V until norm of difference of V_t and V_{t-1} was less than a threshold. The threshold was set to $1e^{-8}$.

$$\|V_t - V_{t-1}\| < 1e - 8$$

where,

$$(B^*(V))(s) = \max_{a \in A} \Sigma_{s1 \in S} T(s, a, s1)(R(s, a, s1) + \gamma V(s1))$$

1.3 Howard Policy Iteration

Initialisation

- Randomly initialized the policy $\pi(s) \in A(s) \; \forall s \in S$
- Initialised the Value function to be zero for all the states.

Implementation Applied the policy evaluation as described above for this part based on the policy π. Then improved π until no improvement is possible. In each improvement loop, first evaluate the V for the previous π using the same policy evaluation method. Found the best action for each state stored it in π_{new}, and break from the improvement loop if $\pi_{new} == \pi_{old}$

1.4 Linear Programming

Initialisation Used the `LpVariable.dicts` function of the PuLP library in python to initialise total variables equal to number of states which represents the value of the corresponsing state.

Objective Function Used the objective function given as belows:

$$\min(\Sigma_{s \in S} V(s))$$

Constraints Added the following constraints equations to the solver:

$$V(s) \geq \Sigma_{s \in S} T(s,a,s1)(R(s,a,s1) + \gamma V(s1))$$

added the above constraints for all the pairs (s,a). Then found out the optimal policy achieved by finding the action which gives V(s) for each state and then printed V(s) $\pi(s)$ for each state. Also for the episodic MDP the Value of end states was equatted to zero.

2 Task 2

For this task, I have used `2n+2` states to represent the MDP. The first `n` states represented the middle order batsman or the batsman **A**. Then the next `n` states represented the tail-ender. The two extra states represent `win` state and `lose` state. The number of actions for the each states were taken to be five which were, `[0,1,2,4,6]`. So essentially created a transition matrix of size $(2n+2) \times 5 \times (2n+2)$ to store the probabilities of transition from one state upon taking an action a from it. I used the `cricket_state_list.txt` file to get the states corresponding to MDP and used the parameters file to connect two reachable states and assigned probability to the given transition.

Naming convention for the states was as follows:

- States in which the batsman was A were represented by `bbrr0`
- States in which the batsman was B were represented by `bbrr1`

For the creation of MDP I considered the following cases:

- When the remaining runs left is less than or equal to zero then transition to winning state
- When for the current state only one ball is left and the remaining runs after making a transition are still greater than zero then make the transition to losing state
- If the batsman gets out then make a transition to losing state
- If none of the above case then make transition to state `(bb-1)(rr-e)(strike)`, where $e \in \{0,1,2,3,4,6\}$ and $strike \in \{0,1\}$ which was set considering the fact whether $(bb-1)$ mod $6 \cong 0$ (i.e. over has completed or not) and where $e \in \{1,3\}$ (to check if the strike of the batsman changed or not).

I used the above cases for the states of both batsman. The batsman B could only make action -1, 0 and 1 with fixed probabilities if it attempted one of the five actions listed above.

2.1 Plot 1

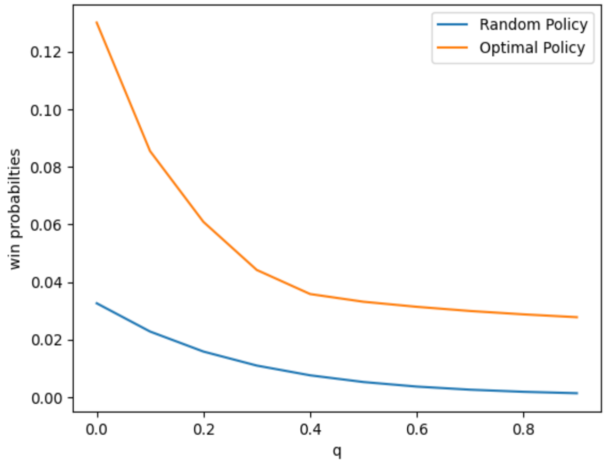

Figure 1: win probability (playing as per the policy) vs. B's ”strength” (q, varied from 0 to 1)

2.1.1 Inferences

- As you can observe, the win probability is decreasing as the value of q approaches 1.
- Decreasing plot is because of the fact that as q tends to 1 then the probability for batsman B to get out if he is on strike increases, and therefore the chances of winning also decreases.
- Random policy plot as it can be seen that it lies below optimal policy plot which is expected.

2.2 Plot 2

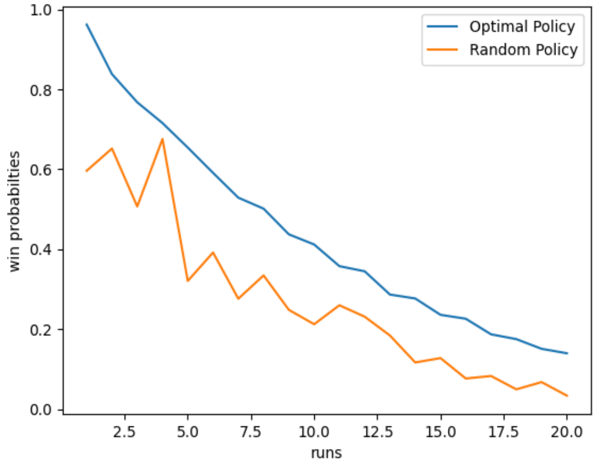

Figure 2: Plot of win probability (playing as per policy) vs. runs. Number of balls was fixed to be 10 in this case and the number of runs required was varied between 1-20

2.2.1 Inferences

- The Plot suggests that the probability to win decreases with an increasing target for the optimal policy as expected.
- The optimal policy is always better than the random policy as expected.
- If we ignore the irregularities in the plot for random policy the overall trend is decreasing owing to the fact that it becomes more hard to win as the runs to score increases
- The random policy is close to optimal policy for runs = 4. This suggests that random policy is almost close to optimal policy.

2.3 Plot 3

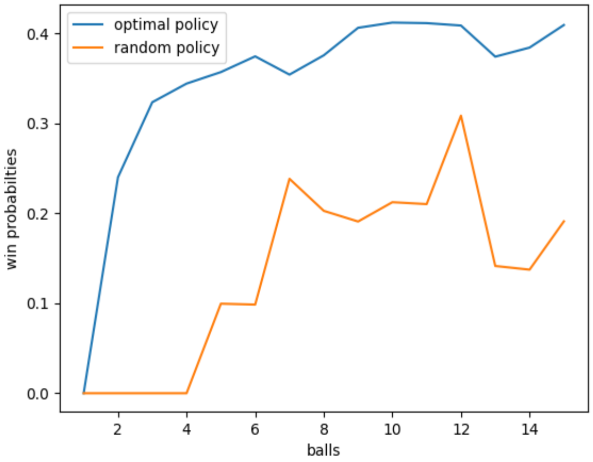

Figure 3: Plot of win probability (playing as per policy) vs. balls. Number of runs required to win were fixed to be 10 in this case and the number of balls left was varied between 1 to 15

2.3.1 Inferences

- As you can see in the plot that the probability to win increases as the number of balls increases which is expected as the more the number of balls left more time batsman has to score the given target which results in increasing win probability.
- A flat line can be seen for the random policy that is the win probability is zero for the given target which was expected as the action in the random policy for the state 0210 is 1 run which will eventually lead to state 0109 or 0110 from where winning is not at all possible
- We can see that the graph is increasing till 6 then as sudden decrease at 7 and again increasing till 12 and as sudden decrease after 12. This result can be think of in this way that when the over is about to change it is better for the batsman A to keep the strike in the next over and thus A has to hit odd number of runs at this state. Such strategy is also common in regular cricket that weaker tail enders are kept away from strike as much as possible