Prateek Garg
20D070060
prateekg@iitb.ac.in

Programming Assignment 2
CS747: Foundations of Intelligent and Learning Agents, Autumn 2023

2023-10-15

Task 1

The implementation of Value Iteration, Howard's Policy Iteration and Value function using linear programming algorithms.

1.a Value Iteration

We initialise the value vector(V^0) with all zeros, and apply Bellman Optimality Operator(B^*) – defined by Transition function T, Reward function R – repeatedly until convergence.
For convergence we consider $||V^{t+1} - V^t|| < \texttt{tol}$, `tol` = 1e-7 and we also consider that $||V^* - V^t|| < \texttt{tol}$, to enforce this, we have

$$\gamma^N \cdot ||V_0 - V^*|| < \texttt{tol}$$

since $||V^* - V^N|| < \gamma^N \cdot ||V_0 - V^*||$

$$N \log \gamma + \log ||V_0 - V^*|| < \log(\texttt{tol})$$

$\because V^0 = 0$

$$\frac{\log(||V^*||/\texttt{tol})}{\log(1/\gamma)} < N$$

for numerical stability, $\epsilon = 1e-7$

$$\frac{\log(||V^*||/\texttt{tol})}{\epsilon + \log(1/\gamma)} < N$$

Since, from our definition V^t has converged to a value($||V^{t+1} - V^t|| < \texttt{tol}$) we check, if that value is indeed optimal within tolerance.

$$\frac{\log(||V^t||/\texttt{tol})}{\epsilon + \log(1/\gamma)} < N$$

We also define `MAX_ITER=1E6` for maximum number of iterations to run before terminating, to ensure algorithm terminate always.
Observation: Value iteration converge very slow if $\gamma = 1$ as expected.

1.b Howard's Policy Iteration

Starting from a random policy π^0, we repeatedly apply policy improvement step and get better policy. Since we need to do policy evaluation in each step, There are two methods – Value Iteration with Bellman operator(B^π), inverting the matrix $I - \gamma \cdot T^\pi$. Matrix Inversion is faster for small number of states but Value Iteration is faster for large number of states. For our implementation, we use matrix inversion. Matrix multiplication becomes non-invertible for $\gamma = 1$, so we set the values of terminal states to 0 and for the rest we calculate using matrix inversion of truncated matrices which doesn't include terminal state transitions.
Observation: Howard's PI is the fastest algorithm out of all 3, and thus our default algorithm as well.

1.c Linear Programming

We use `PuLP` solver to encode this as linear program. The objective is to maximise $-\sum_s V(s)$ subject to some inequality constraints and some equality constraints described below.
Inequality Constraints:

$$V(s) \geq \sum_{s' \in S} T(s, a, s')\{R(s, a, s') + \gamma V(s')\}, \forall s \in S, a \in A$$

Equality Constraints:

$$V(s) = 0, \forall s \in S_{terminal}$$

Equality constraints are required in case $\gamma = 1$

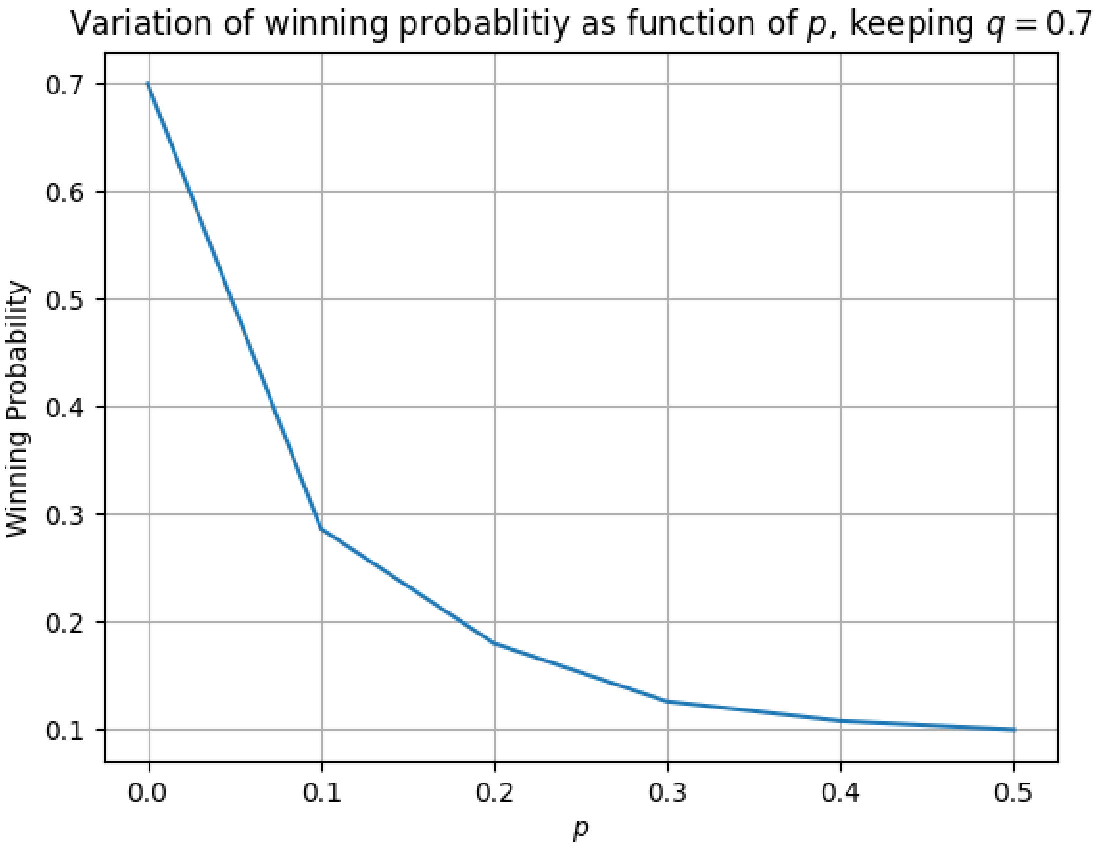
Task 2

Designing MDP for the football game.

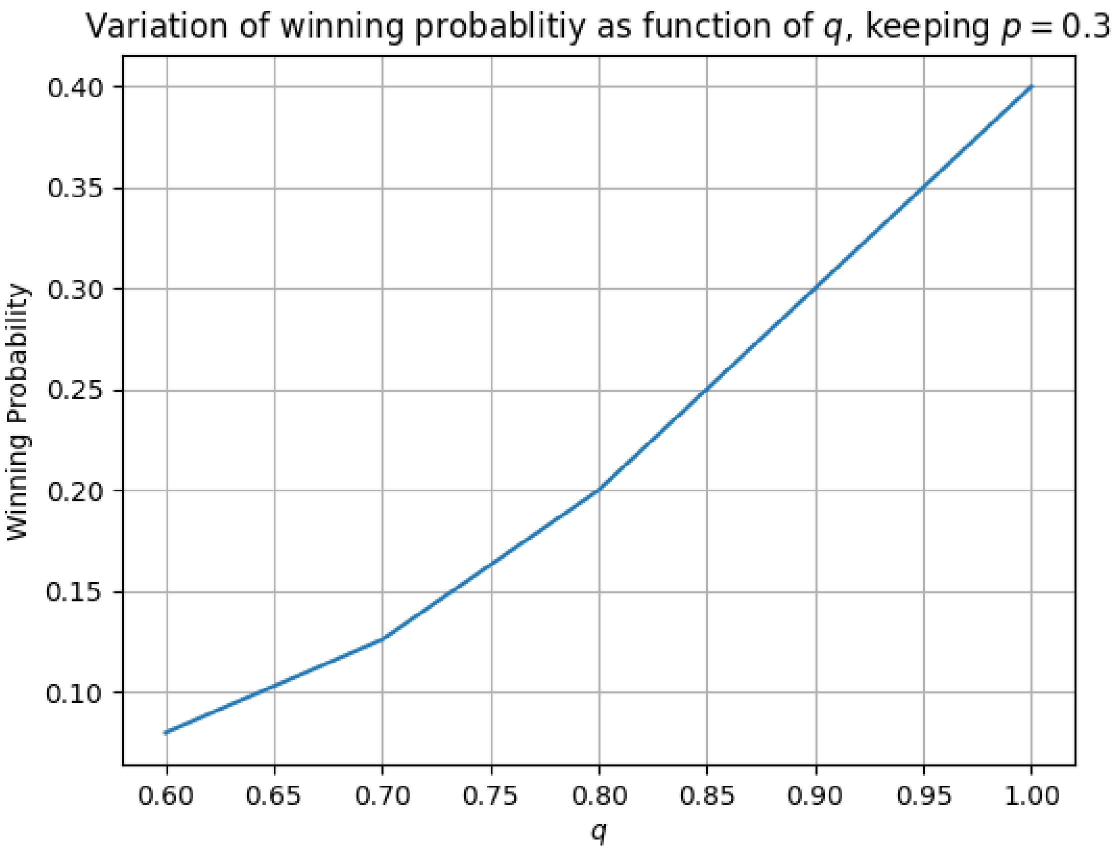
2.a Formulation

We map each state $[B1, B2, R, P]$ to a number s which varies from $[1, 8192]$, we have 2 extra states $\{0, 8193\}$. 0 is the terminal state, which the last state before game ends without a goal. If game ends with a goal, the terminal state is 8193. The transition function is defined by the rules of the game and the corresponding probabilities, once the players decides an action $\{0...9\}$. The reward function is zeros for almost all transitions except when transitioning to $s = 8193$, when the reward is 1. The expected reward of each state gives us the value function which is also the probability of winning(since $\mathbb{E}[r] = p\dot{1} + (1-p) \cdot 0 = p$) given the starting state is that particular state. This is all done in `encoder.py`

2.b Comparison & Inferences



Since as p increases the probability of failure of an attempted movement increases($\{2p, 0.5 + p, p\}$ are probabilities that the game ends depending upon the case), thus the probability of winning decreases.



Since as q increases the probability of success of an attempted pass or goal increases (for passing $\{0.5 * (q - 0.1 * \max(|x_{B1} - x_{B2}|, |y_{B1} - y_{B2}|)), q - 0.1 * \max(|x_{B1} - x_{B2}|, |y_{B1} - y_{B2}|),\}$ are probabilities that the attempted pass succeeds, $\{q - 0.2 * (3 - x_{distance}), 0.5 * (q - 0.2 * (3 - x_{distance}))\}$ depending upon the case), thus the probability of winning increases.