CS 747: Programming Assignment 2

Vinayak K (150050098)

8 Sep 2018

Files

* MDP solving methods (Linear programming and Howard's Policy Iteration) are implemented in **extras/mdp.py**.
* **extras/gambler.py** creates MDP file corresponding the gambler's problem.

MDP Formulation of Gambler's Problem

* **States**: Gambler's capital, from 1 to 99. And two extra terminal states 0 (lose) and 100 (win).
* **Actions**: Bet stakes. At state s, valid actions are $\{1, 2, ..., min(s, 100 - s)\}$.
* **Rewards**: for any state s and a valid action a, if $s + a = 100$, then reward $R[s][a][100] = 1.0$. Else 0.0.
* **Transition Probs**: for any state s and a valid action a, $T[s][a][s + a] = P_h$ and $T[s][a][s - a] = 1 - P_h$.
* **Discount factor**: 1.0, since reward is given only at the end of the episode.

Invalid state, action pairs will have zero transition probability so that it will never be taken. Since a reward of 1.0 is obtained only if state 100 is reached, value function in this setting will correspond to *probability of winning from each state*. A policy is a mapping from levels of capital to stakes. The optimal policy maximizes the probability of reaching the goal.

Effect of P_h value

When P_h value is higher, even at low capital there is a high chance of reaching the goal and that is reflected correctly in the plots of value function. Also for larger P_h, optimal policy is to bet just 1 for small capitals.

Plots

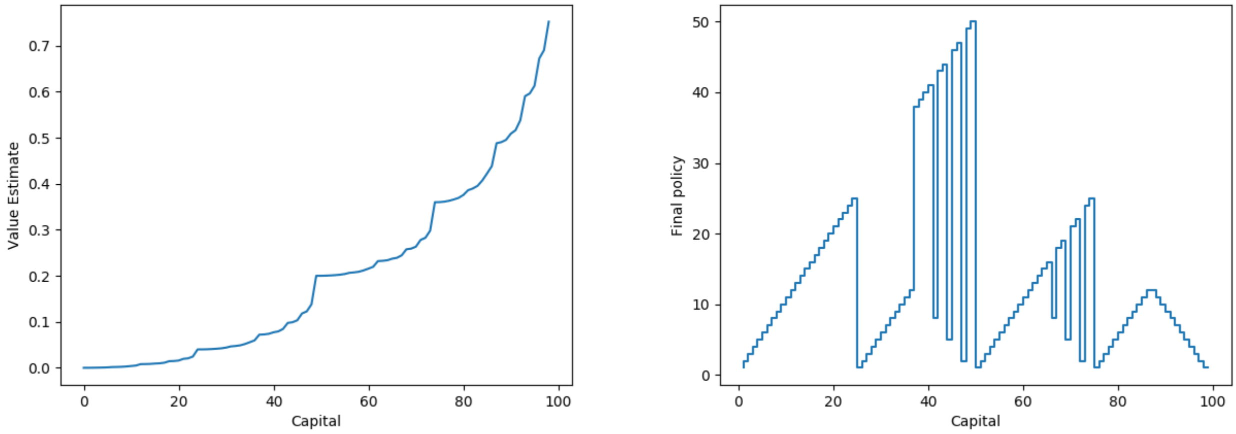

Figure 1: $P_h = 0.2$

Big spikes/troughs seen in the graph could be caused by noise and a result of roundup errors of the floating point limitations of CPUs. Also the policy is optimal, but not unique. In fact, there is a whole family of optimal policies, all corresponding to ties for the argmax action selection with respect to the optimal value function.

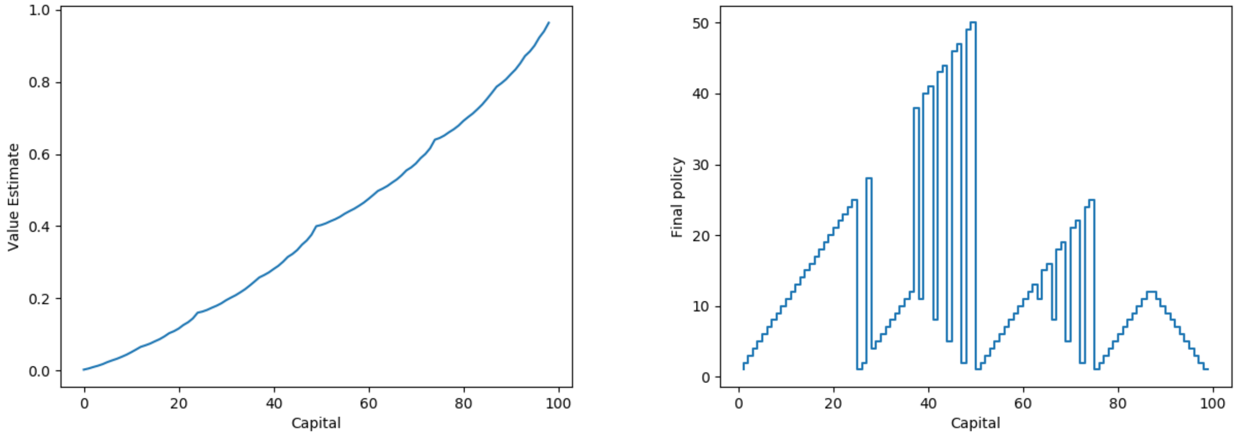

Figure 2: $P_h = 0.4$

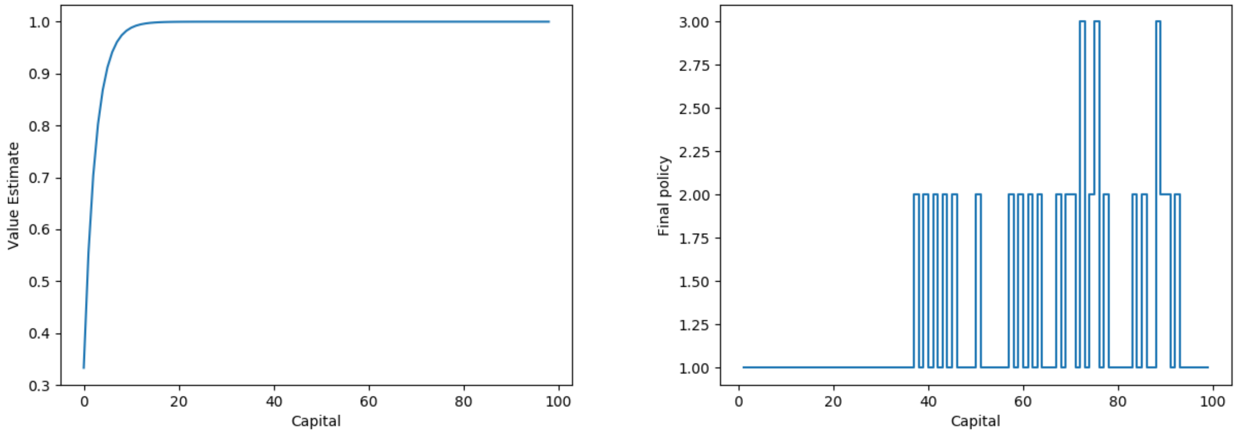

Figure 3: $P_h = 0.6$

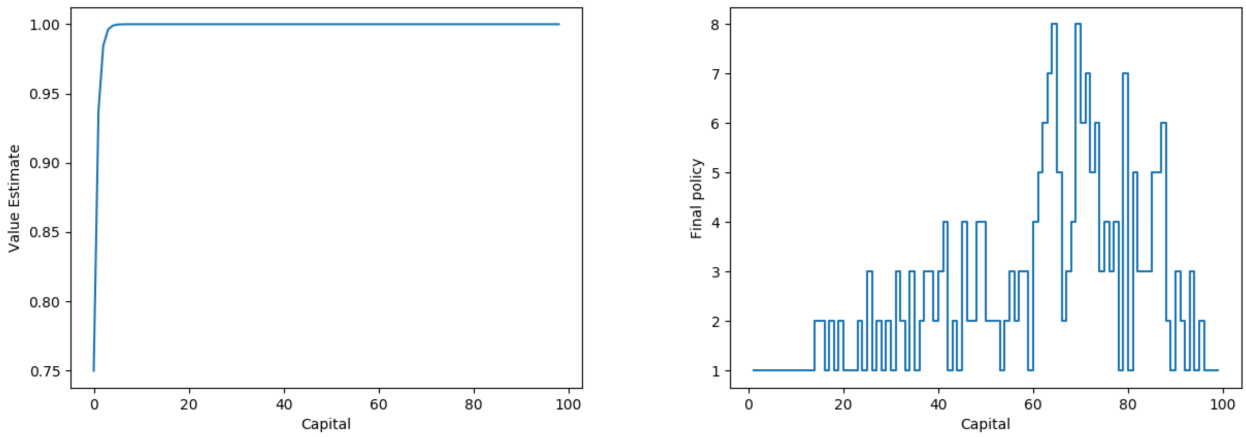

Figure 4: $P_h = 0.8$