Reinforcement Learning Assignment 2

James Giroux
Department of Data Science
William & Mary
Williamsburg, VA 23185
jgiroux@wm.edu

Abstract

Deploy Monte-Carlo (MC) Control and Sarsa(λ) in an Easy21 environment. Easy21 functions similarly to a traditional blackjack game, with some unique twists. Specifically, the deck of cards is infinite and sampled with replacement, face cards, and aces do not exist, and the color of cards indicates whether subtraction or addition occurs in the card summation.

1 Easy21

The Easy21 environment functions similarly to traditional blackjack with some differences. The game starts with the player and dealer both drawing a single card, in which we define the color to be black. Black cards correspond to positive values in the summation. Red cards will correspond to subtraction. The probability of black or red is defined as $2/3$ and $1/3$, respectively. As such, a summation greater than 21 will result in a bust, along with a negative summation. The dealer follows a fixed policy such that they stick at any value greater or equal to 17.

The code base functions off a configuration file (config.json), which specifies the following fields:

- "gamma": 1.0,
- "lambda_values": [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0],
- "episodes": 500000,
- "run_MC": 1,
- "run_sarsa": 1

The "run" fields are Boolean flags to run specific algorithms. If the MC algorithm is not run, only certain Sarsa plots will be generated. The code will run Sarsa(λ) for all values specified. Generated figures will be placed in the newly created Figures folder. If the folder already exists, *i.e.*, you have run the code before, it will overwrite existing files or create new ones in the folder depending on the configuration. Once the configuration file has been set, the code can be run with the following command:

python run.py –config config.json

2 Monte-Carlo Control

We deploy Monte-Carlo (MC) control with an epsilon greedy policy on the Easy21 environment. Fig. 1 depicts the optimal value function after 500k episodes.

From the figure, it can be seen that the highest expected return occurs when the players' hand is closer to 21. This is relatively uniform as a function of the dealers' hand. The expected reward also

Preprint. Under review.

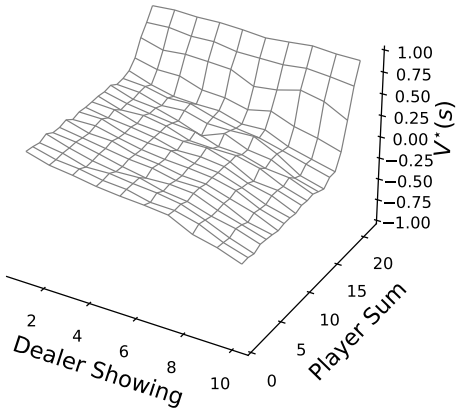

Figure 1: Optimal Value function over the state space for Monte-Carlo control.

decreases as the dealers' hand increases. At the low value of the dealers' hand, the expected return increases being that it is more likely they will bust from a negative summation.

3 Sarsa(λ)

We deploy Sarsa(λ) with an epsilon greedy policy on the Easy21 environment. Fig. 2 depicts the optimal value function after 500k episodes, for $\lambda = 0$ (left) and $\lambda = 1$ (right).

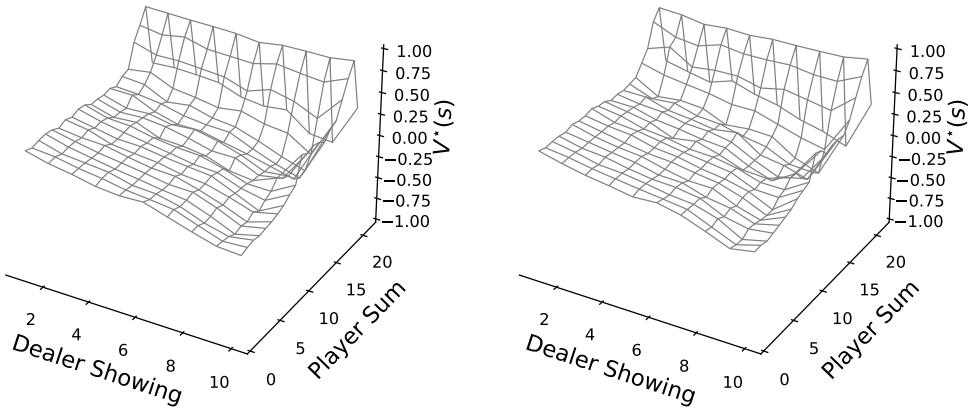

Figure 2: Optimal value function over the state space for Sarsa(λ), where $\lambda = 0$ (left) and $\lambda = 1$ (right).

There exists a slight difference in the value functions, albeit minor after half a million episodes. The value function for $\lambda = 1$ possesses a steeper gradient towards higher values of the player's hand. Value function plots for all values of λ can be found in the Figures folder.

Fig. 3 depicts the mean squared error (MSE) between the optimal state value function ($Q^\star$) from Monte-Carlo control, and different lambda values for Sarsa(λ) as a function of episodes (left), and final MSE as a function of lambda (right) for 500k episodes.

From the left plot, it can be seen that larger values of lambda correspond to larger initial values of MSE and larger final values. Given we have run a significant amount of iterations and the problem is relatively simple, the values converge to similar minimums in the end. In the right plot, where we

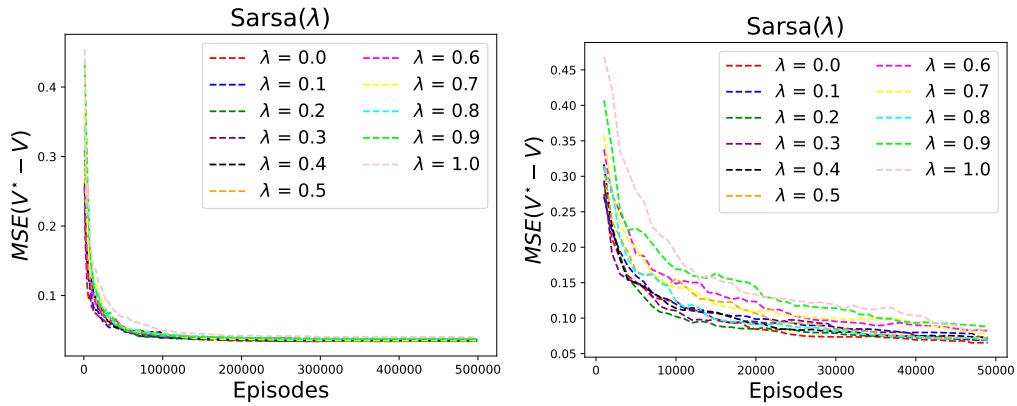

Figure 3: Mean squared error (MSE) between the optimal state value function ($Q^\star$) from Monte-Carlo control, and different lambda values for Sarsa(λ) as a function of episodes, for 500k episodes (left) and 50k episodes (right).

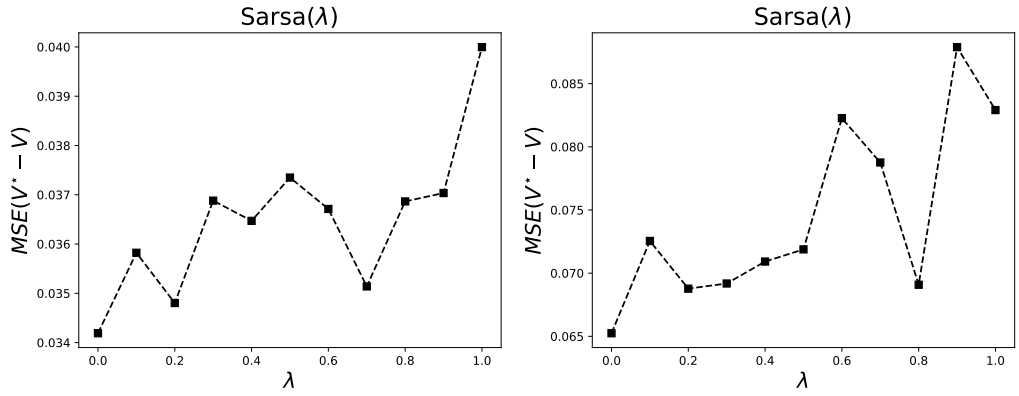

Figure 4: Mean squared error (MSE) between the optimal state value function ($Q^\star$) from Monte-Carlo control, and different lambda values for Sarsa(λ) as a function of episodes, for 500k episodes (left) and 50k episodes (right).

decrease the number of iterations, it can more easily be seen that MSE decreases faster, and converges to lower minimums as lambda decreases. This trend can also be seen in Fig. 4, in which the MSE increases as a function of lambda, for both 500k (left) and 50k episodes (right).

4 Discussion

Bootstrapping introduces a high level of bias into the RL algorithm, as opposed to full MC backups which are high in variance. High bias is problematic in the sense that the model can fall into undesirable states given biased experiences. On the other hand, high variance may inhibit convergence of the model and require more episodes for convergence. In Easy21, bootstrapping could lead to a biased policy.

Being that we are sampling with replacement, and introducing the operation of subtraction to the game, bootstrapping would benefit Easy21 more. The average length of an episode for blackjack would be shorter since we start with two cards, and are limited to only addition. Therefore, we can learn more in a single episode in Easy21 providing faster convergence.