Reinforcement Learning Assignment: Easy21

William Ferreira

March 30, 2015

1 Implementation of Easy21

The game environment is implemented by the function `step(s, a)` in the file `question1.py`, where `s` is a tuple containing the dealer's initial card as first element and the player's current score as the second element, and `a` is an action: `'stick'` or `'hit'`. Calling `step(s, 'hit')` will advance the player's score by drawing a card, with replacement, from the deck and adding it to the player's current score, and returning a new environment and reward. If the player's new score is < 0 or > 21 then the player loses with a terminal reward of -1; otherwise play can continue. Calling `step(s, 'stick')` will play out the dealer's hand until a terminal state is reached, with a score of -1 if the dealer wins, 0 if the game is a draw, and 1 if the player wins.

2 Monte-Carlo Control in Easy21

Monte-Carlo Control in Easy21 is implemented by the class `Easy21MCControl` in the file `question2.py`. Figure 1 contains a surface plot of the optimal value function $V^*(s) = \max_a Q^*(s, a)$ with $100,000,000$ episodes and $N_0 = 100,000$. The code can be run in the following ways:

- `python question2.py` - runs the code for default values of $100,000,000$ episodes and $N_0 = 100,000$ and plots the value function; WARNING: this can take some time.
- `python question2.py --episodes=<num episodes> --N0=<n>` - runs the code for <num episodes> episodes and $N_0 = $ n and plots the value function.
- `python question2.py --plot=mc_v.csv` - plots the surface in Figure 1 using pre-computed data.

Choosing a value of N_0 that scales with the number of episodes seems to produce a smoother surface.

3 TD Learning in Easy21

TD Learning in Easy21 is implemented by the class `Easy21Sarsa` in the file `question3.py`. The plots of MSE against $\lambda \in \{0, 0.1, \ldots, 1\}$ and the learning curve MSE against episode number are shown in Figure 2. The code can be run in the following ways:

- `python question3.py` - runs the code for default values of 1,000 episodes and $N_0 = 100$ with MSE estimated using the $Q(s, a)$ function estimated in part 2, and plots the results.
- `python question3.py --episodes=<num episodes> --N0=<n>` - as above but with specified values of <num episodes> episodes and $N_0 = \text{n}$.

4 Linear Function Approximation in Easy21

Linear Function Approximation in Easy21 is implemented by the class

`Easy21LinearFunctionApprox`

in the file `question4.py`. The plots of MSE against $\lambda \in \{0, 0.1, \ldots, 1\}$ and the learning curve MSE against episode number are shown in Figure 3. The code can be run in the following ways:

- `python question4.py` - runs the code for a default value of 1,000 episodes with MSE estimated using the $Q(s, a)$ function estimated in part 2, and plots the results.
- `python question4.py --episodes=<num episodes>` - as above but with specified values of <num episodes> episodes.

5 Discussion

- *What are the pros and cons of bootstrapping in Easy21?*

  Easy21 episodes can become quite long, and so Monte Carlo methods, which are non-bootstrapping, have to wait a long time before the result is is known; whereas, bootstrapping methods only need to wait for one time-step to get a result.

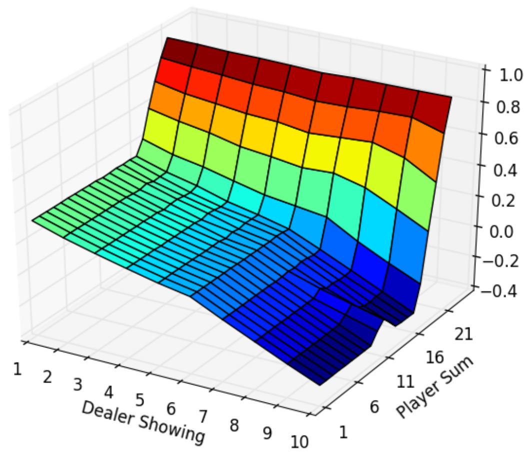

Figure 1: Monte-Carlo Control in Easy21. The plot shows the optimal value function $V^*(s) = \max_a Q^*(s, a)$ with 100, 000, 000 episodes and $N_0 = 100, 000$.

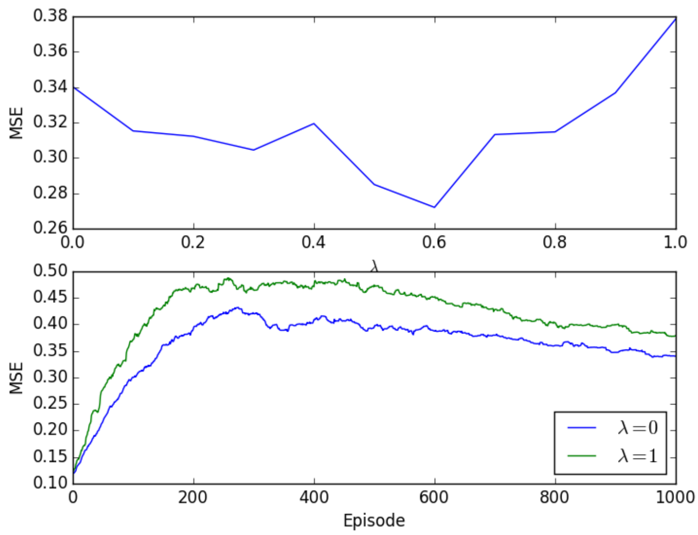

Figure 2: Sarsa(λ) in Easy21. Top plot: λ against MSE. Bottom plot: learning curve episode number against MSE, with 1, 000 episodes and $N_0 = 100$.

- *Would you expect bootstrapping to help more in blackjack or Easy21? Why?*

  Bootstrapping should help more in Easy21. This is because bootstrapping builds estimates based on estimates, and in Easy21, unlike in Blackjack, states can be re-visited which should improve accuracy and convergence.

- *What are the pros and cons of function approximation in Easy21?*

  The pros of function approximation in Easy21 are a smaller memory footprint due to fewer features than states; a more accurate representation of the optimal value function in fewer iterations. The cons are the challenge in finding an accurate feature representation of the state space.

- *How would you modify the function approximator suggested in this section to get better results in Easy21?*

  The function approximator could be improved by making it smoother and more granular by introducing additional features.

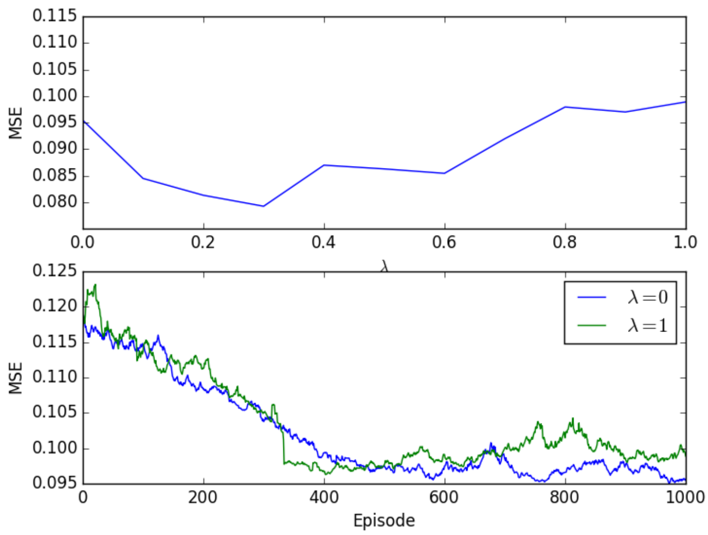

Figure 3: Linear Function Approximation in Easy21. Top plot: λ against MSE. Bottom plot: learning curve episode number against MSE, with $1,000$ episodes.