Programming Assignment 3: Solving Blackjack

Robert D. Grant

1 The problem

For this assignment I experimented with Example 5.3 from Sutton and Barto: Solving Blackjack through Monte Carlo ES.

The authors describe the problem fairly thoroughly in Examples 5.1 and 5.3. Blackjack is described as an MDP with state formed by three variables:

1. Does the player have a usable ace?
2. What card is the dealer showing?
3. What is the current sum of the player's cards?

Excluding the range of current_sum where there is no decision to be made, the remaining ranges of these variables are as follows:

- usable_ace (True, False)
- current_sum (12-21)
- dealer_showing (A-10)

The player's only actions are to "hit" or "stick".

2 Implementation

In my implementation, I represent the Q function as a 4-D matrix, with each of the first three dimensions corresponding to a state variable, and the final dimension corresponding to the action. As in the example, I compute state-action values as incremental sample averages.

As it was not specified, in my experiments I assume the dealer counts aces as 11 unless he busts, whereupon they become valued as 1.

Each episode begins by choosing a coordinate in the state-action space though a given strategy, and then continuing a standard game of blackjack from that point. After the player sticks, the dealer completes his run and the reward is determined.

3 Experiments

In my experiments, I compare the convergence speed of the following:

1. Merely choosing a random state value (always following policy for the action), denoted rnd_states
2. Choosing a random state-action pair (traditional exploring starts), denoted rnd_pairs
3. Enumerating each state-action pair in order, denoted enum_pairs

My hypothesis was that the first would not converge to the optimal policy at all (as it would not visit all the states), and the last would converge more quickly than the second (as its sweeps through the state space start with less commonly visited states).

4 Results

In my figures I plot policies in red and blue, where red indicates "stick" and blue indicates "hit". All value functions are plotted as heat maps, with lighter colors indicating higher values.

My hypothesis about rnd_states was correct: it did not converge to anything like an optimal policy. After looking at the counts of it's visits to each state, it was obvious why. It does not visit all states, and many states very few times. Below is a slice of a table of state visits for each strategy, after 1e6 episodes:

rnd_states:

```
8889,     1,  8466,     1,     5,  7118,  9009,  8392,  9138, 8362
9411,     1,     1,  8046,  7573,  7931,     1,     4,  9376, 7566
   5,     1,     1,  8385,  7736,  9581,     4,     1,     5, 9354
   1,     1,  8560,  7164,     1,  9997,  8552,  8269,  9484, 9600
```

```
8990,     1,  7306,  9072,     1,     1,     4,     1,  9837, 1
   3,     2,     1,     1,  6075, 10138,  8497,     1,  8469, 1
   1,  5534,  8960,  7585,     1, 10369,  7055,     1,     1, 1
   1,     1,     1,     1,     3,     2,     1,     2,     1, 4
   2,     0,  7467,     0,     0,     0,     0,     0,     1, 0
   0,     0,     0,     0,     0,     0,     0,     0,     0, 0
```

rnd_pairs:

```
5211, 5354, 5295, 5174, 5298, 5306, 5330, 5225, 5293, 5337
5501, 5614, 5446, 5480, 5583, 5548, 5426, 5591, 5512, 5645
5748, 5738, 5819, 5829, 5871, 5521, 5716, 5703, 5871, 5890
6095, 6180, 6016, 5984, 6144, 5881, 6028, 6004, 6057, 6166
6363, 6356, 6240, 6298, 6161, 6257, 6290, 6363, 6481, 6489
6155, 3236, 2681, 3071, 2546, 2750, 2680, 6565, 6906, 6789
2395, 2481, 2551, 2446, 2571, 2542, 2474, 2559, 2513, 2471
2513, 2444, 2476, 2438, 2439, 2526, 2491, 2548, 2445, 2453
2555, 2480, 2533, 2508, 2537, 2556, 2553, 2549, 2535, 2521
2481, 2415, 2401, 2586, 2564, 2533, 2524, 2423, 2536, 2496
```

enum_pairs:

```
5329, 5320, 5373, 5354, 5255, 5280, 5276, 5364, 5387, 5336
5626, 5580, 5622, 5606, 5581, 5499, 5424, 5449, 5601, 5479
5926, 5866, 5820, 5908, 5871, 5671, 5770, 5859, 5922, 5792
6149, 6103, 6148, 6130, 6001, 5914, 6035, 6204, 6135, 6161
6409, 6464, 6434, 6406, 6297, 6228, 6404, 6457, 6419, 6422
6417, 6100, 5922, 2508, 2554, 2513, 2506, 6056, 6397, 6557
2503, 2500, 2502, 2501, 2500, 2501, 2502, 2505, 2523, 2524
2501, 2500, 2500, 2500, 2501, 2501, 2500, 2500, 2500, 2500
2501, 2500, 2500, 2500, 2500, 2500, 2500, 2501, 2500, 2500
2500, 2500, 2500, 2500, 2501, 2500, 2500, 2500, 2500, 2500
```

rnd_states is the least even, while enum_pairs is the most even.

rnd_pairs and enum_pairs both seemed to converge; however, they did not perform significantly differently from each other. After a low number of episodes, such as 1000 in the figures below, it's possible that enum_pairs could be ahead; if there is any real difference, it is slight. Though my "Usable Ace" plots look very similar to Figure 5.5 in Sutton and Barto, none of my "No Usable Ace" plots exhibit the large notch as seen in that figure. I can only assume the authors set up some parameter differently than I did; perhaps their dealer's strategy is different in some way.

Plots of policy and value function convergence are shown below.

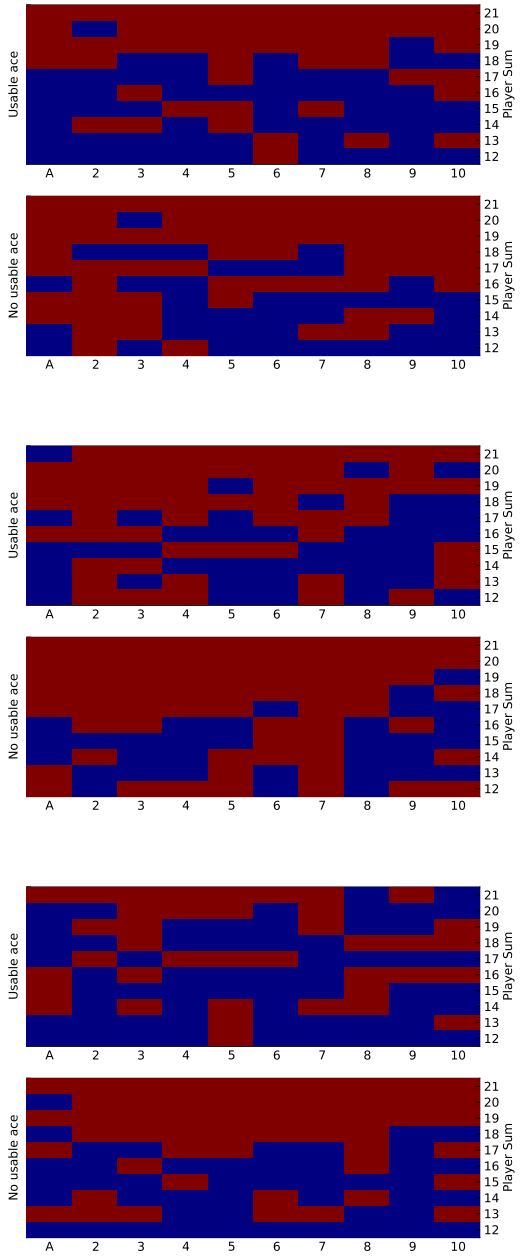

Figure 1: policy with rnd_states, rnd_pairs, and enum_pairs after 1000 episodes

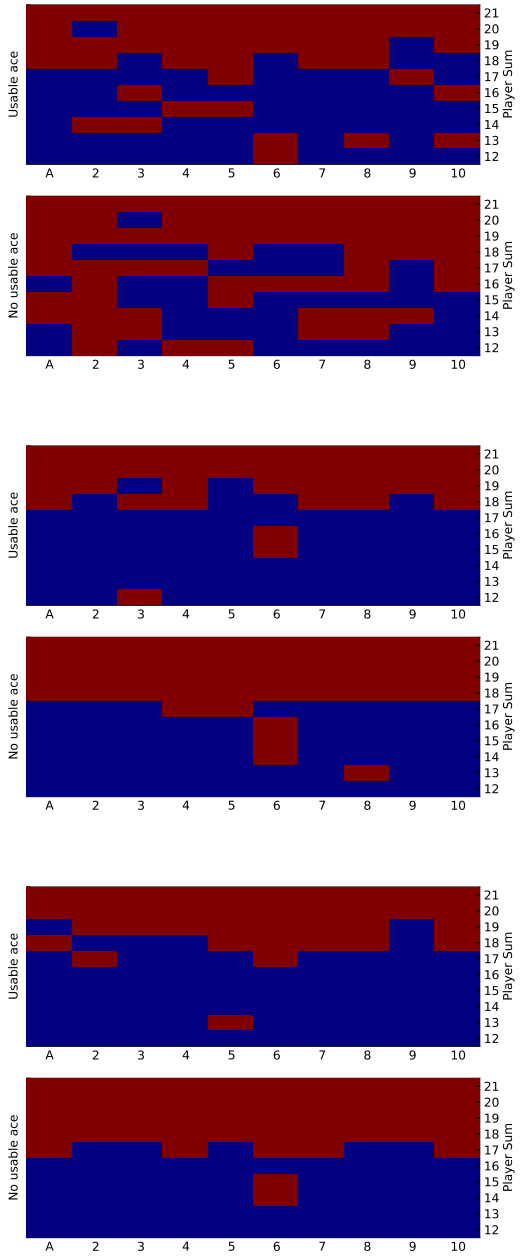

Figure 2: policy with rnd_states, rnd_pairs, and enum_pairs after 10000 episodes

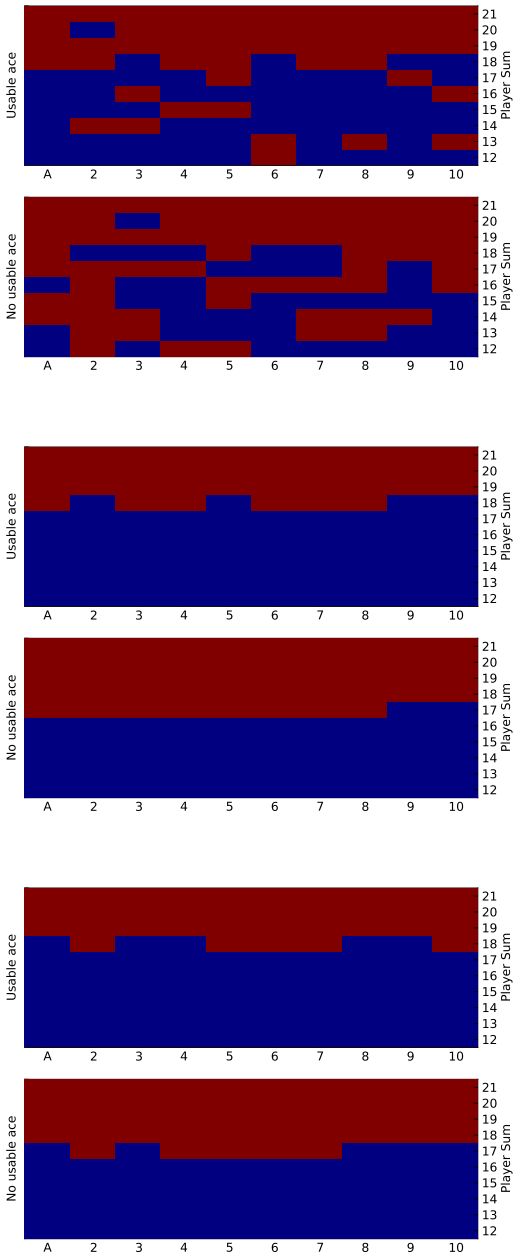

Figure 3: policy with rnd_states, rnd_pairs, and enum_pairs after 100000 episodes

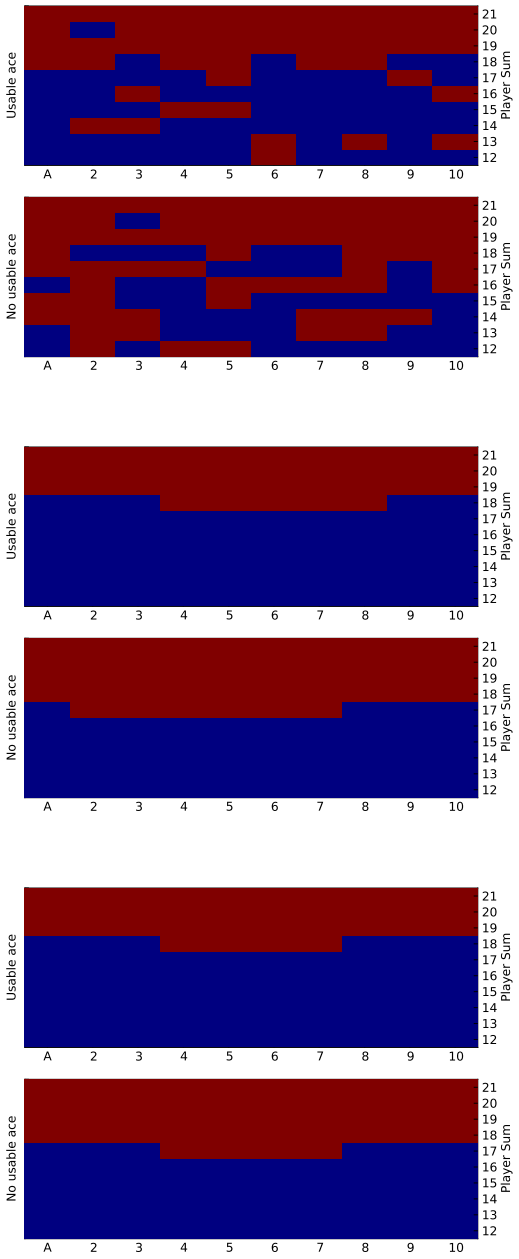

Figure 4: policy with rnd_states, rnd_pairs, and enum_pairs after 1000000 episodes

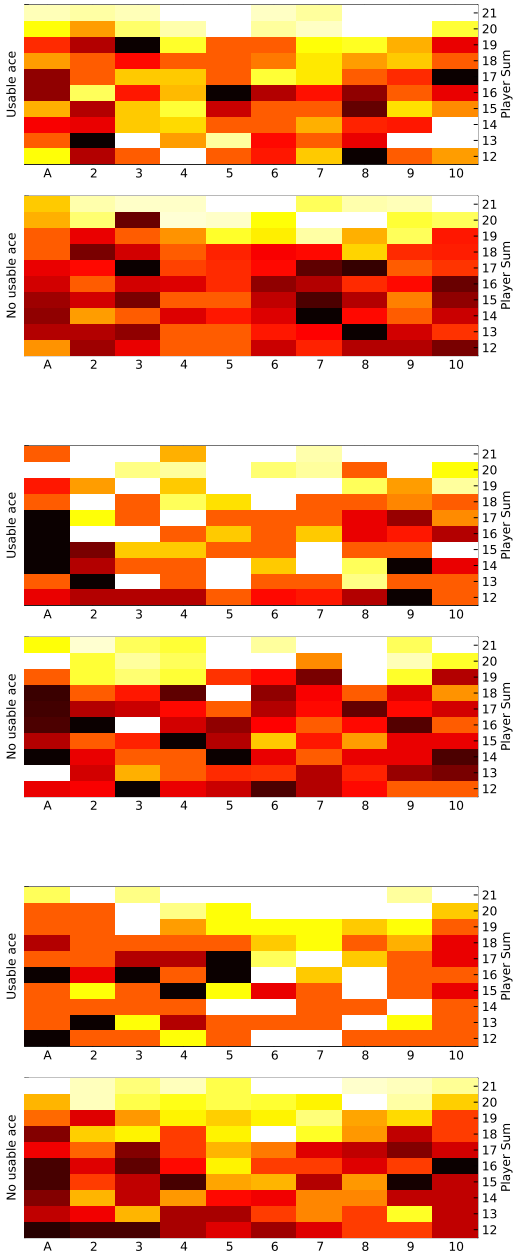

Figure 5: value with rnd_states, rnd_pairs, and enum_pairs after 1000 episodes

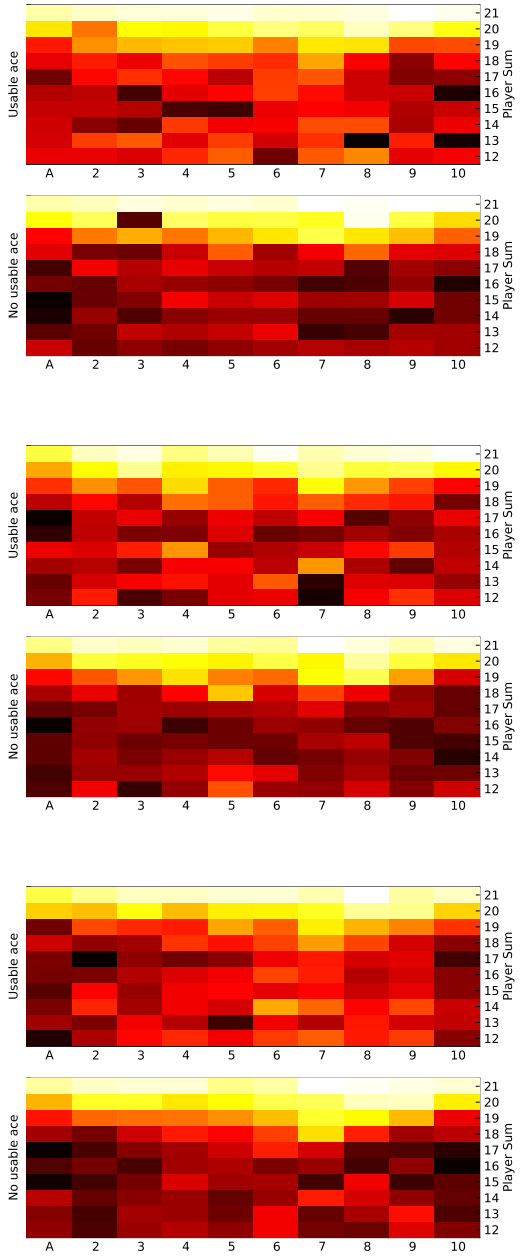

Figure 6: value with rnd_states, rnd_pairs, and enum_pairs after 10000 episodes

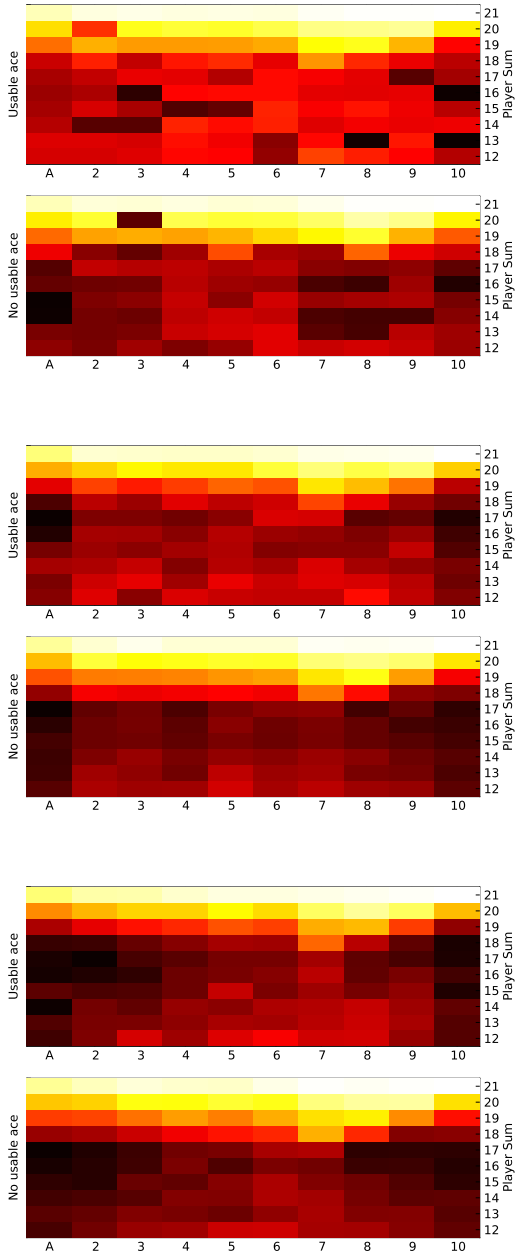

Figure 7: value with rnd_states, rnd_pairs, and enum_pairs after 100000 episodes

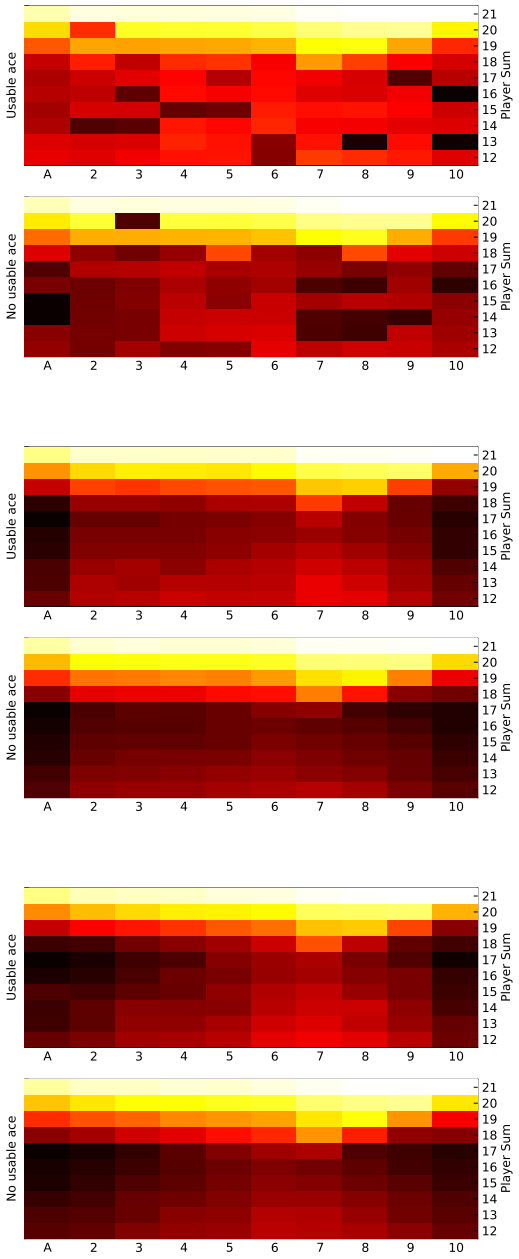

Figure 8: value with rnd_states, rnd_pairs, and enum_pairs after 1000000 episodes