COL870 A1 Report

Shashwat Shivam (2016CS10328)

October 4, 2019

1 State Space

1.1 State Representation

The state representation consists of the raw total (not invoking special cards) of the players card, three booleans indicating the presence of atleast one card of each of the special cards (Ace, Card 2 and Card 3) and the dealer's initial card.

The representation of state in the code can be seen as follows :-

```
def reset(self):
    """
    Reset State and start new game
    """

    self.state = {
        'total': 0, # -30 to 31
        'trump1': 0, # 0 or 1
        'trump2': 0, # 0 or 1
        'trump3': 0, # 0 or 1
        'dealer_card': 0, # 1 to 10
    }
```

The **'total'** field can vary from -30 (in presence of all 3 special cards) to 31. The **'trump'** fields can be 0 or 1 depending on the presence of atleast one special card of that type. The **'dealer_card'** field is allowed to vary from 1 to 10.

Apart from these states we have terminal states for **'win'**, **'lose'** and **'draw'**.

1.2 Valid States

All states in which it is not possible to attain a score between 0 and 31 (inclusive) , with or without the use of special cards are discarded and not considered valid states.

1.3 Unactionable States

During the start of the game if any of the side (player or dealer) gets a negative card, there is no possible action to be taken and these states are unactionable states (these are not

considered in our simulator).

When a player achieves a score of 31 (in any possible combination of values of existing cards), the player is forced to stick and thus such states are made unactionable states. Sticking is the best choice rationally as a score higher than 31 is not possible.

1.4 Compressed States

For better results similar states are combined into one. These are the states in which **total number of special cards is equal** along with equal total and dealer card (e.g :- state with 1 Ace is equivalent to 1 Card 2).

After compressing states we have a total of **1883 states**.

2 Simulator

A simulator was built according to the specifications provided.

A small change was made in the simulator to improve learning speed. The unactionable states which are generated at the start of the game (negative cards in starting) was not simulated.

3 Policy Evaluation

All graphs follow policy same as dealer (hit until score of 25 or above is obtained).

3.1 General Observations

First let us observe the best policy graphs obtained for insights in relation to the policy followed. These graphs are for 1000 step TD based method averaged over 1000 runs.

- There are several areas with Q value 0. These are the areas which are never visited due to the fixed nature of the policy followed. (Stick graphs' extended portions have 0 Q values).
- We can observe that all of the stick graphs are similar. This is due the fact that rewards after stick do not depend on the number of special cards in hand or the raw total. These only depend on the final score achieved which is same for multiple points in these graphs.
- In the negative regions in the hit graphs, player is forced to use special cards to prevent from going bust. This results the player being in a similar state as in the 0-15 region with no trumps. This can be easily seen with the similar nature of the graphs.
- The 3 trump graphs look different from the other graphs due to the fact that these are really low probability states and propagating rewards from these states occurs very slowly.

- A peak in value is seen in hit graphs at dealer card value 4. This is due to the reason that near this range the dealer has the highest chance of going bust due to its policy.
- In the stick graph we can see a lower value for lower total scores. This is due to the fact that the lower the score at which player sticks, the higher the chance of dealer winning (as they are following the same policy and the dealer may get a higher score with more probability).

1000 Step TD averaged over 1000 runs

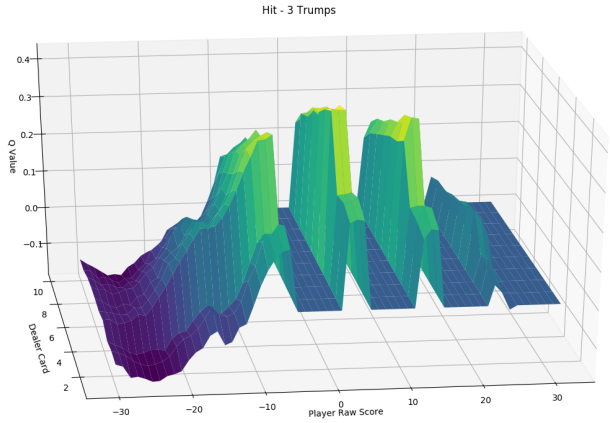

Figure 1: Hit - 3 Trumps

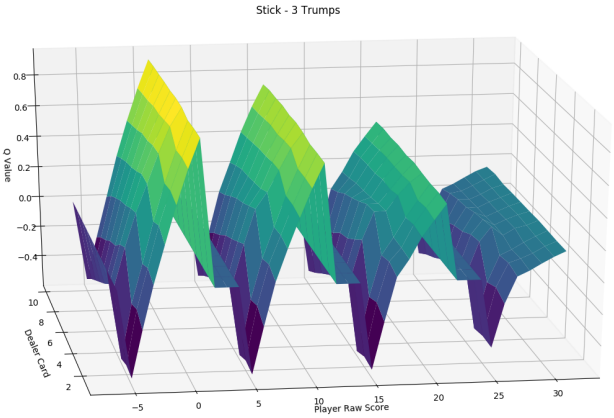

Figure 2: Stick - 3 Trumps

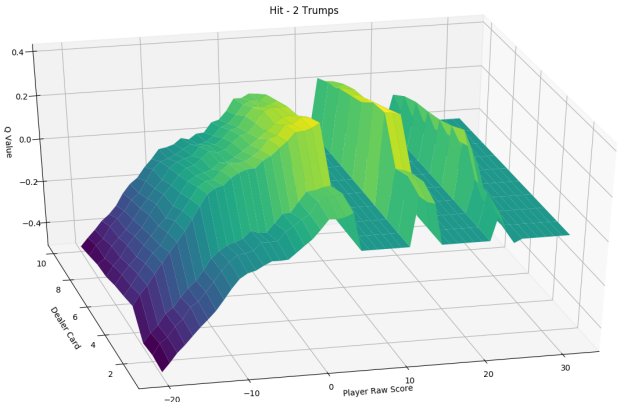

Figure 3: Hit - 2 Trumps

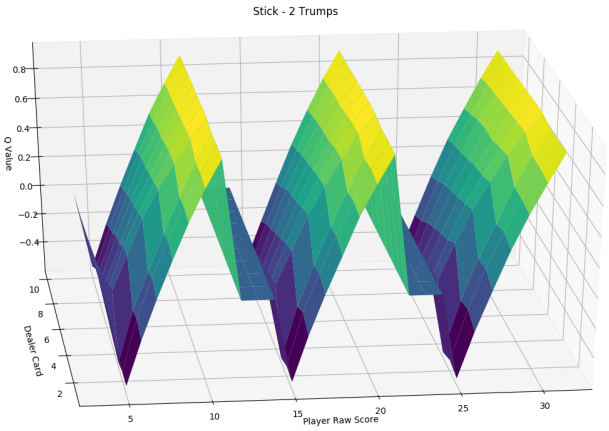

Figure 4: Stick - 2 Trumps

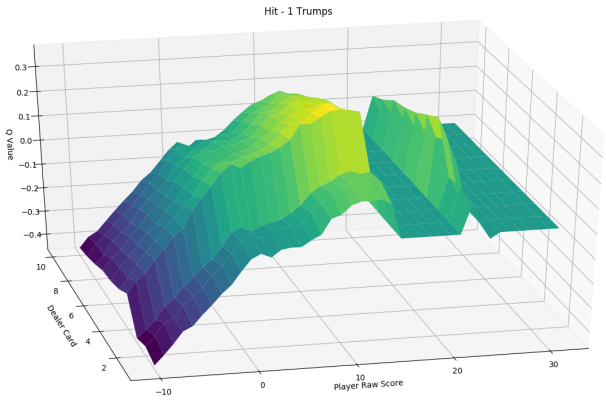

Figure 5: Hit - 1 Trump

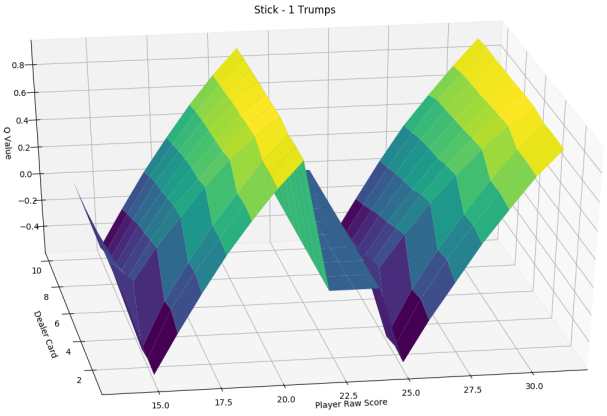

Figure 6: Stick - 1 Trump

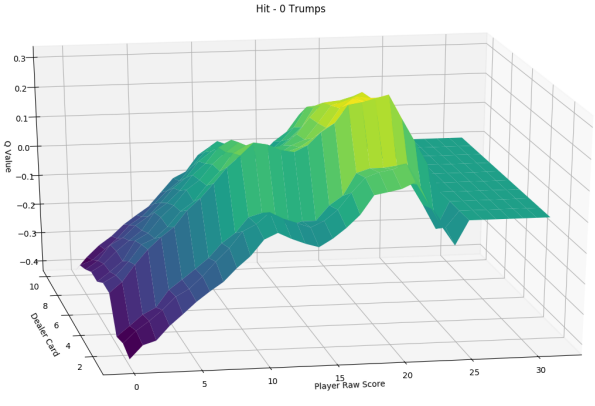

Figure 7: Hit - 0 Trump

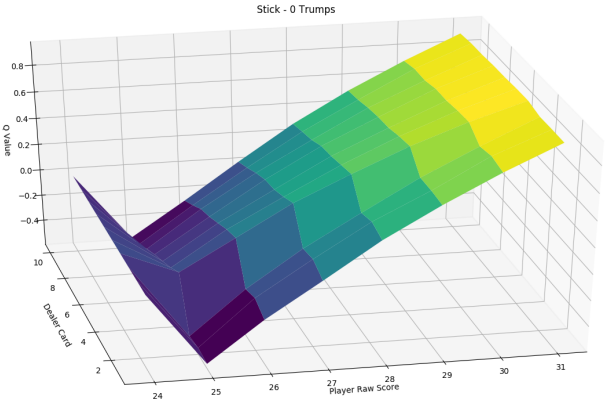

Figure 8: Stick - 0 Trump

For clarity of comparison I report only lower probability states i.e. 3 trump states from here on. Complete set of graphs can be found Here. .

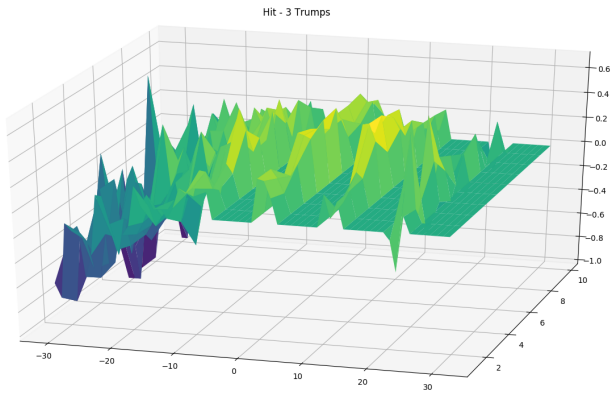

Figure 9: MC Every Visit - Hit - 3 Trumps

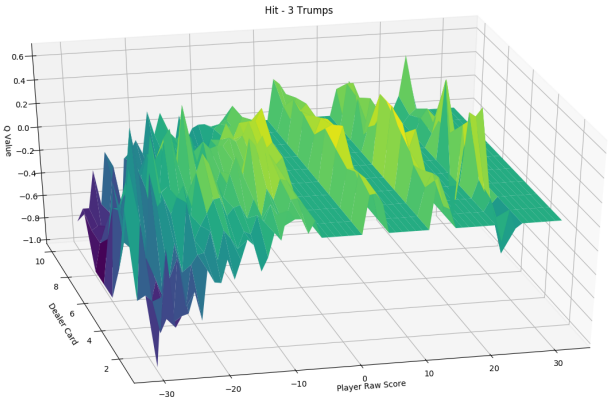

Figure 10: MC First Visit - Hit - 3 Trumps

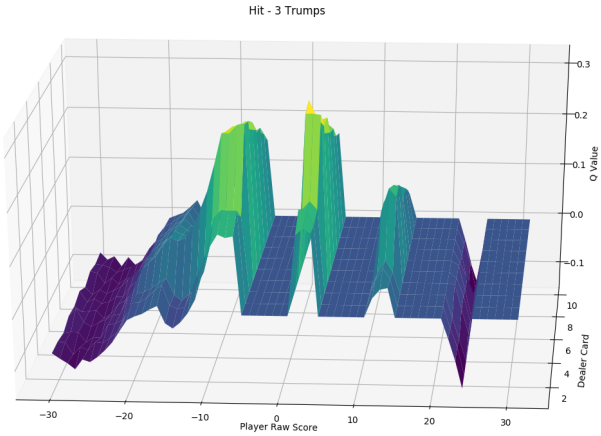

Figure 11: 1 Step TD Averaged over 100 Runs - Hit - 3 Trumps

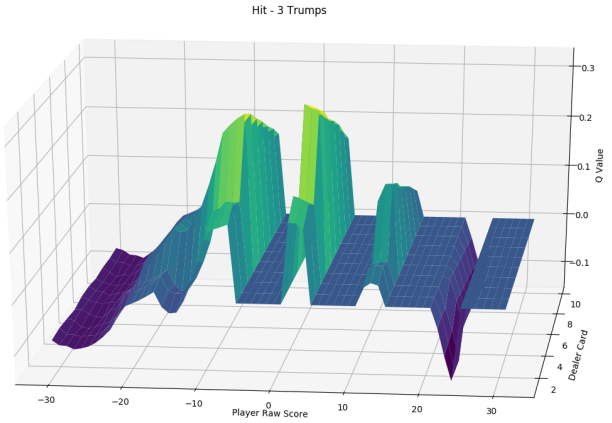

Figure 12: 1 Step TD Averaged over 1000 Runs - Hit - 3 Trumps

3.2 Monte Carlo Methods

Both the monte carlo methods (first visit and every visit) were run for 1,000,000 episodes each. When seeing the low probability states (Fig. 9 and Fig. 10), we can see that MC methods are able to learn the values of the low probability states fast. Also we observe that MC Every visit shows a lower variance then MC First Visit.

3.3 K Step TD Methods

All TD algorithms were run for 500,000 episodes.

While comparing 1 step TD (Fig. 12) with 1000 step td (Fig. 1) we can see that as the number of lookahead steps is increased the algorithm is able to learn the value of low probability states faster as the propagation of updates takes place faster. By observing the variation of number of steps of lookahead, it was found that after increasing lookahead beyond 10 no improvement is seen. This is justified as maximum episodes are less than 10 in length.

When comparing averaging over 100 and 1000 runs (Fig. 11 and Fig. 12) we can see that a higher number of averaging gives a much more cleaner and lower variance result.

3.4 MC methods vs. TD methods

- It is clearly evident that even after running MC methods for a larger number of episodes results in very high variance value functions whereas averaging TD methods results gives a very low variance result.
- It can also be seen that MC methods learn correct values of low probability states much faster than TD methods.

4 Policy Control

4.1 Algorithms

The algorithms implemented and tested are as follows :-

- k-step Lookahed SARSA with k = 1 , 10,100,1000 and epsilon(0.1) greedy policy
- k-step Lookahead SARSA with decaying epsilon(0.1) greedy policy and k=1,10,100,1000
- Q learning with 1 step greedy lookahead policy for update and epsilon(0.1) greedy policy for exploration.
- Forward View TD(0.5) with decaying epsilon(0.1) greedy policy.

4.2 Performance

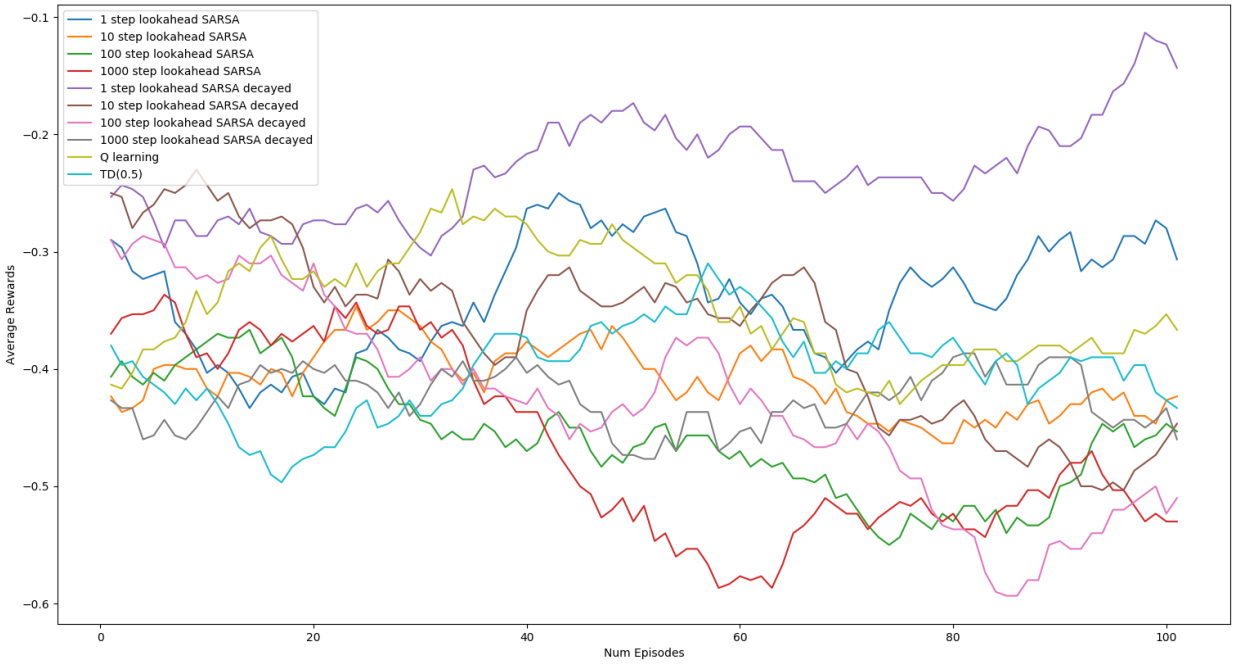

Figure 13: Performance while training over 100 episodes (Averaged over 10 runs)

As can be seen in Fig. 13, the best performing algorithms are 1 step lookahead SARSA decayed and TD(0.5). These results are fairly intuitive as TD(0.5) learns Q value function quite fastt and 1 step lookahead SARSA uses its learnt results instantly and also reduces exploration while increasing exploitation very early on in the test.

For a more clear view of the results the number of training episodes were increased to 100,000.

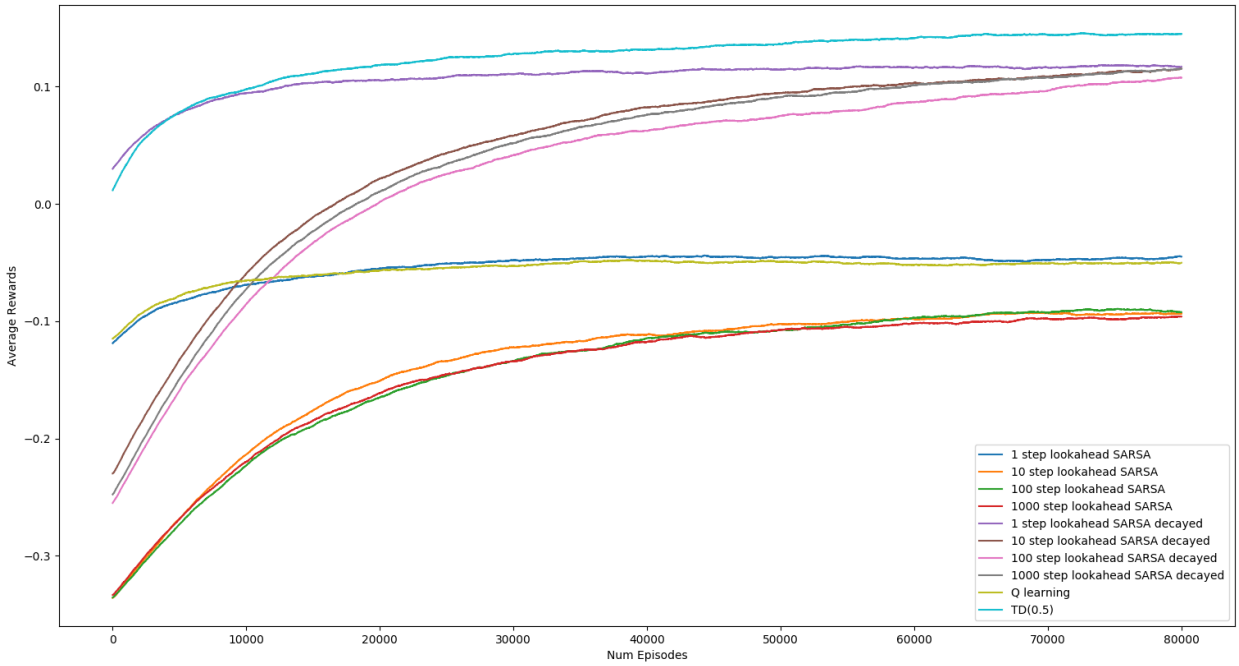

Figure 14: Performance while training over 100,000 episodes (Averaged over 10 runs)

In Fig. 15 we can see :-

- TD(0.5) is the best performing algorithm while the result of 1 step decayed SARSA reaches a constant as it stops learning.
- SARSA without decay results in constant reward after a certain number of iterations.
- 1 step lookahead SARSA is identical to Q learning under the given parameters and policy.
- SARSA with decay algorithms keep improving with small exploration term and increasing exploitative nature and further improvement in Q values.

4.3 Learning Rate vs. Performance

Results are reported after training algorithms for 100,000 episodes and testing over 10 runs.

- We can generally see a decrease in performance as learning rate is increased followed by an increase again as learning rate approaches 0.5.
- A continuous decrease is seen in the cases of k step lookahead SARSA without decay as a higher learning rate prevents the algorithms from converging.
- Algorithms with decay are seen to perform better with higher learning rates.

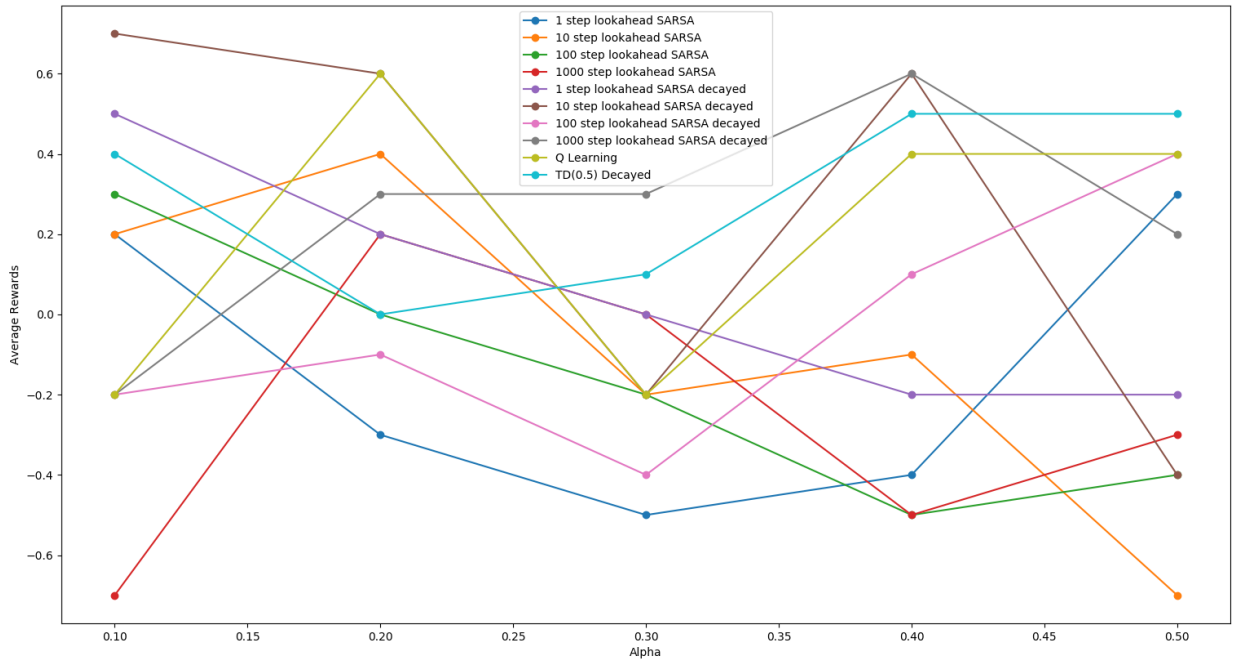

Figure 15: Learning Rate vs. Average Reward (10 Test Runs)

4.4 TD(0.5) Q Value Functions

TD(0.5) averaged over 100 runs

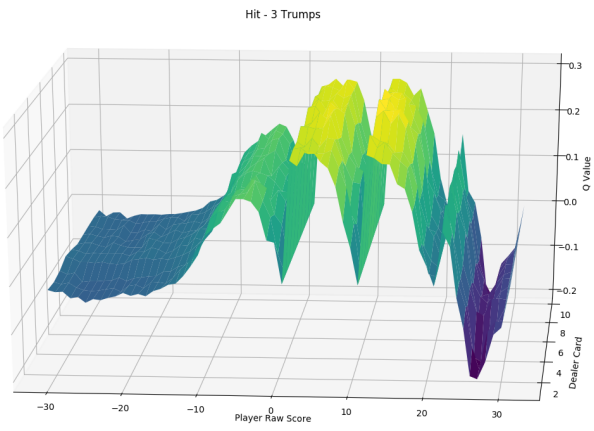

Figure 16: Hit - 3 Trumps

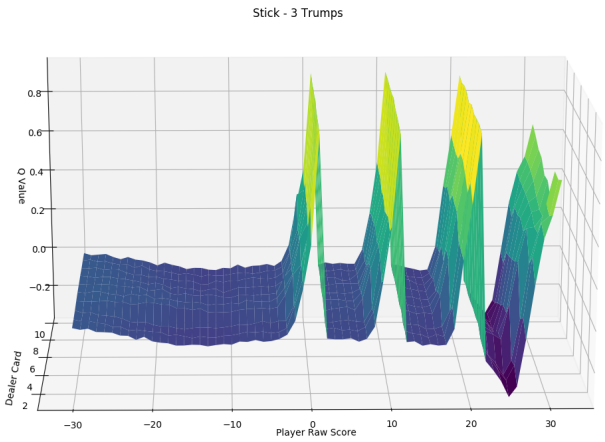

Figure 17: Stick - 3 Trumps

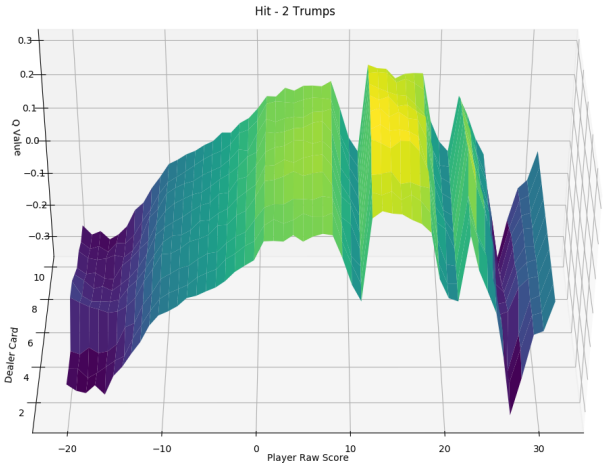

Figure 18: Hit - 2 Trumps

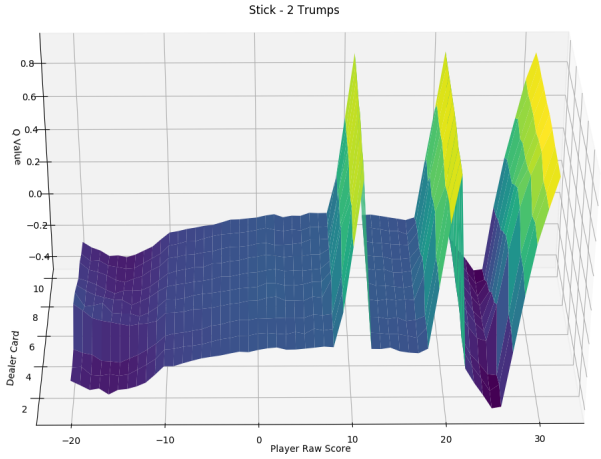

Figure 19: Stick - 2 Trumps

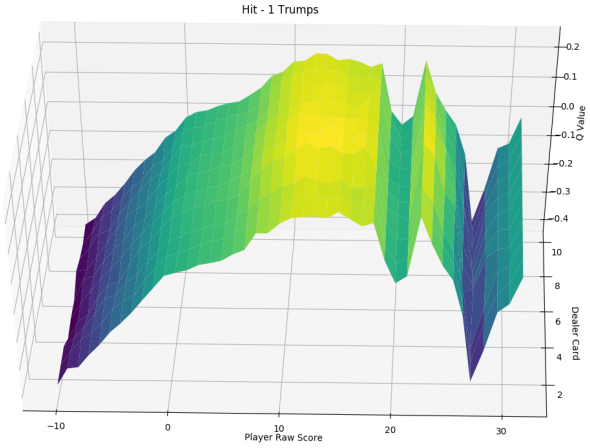

Figure 20: Hit - 1 Trump

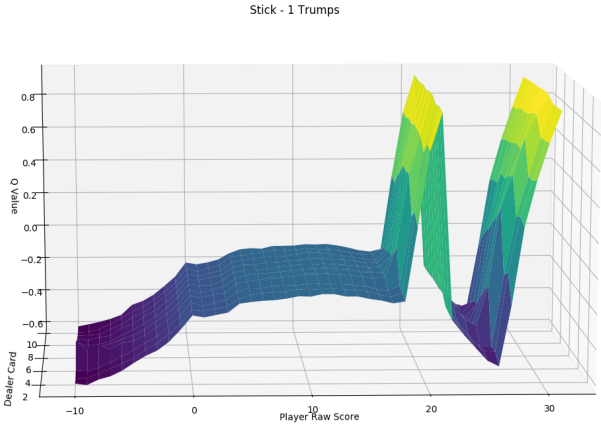

Figure 21: Stick - 1 Trump

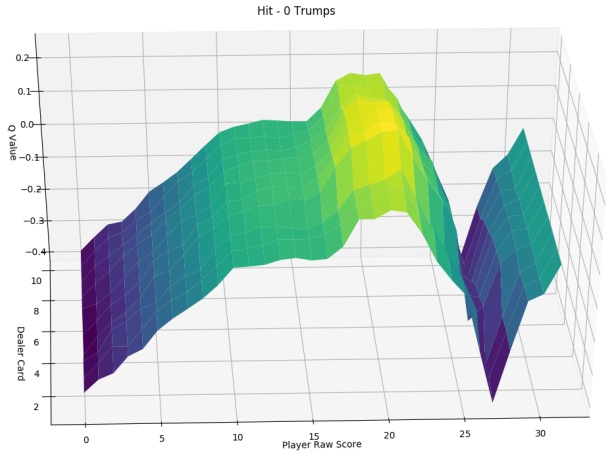

Figure 22: Hit - 0 Trump

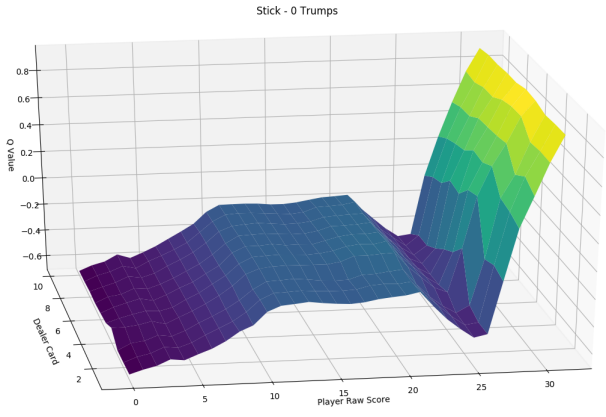

Figure 23: Stick - 0 Trump

We can see that the Q value functions are similar to the ones obtained with a fixed policy with the following changes :-

- There are no flat regions now as all states are explored with non-zero probability.
- The initial policy was not very bad as we can see that the zero regions in the earlier Q graphs have now been replaced by negative regions. (Evident in stick graphs)
- Stick policy no more depends on just the total score of the player but also the number of trumps and raw score of the player.