COL 870 – Reinforcement Learning

Assignment 1 – Achieve 31

Shreshth Tuli

2016CS10680

State Space

My state representation is of the form:

$$S = (X, H, D) \cup Bust$$

Where S represents the set of possible states, X represents number of total usable special cards, H represents hard sum of player and D represents set of possible dealer cards at start. $Bust$ represents the state in which player busts in the game as described in the problem statement. Now:

1. As the player can have at max 3 special cards each of black cards of 1, 2, 3:

   $$X \equiv \{0, 1, 2, 3\}$$

2. We define hard sum as the sum of all cards of the player without considering any card as special. Hence, we add all black cards face values and subtract red cards face values to obtain the hard sum of the player. Now, for a no special cards the hard sum can vary from 0 to 31, otherwise the player would go bust if it exceeds 31 or goes below 0. For 1 special card, the hard sum can range from -10 to 31 as for {-10, ... -1} the special card can be used as higher value and for {20, ... 31} can not be used as higher value. Similarly, for 2 and 3 cards the hard sum can vary from -20 to 31 and -30 to 31. Hence, hard sum has overall range:

   $$H \equiv \{-30, -29, \ldots., 31\}$$

   Note that the states where hard sum is less than the value possible as per the number of special cards are considered as bust state

3. As the dealer can have cards 1 to 10 the dealer hand:

   $$D \equiv \{1, 2, \ldots, 10\}$$

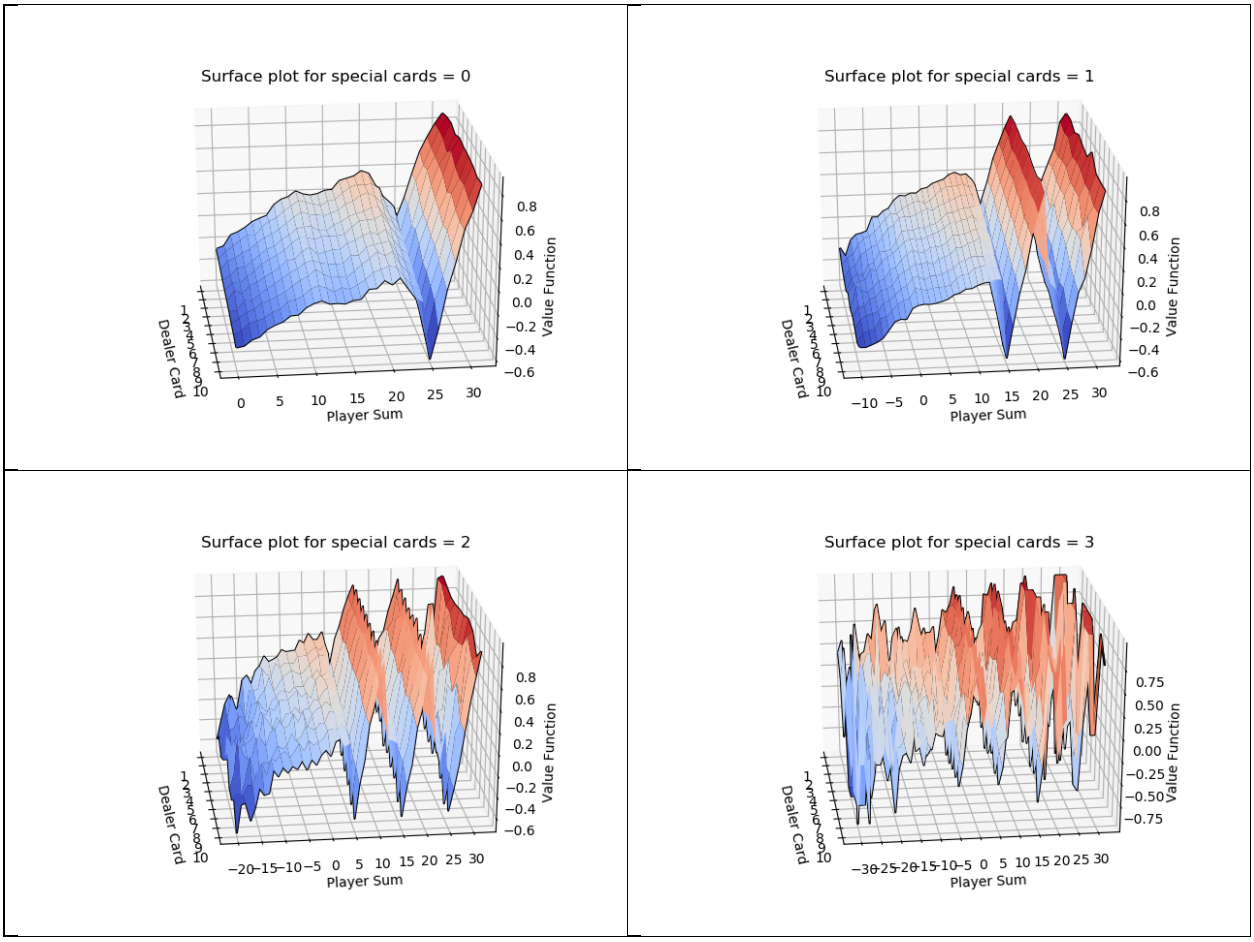
Among these states, the non-actionable states are those in which soft sum i.e. the maximum sum that can be formed by the cards (including the special ones) such that the sum is less than or equal to 31 is 31, or the player is in bust state. This is because when the soft sum is 31 it is obvious that the only reasonable option would be to stick and not hit. If the player sticks it can win or lead to a draw. If it sticks it will always go to bust state. The other case in which every possible combination of cards leads to sum of > 31 or < 0 is also non-actionable as player has lost (or draw match as per special case in the statement) in this state.

Policy Evaluation

For the given simple policy of playing 'hit' until the player can reach a sum of 25 or more then playing 'stick', we evaluate using Monte Carlo and TD.

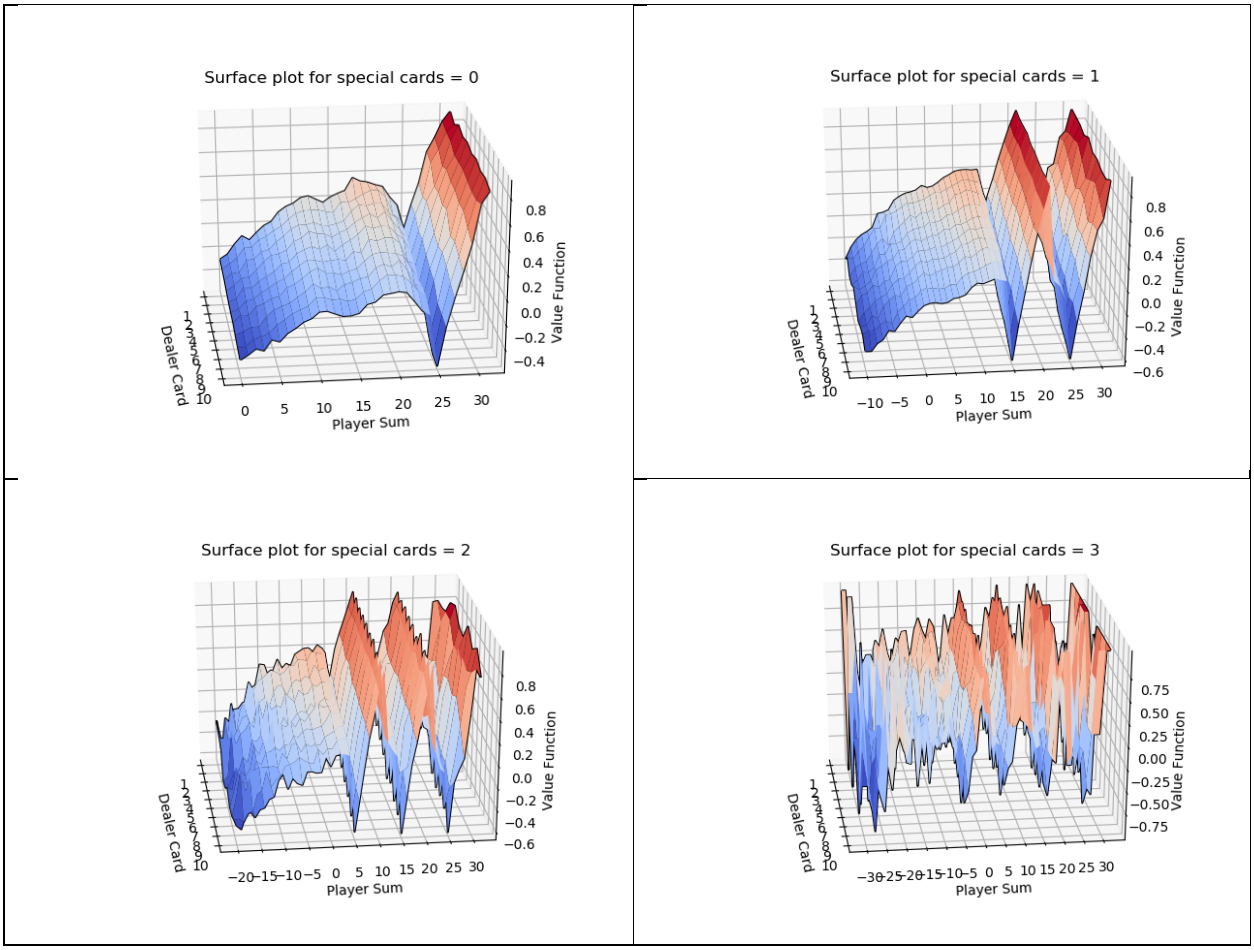
Policy evaluation results in terms of value function for:

- Update strategy: **Monte Carlo**
- Variant: **First visit update**
- Runs: **1**
- Episodes per run: **1 million**



Policy evaluation results in terms of value function for:

- Update strategy: **Monte Carlo**
- Variant: **Every visit update**
- Runs: **1**
- Episodes per run: **1 million**



We see that both graphs are similar in their trends. General trends and reasoning:

1. There is a drop at the value function at 25 soft sum for all special card range 0 to 3. This is because at this point for less than 25 cards, there is possibility of following policy and playing 'hit' to lead to drawing a black card with value >= 7 leading to bust. Moreover, the value function increases drastically after 25 because the policy now changes and the player 'sticks' leading to mostly winning the game as the soft sum is quite high (25 or more).
2. There are similar drop and peak at 15, 20 when there is 1 special card because based on whether the card is used with a higher value or not, the hard sum can be 15 or 25 to lead to the soft sum as 25. Similarly based on special card is used with higher value or not, the peak lies on 31 and 21.

3. As special cards increase, this patter or drop and sudden rise in value function repeats as many number of times as there are special cards with an offset of 10 based on how many special cards are being used with higher value.
4. There is also an increase in the value function as the dealer card value reduces as the chances of winning are higher if the existing dealer card is of low value. Though this increase is gradual still it is noticeable in all graphs.

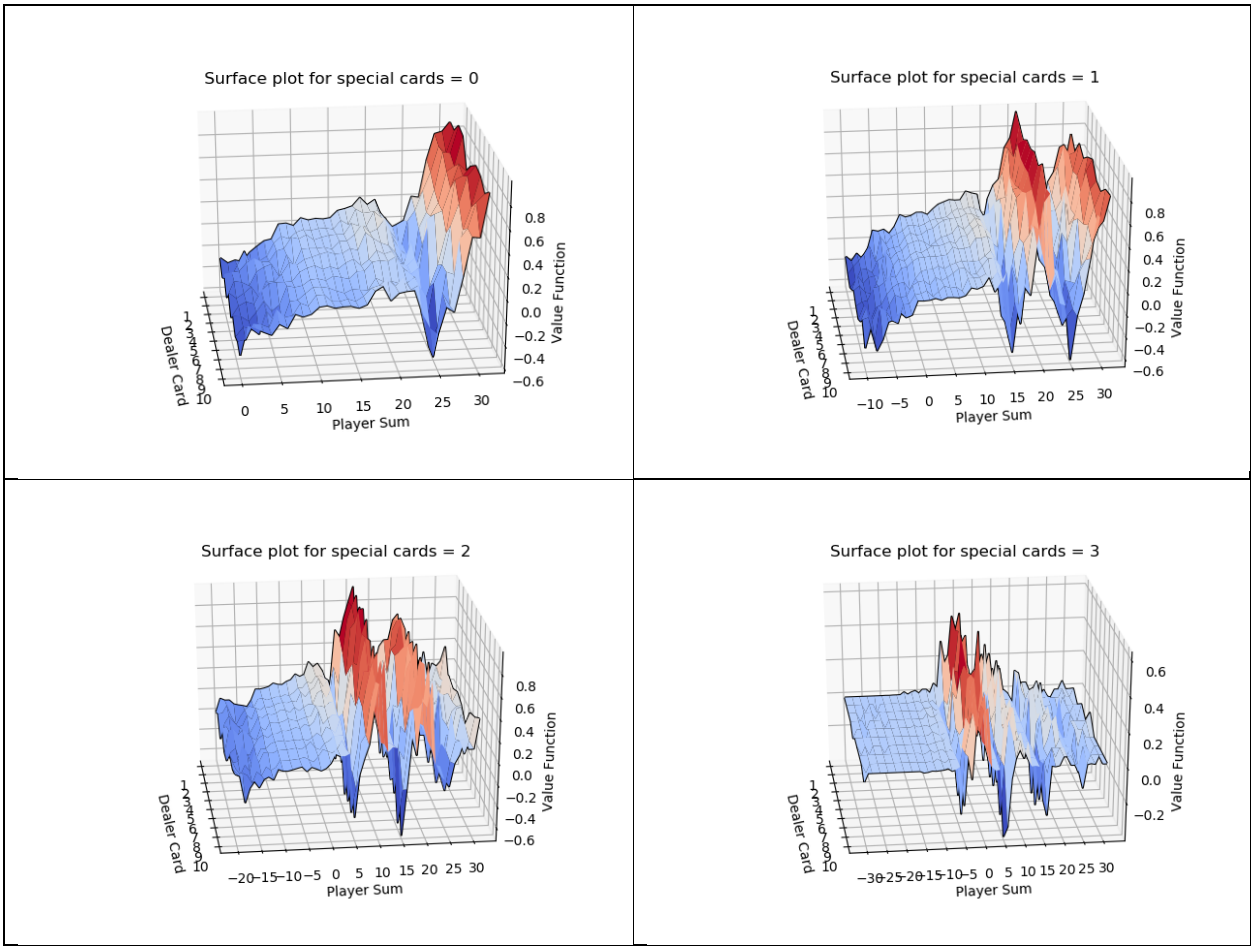
Small differences in MC first visit and every visit:

1. The graph of the every-visit case has slightly higher variance than the first visit one probably because of higher number of samples with different number of occurrences of each sample.

Now we show the graphs obtained for TD methods with varying k. Note that the gamma for all further graphs with TD (prediction/control) is assumed to be 0.7.

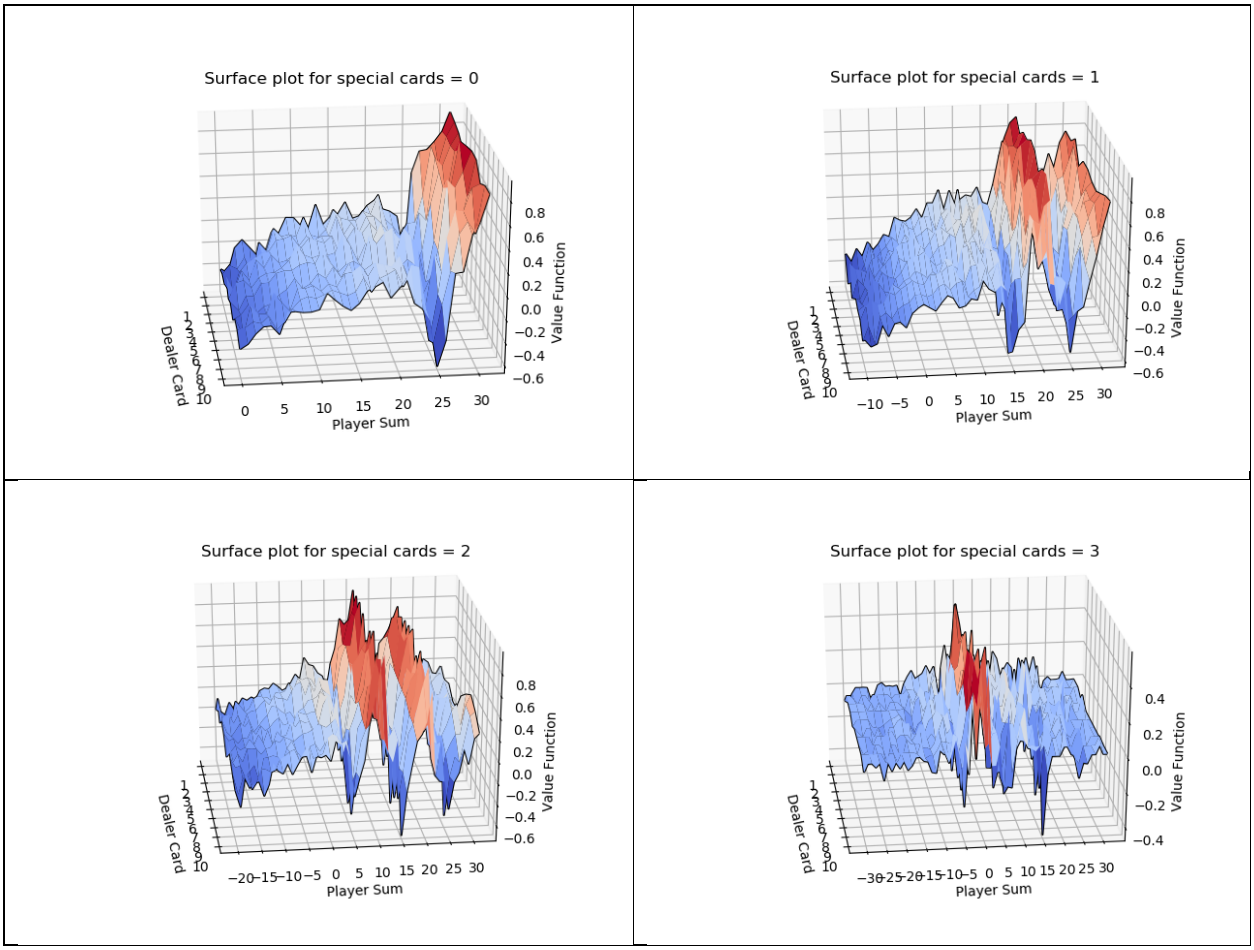
Policy evaluation results in terms of value function for:

- Update strategy: **Temporal Difference**
- Initialized value for each state: **0**
- k: **1**
- Runs: **10**
- Episodes per run: **100,000**



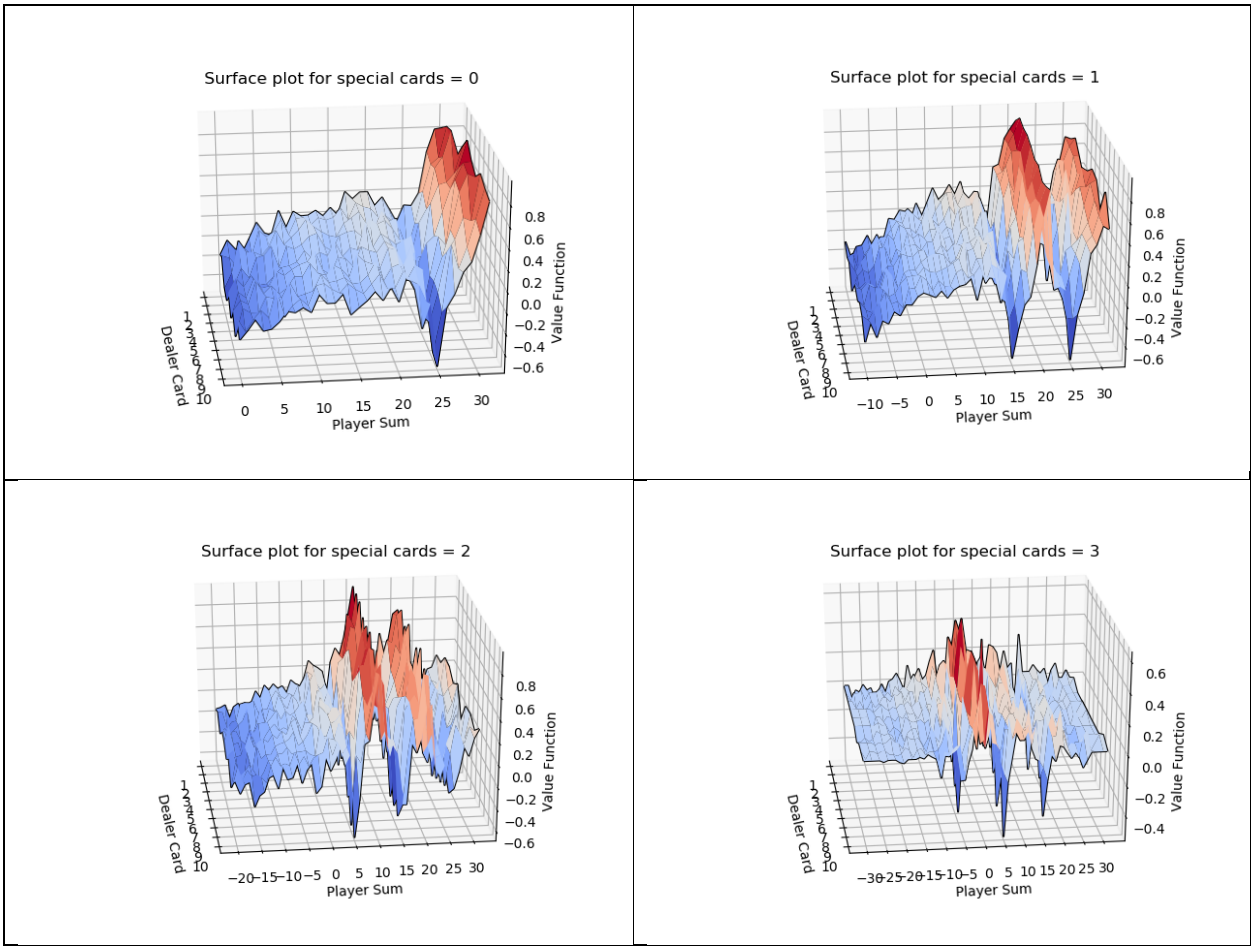
Policy evaluation results in terms of value function for:

- Update strategy: **Temporal Difference**
- Initialized value for each state: **0**
- k: **3**
- Runs: **10**
- Episodes per run: **100,000**



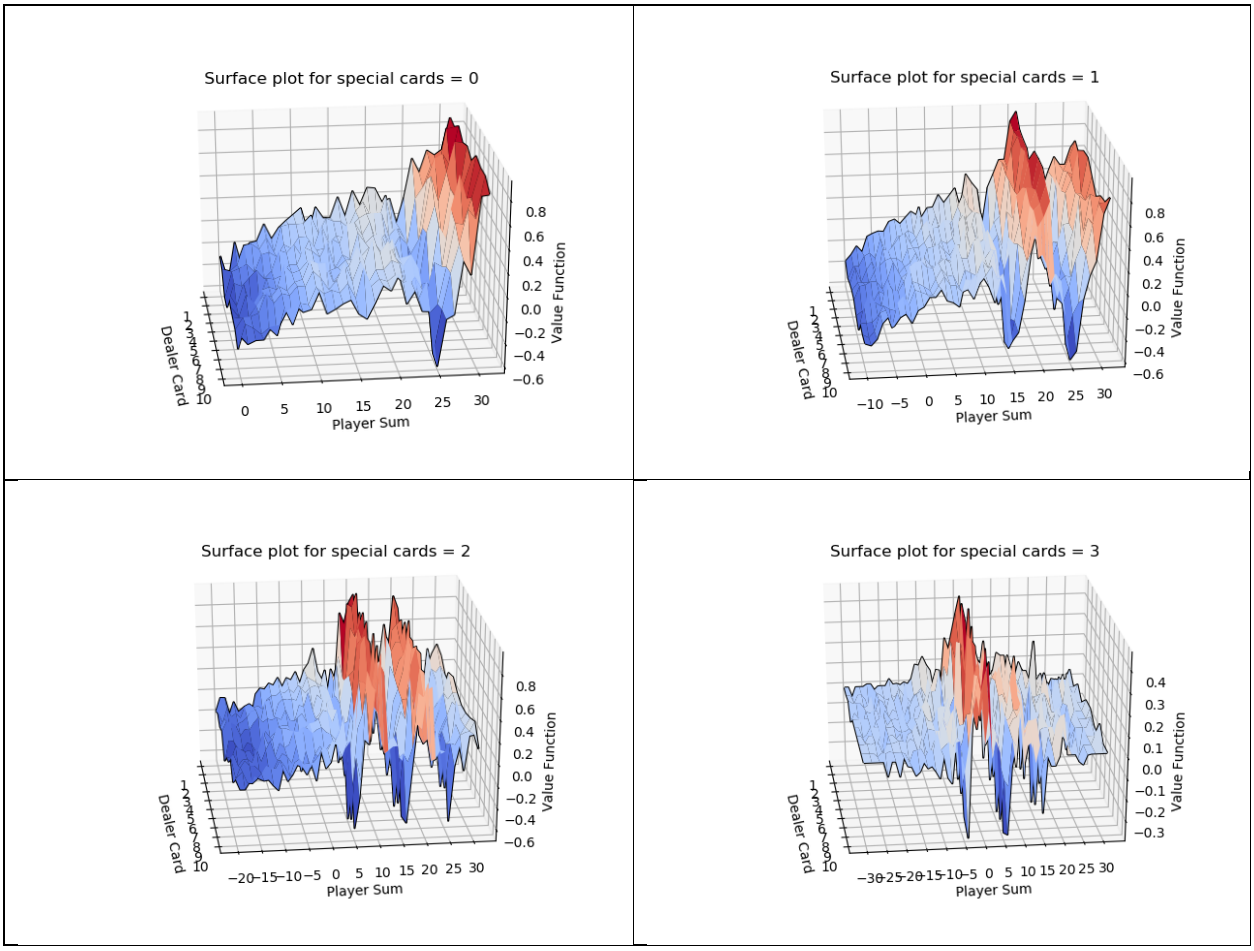
Policy evaluation results in terms of value function for:

- Update strategy: **Temporal Difference**
- Initialized value for each state: **0**
- k: **5**
- Runs: **10**
- Episodes per run: **100,000**



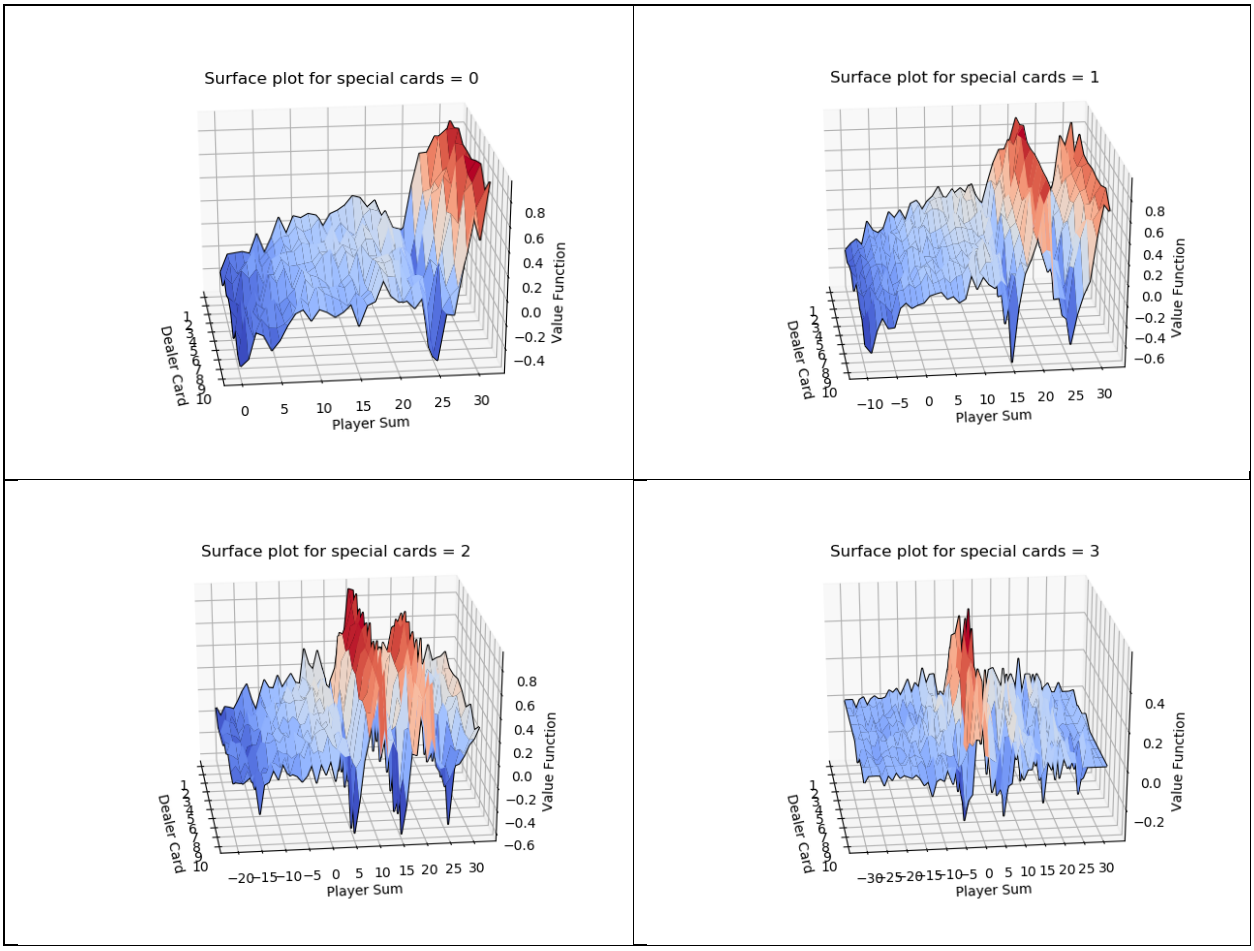
Policy evaluation results in terms of value function for:

- Update strategy: **Temporal Difference**
- Initialized value for each state: **0**
- k: **10**
- Runs: **10**
- Episodes per run: **100,000**



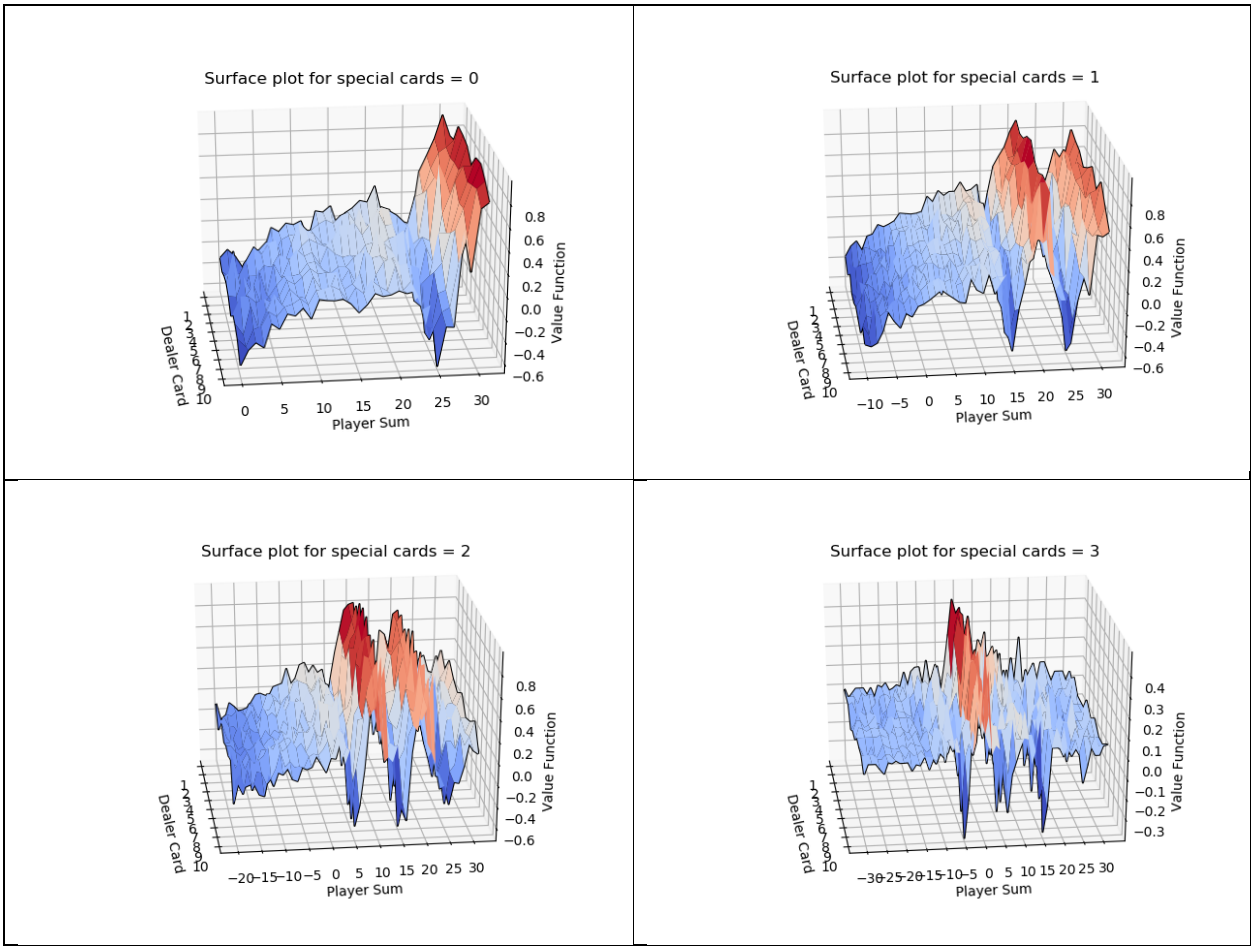
Policy evaluation results in terms of value function for:

- Update strategy: **Temporal Difference**
- Initialized value for each state: **0**
- k: **100**
- Runs: **10**
- Episodes per run: **100,000**



Policy evaluation results in terms of value function for:

- Update strategy: **Temporal Difference**
- Initialized value for each state: **0**
- k: **1000**
- Runs: **100**
- Episodes per run: **100,000**

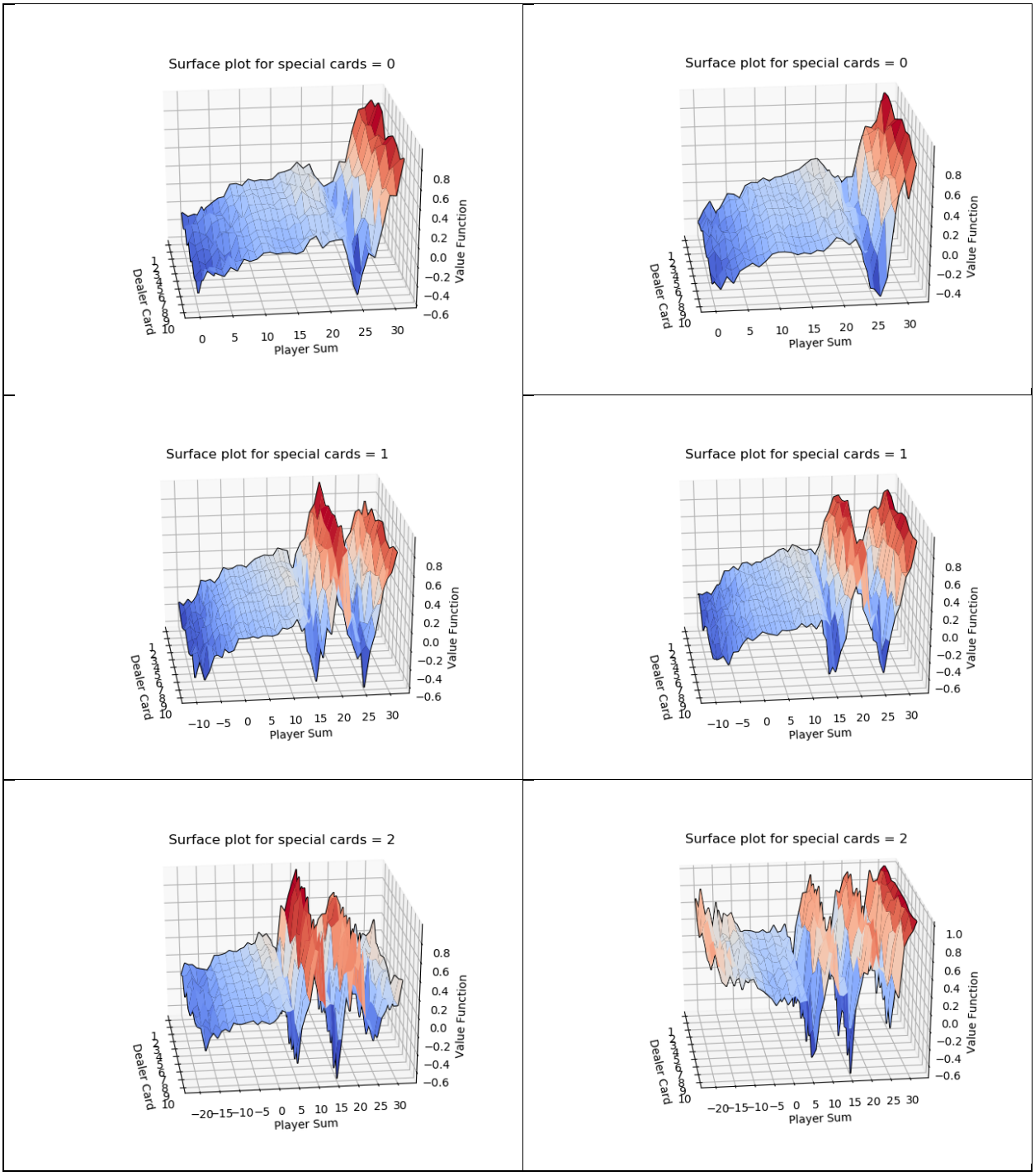


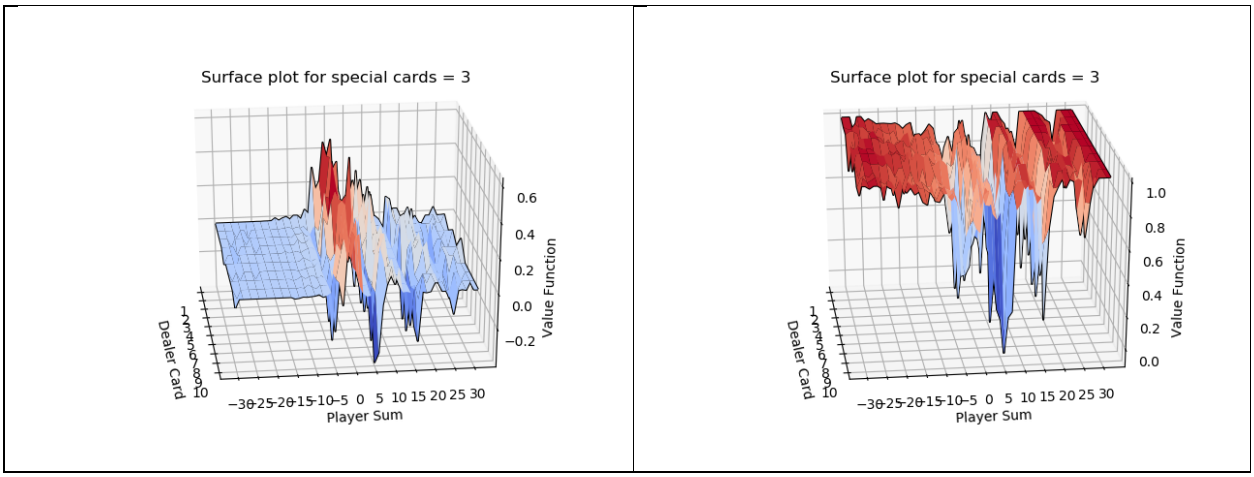
Most of these graphs are similar to the MC case with some differences:

1. The value functions are biased by the initialization. The graphs shown before have initialized value as 0 for all states. This bias can be differentiated by the figures below which show TD value function for different special cards. Both have k = 1 and ran for 100 runs, each for 100,000 episodes. On the left the initialized value for each state was 0, and on the right it was 1. As shown in the graphs, for rare cases especially with 2 or 3 special cards, there is high bias towards the initialized value as the number of samples is quite low.

TD with init value 0

TD with init value 1

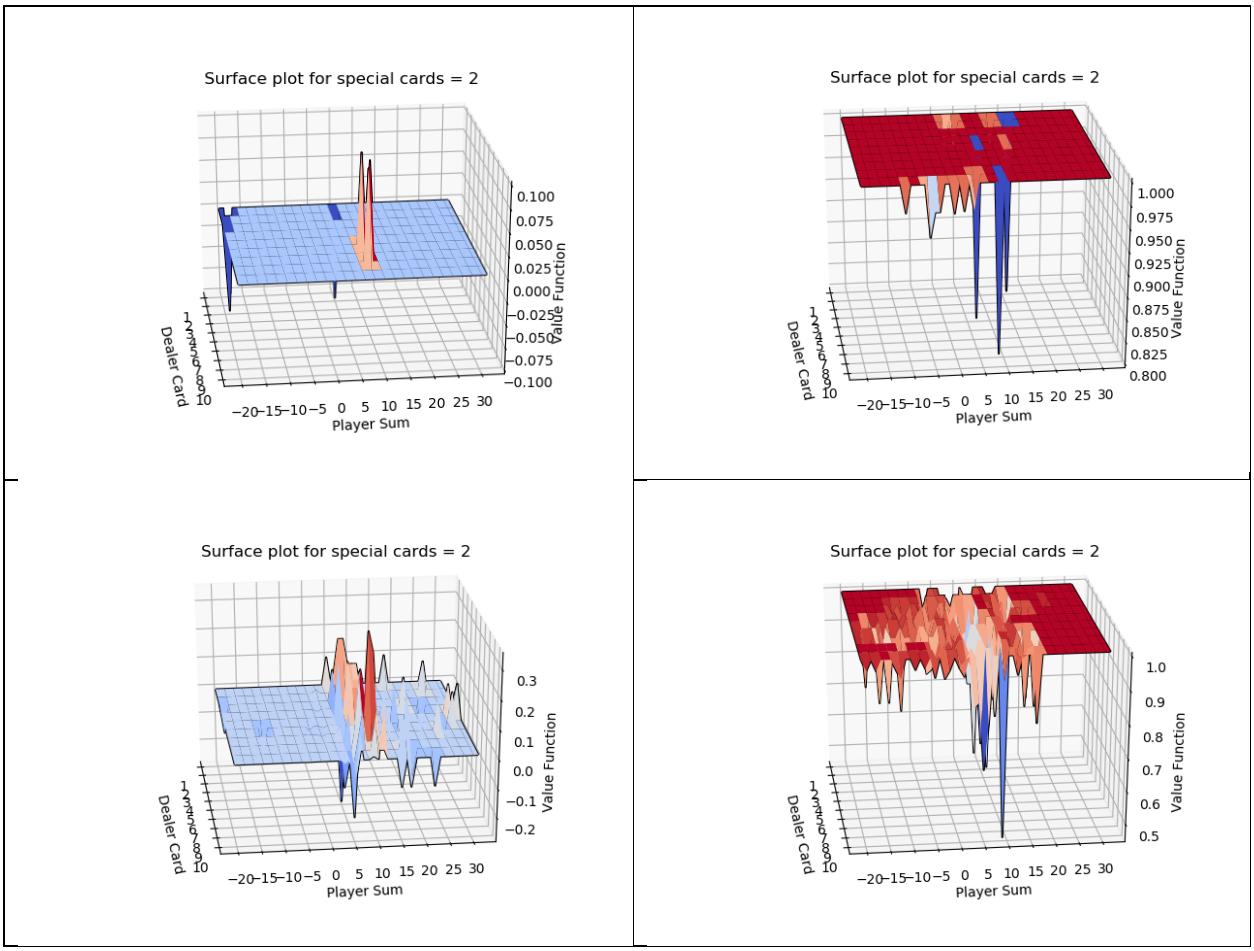


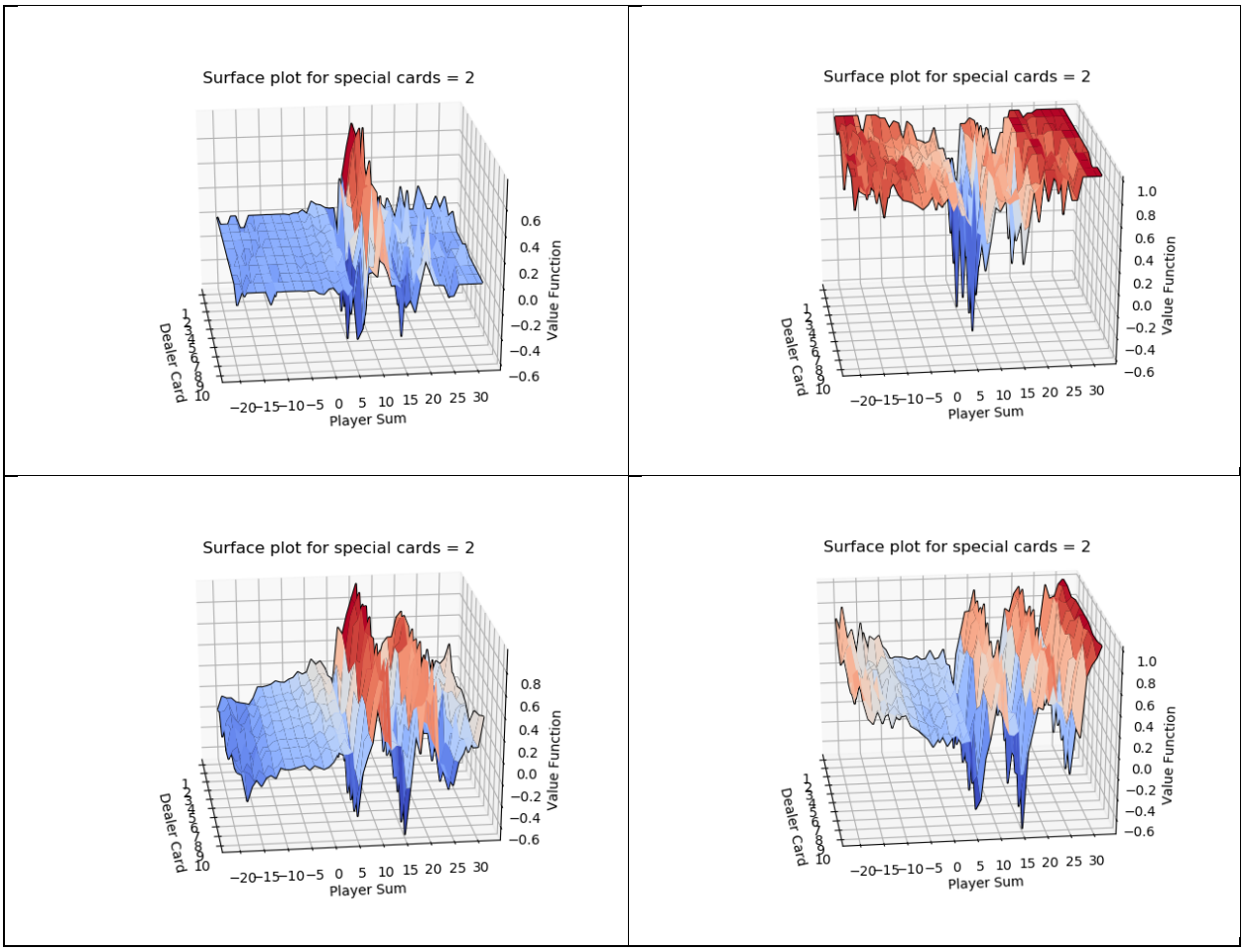


A clearer diagram which shows biasing as the TD algorithm progresses is shown below. We show the value function after 100, 1000, 10,000 and 100,000 episodes for 2 special cards and k = 1. On the left the value function is initialized to 0 and on the right to 1.

TD with init value 0

TD with init value 1



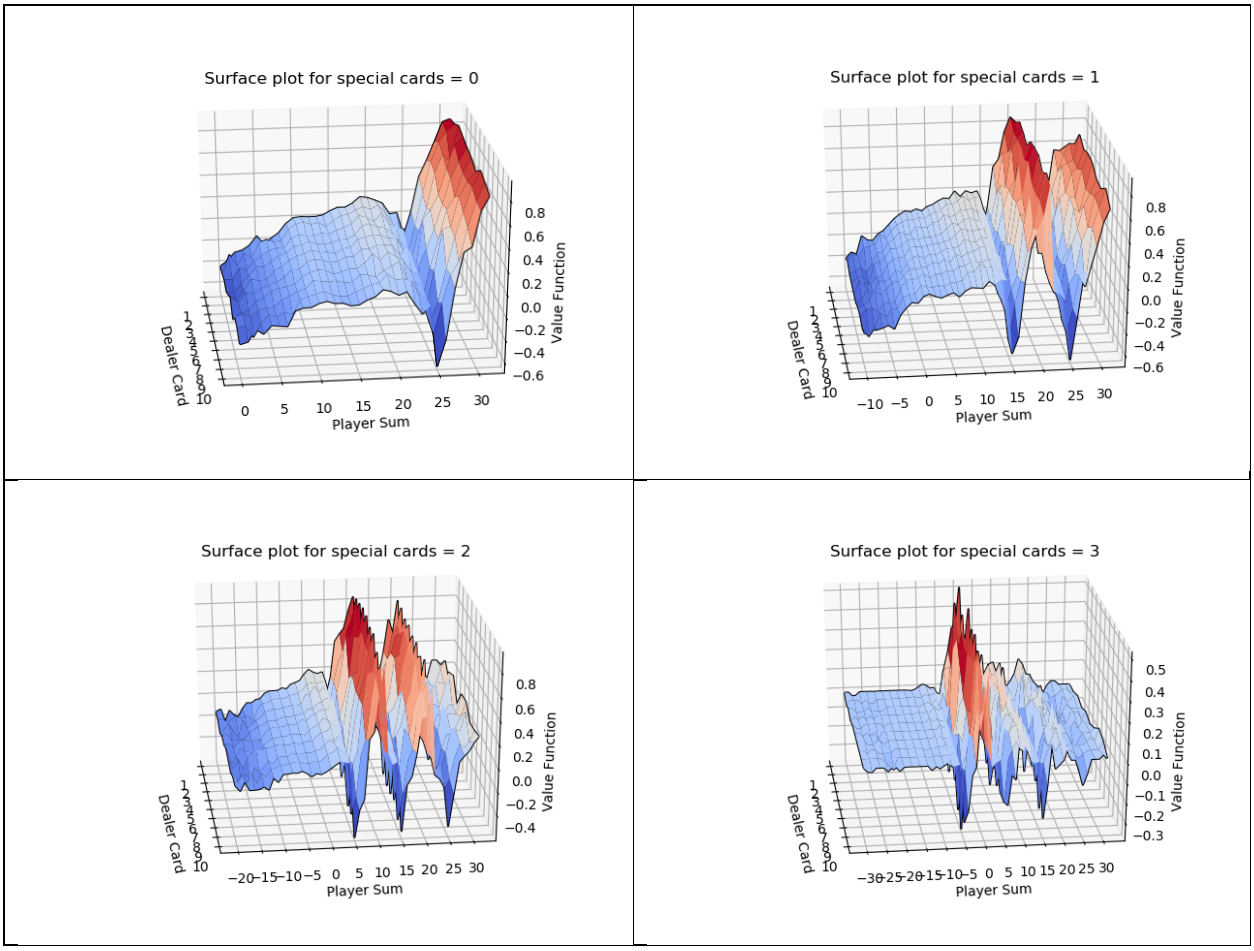


2. The graphs also show high variance i.e. peaks and non smooth surface plots for the cases where special cards are 2 or 3. This too is because of low number of samples in the episodes.

Some representative graphs for 1000 runs can be seen below. Rest can be seen in the 'figs' folder that came along with this report.

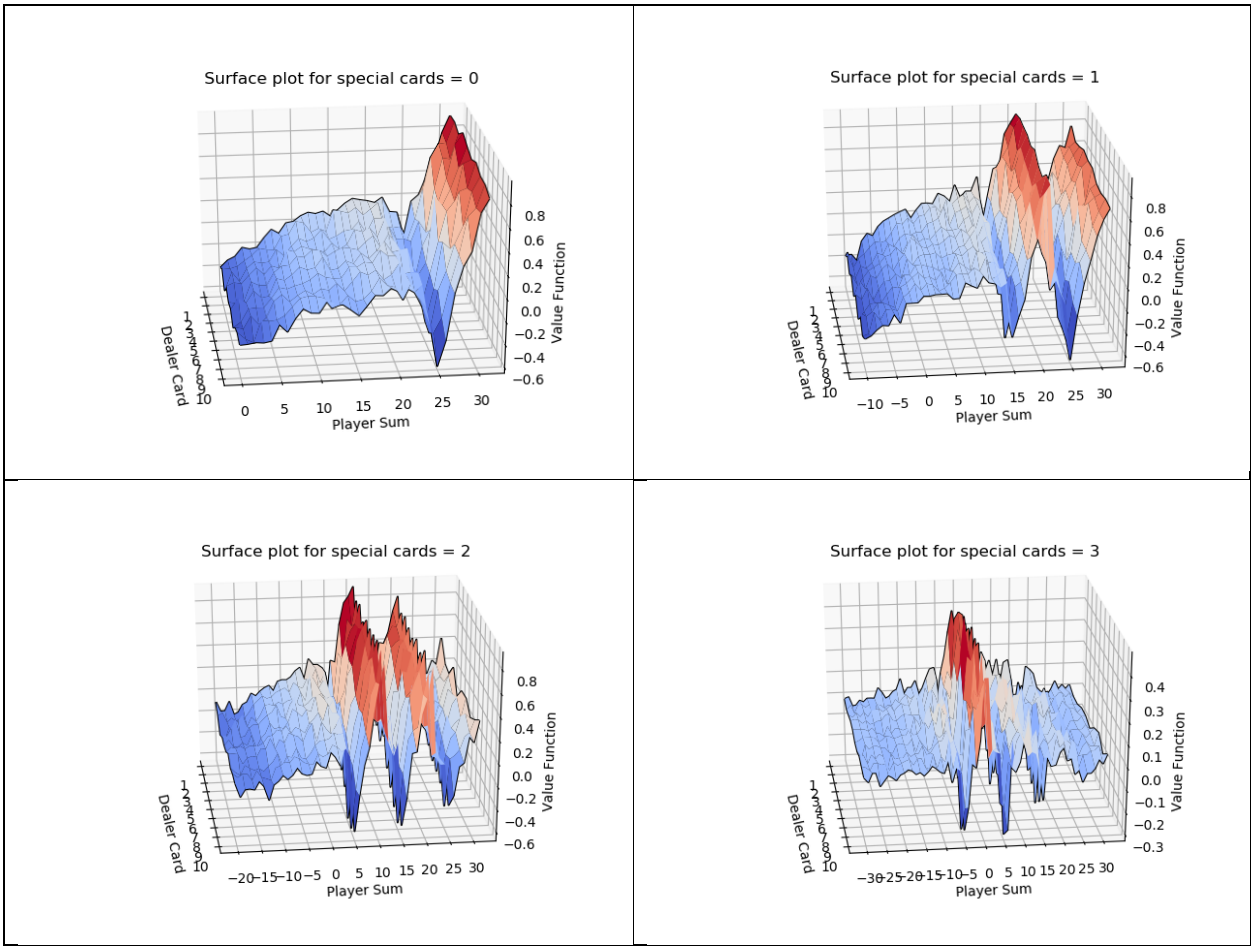
Policy evaluation results in terms of value function for:

- Update strategy: **Temporal Difference**
- Initialized value for each state: **0**
- k: **1**
- Runs: **1000**
- Episodes per run: **100,000**



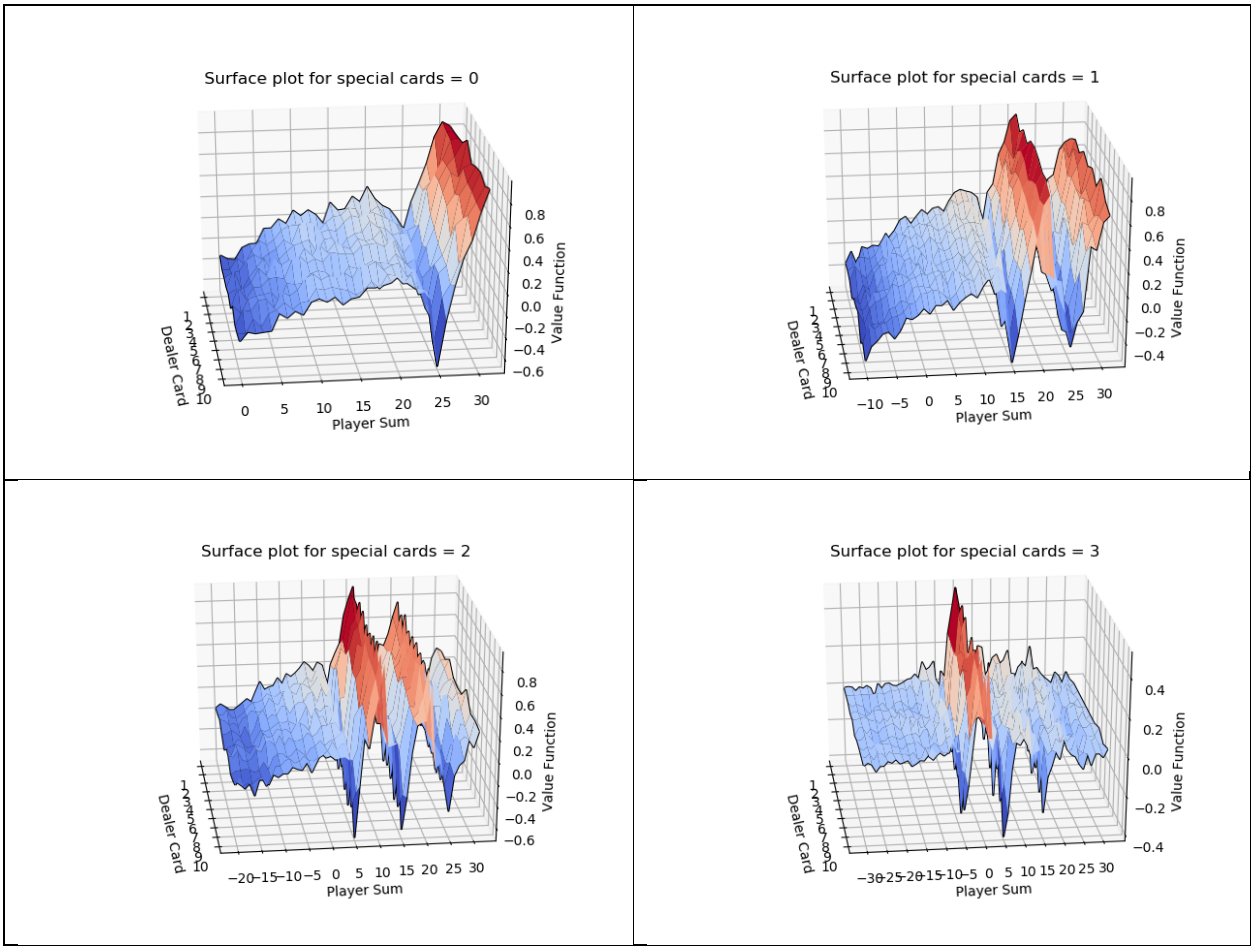
Policy evaluation results in terms of value function for:

- Update strategy: **Temporal Difference**
- Initialized value for each state: **0**
- k: **10**
- Runs: **1000**
- Episodes per run: **100,000**



Policy evaluation results in terms of value function for:

- Update strategy: **Temporal Difference**
- Initialized value for each state: **0**
- k: **1000**
- Runs: **1000**
- Episodes per run: **100,000**



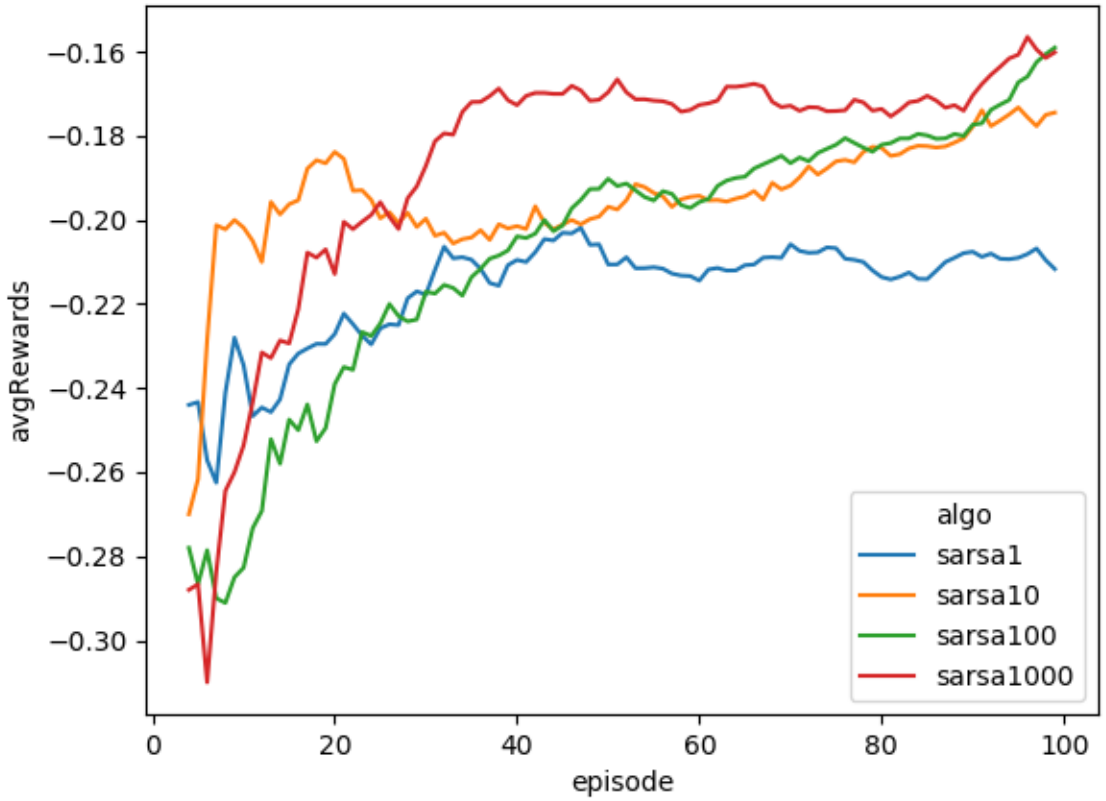
We see that the graphs are much smoother in this case as the number of episodes for averaging of value function is much higher.

Policy Control

We now show the performance of the SARSA policy control algorithm with k-step lookahead.

- Epsilon: **0.1**
- Runs: **10**
- Learning rate (alpha): 0**.1**
- Episodes per run: **100**

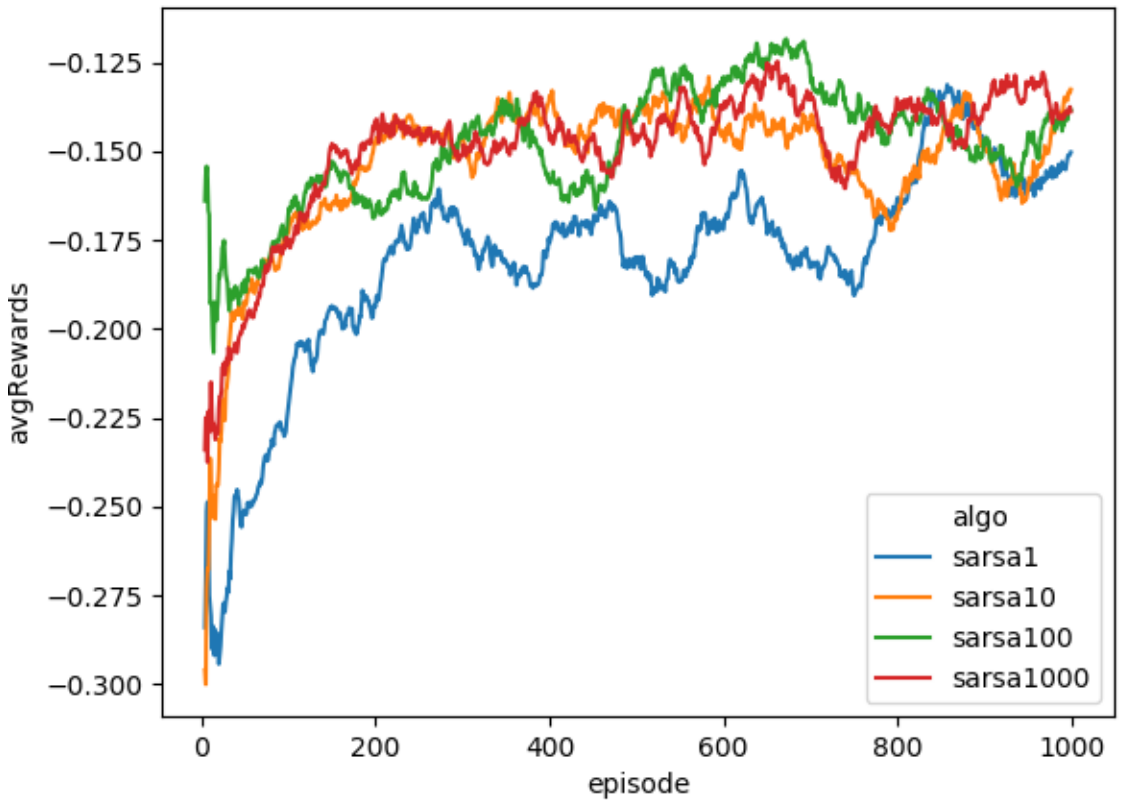
SARSA learning without epsilon decay (100 episodes):



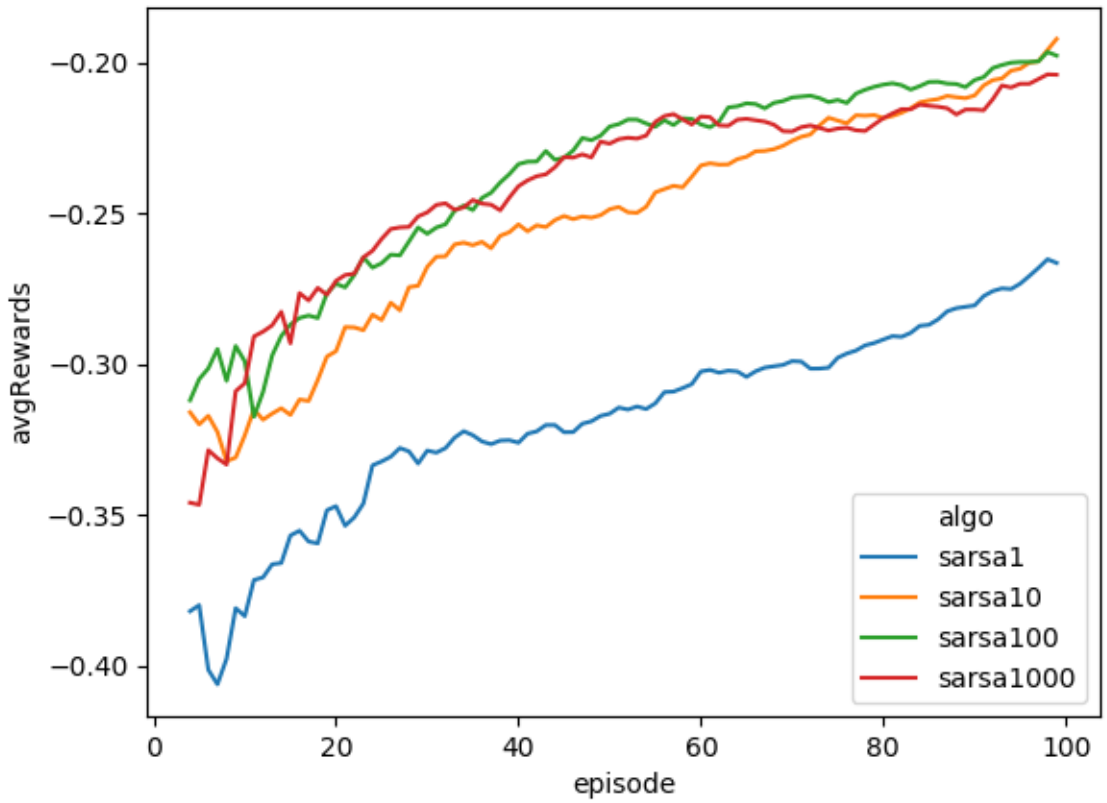
We see that the 1 step SARSA performs worst as it is not able to propagate values in these many episodes (100 in our case). In comparison 10 step SARSA performs much better and 100 or 1000 step SARSA perform best. This is because in 100 episodes they are able to propagate value functions much faster.

When we run this for 1000 episodes, we see that 1 step SARSA also gradually becomes as good as higher step SARSA as now it is able to propagate all values to other states.

SARSA learning without epsilon decay (1000 episodes):

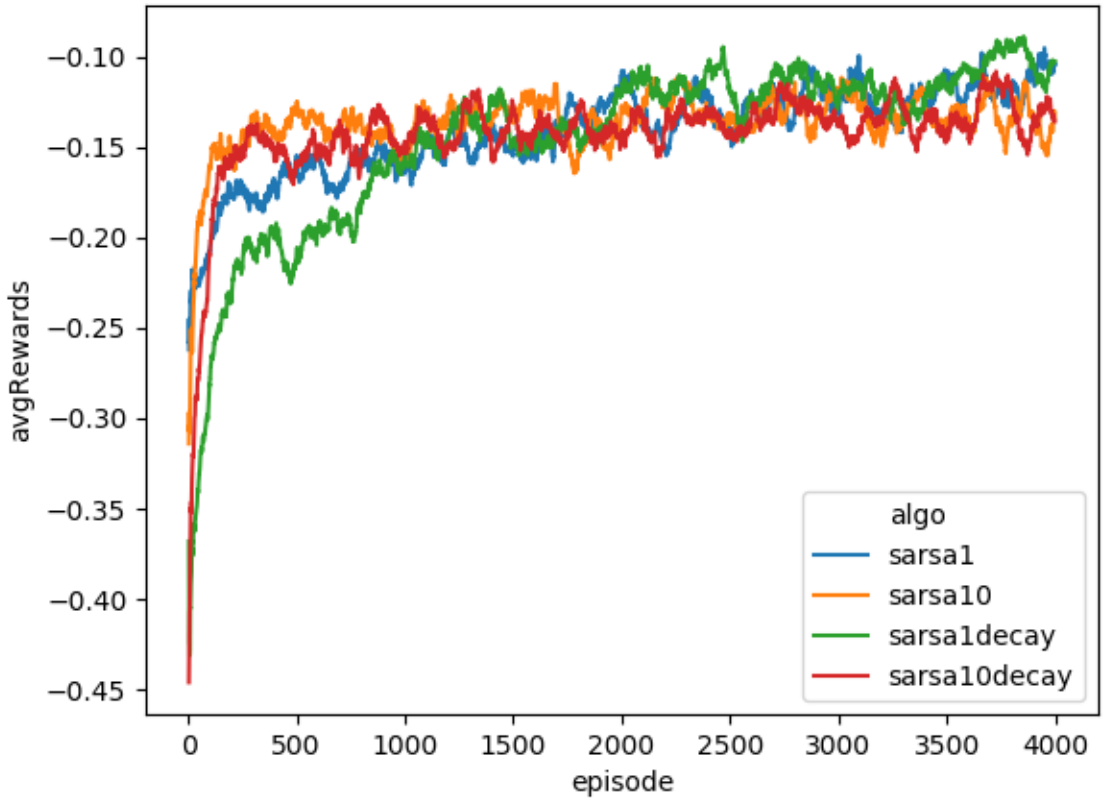


SARSA learning with learning rate decay:

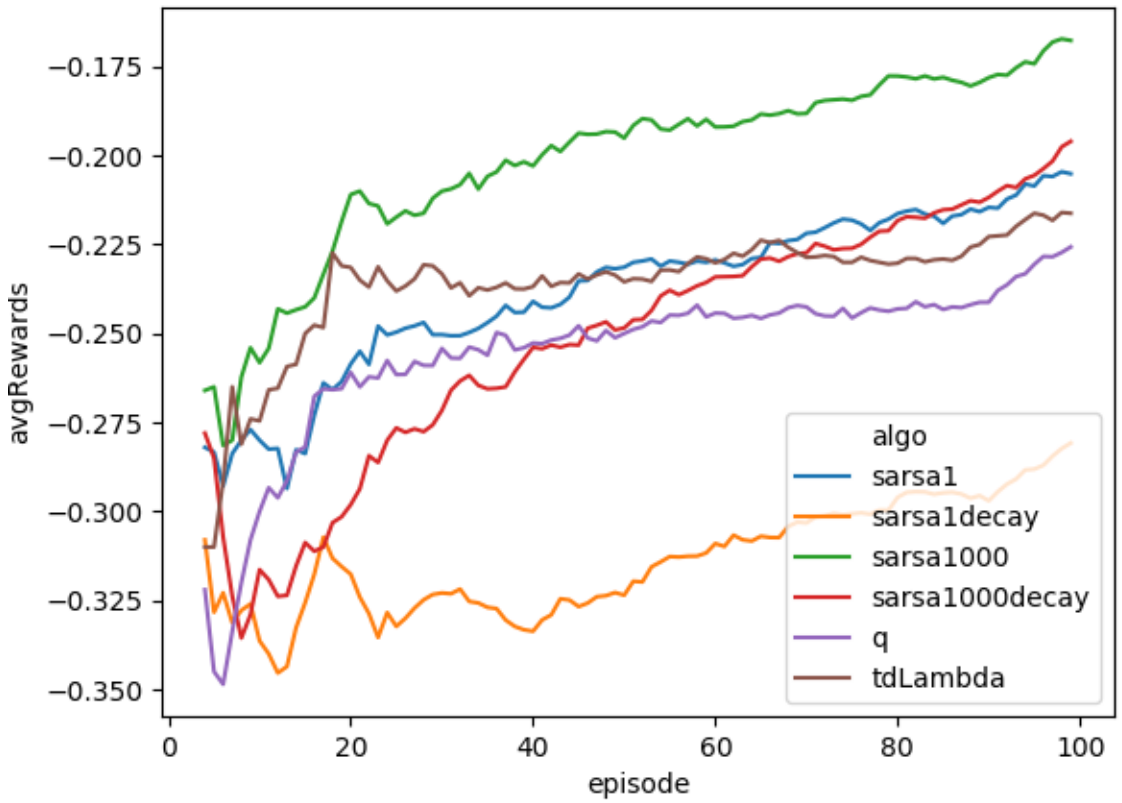


We see that the performance deteriorates but the learning slope is higher than the non-decay case. The performance is less because in 100 episodes the learning rate is much lower hence change in policy is also slow.

We see below that SARSA with decay surpasses the one with no decay in 4000 episodes:

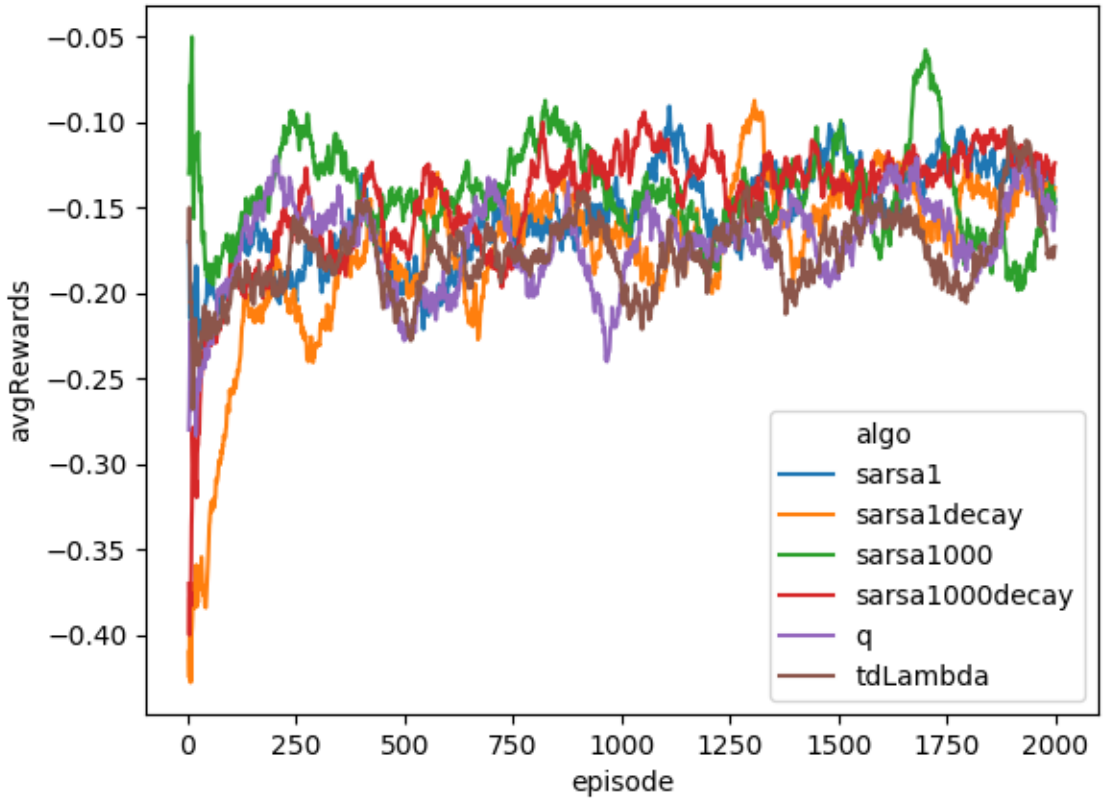


Comparison of all control algorithms for first 100 episodes:



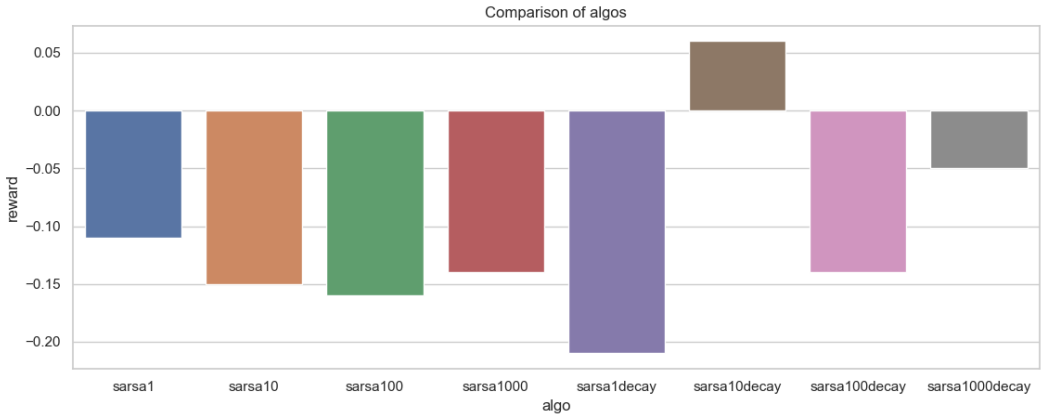
We see that the best algorithms is SARSA 1000 step but slope of SARSA 1000 step with decay is higher. This is because it is almost unbiased for all episodes with <= 1000 steps as it does not use bootstrapping for such episodes. Thus, initialization has little effect for high number of lookahead steps. Also, Q learning performance is poorer than SARSA because it does not play defensive as SARSA does because of its off-policy updates. In 2000 episodes we can see that SARSA 1000 step with decay performs best as it converges to epsilon soft optimal with much smaller epsilon compared to SARSA 1000 without decay.

Comparison of all control algorithms for first 2000 episodes:



We now train all algorithms for 100,000 episodes and compare performance (average reward on 10 episodes).

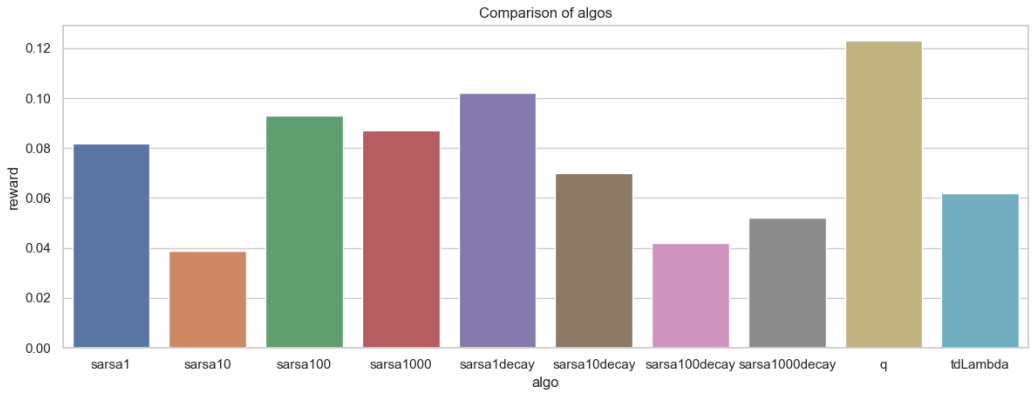
SARSA performance comparison after 100,000 episodes;



There is lot of variance in results for 100,000 episodes and these results do not show converged values.

Comparison of all algorithms after 100,000 episodes:

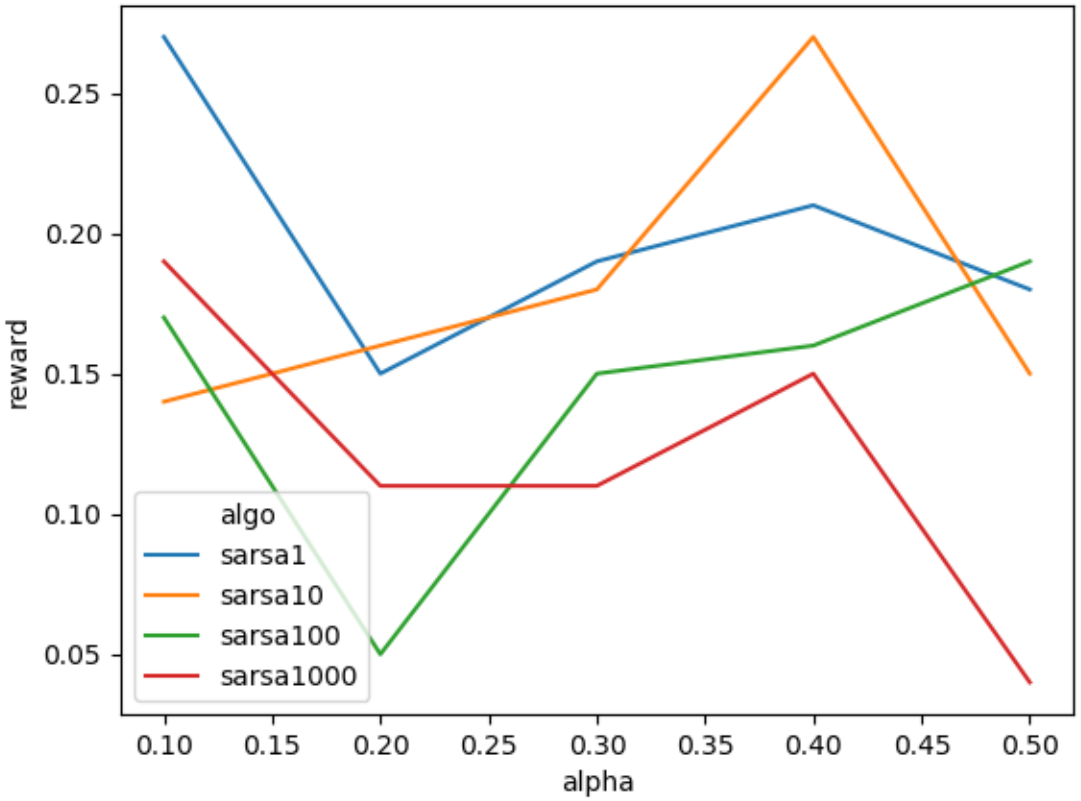
Comparison of all algorithms after 500,000 episodes:



We see that the policies have improved to perform with positive rewards only after 500,000 episodes. Here Q learning performs best as it is off-policy update and converges to optimal policy whereas SARSA only converges to epsilon-soft to optimal policy. For 1 and 10 step lookahead, SARSA decay is better than non-decay case. However, for 100 and 1000 step lookahead, decay reduces the performance.

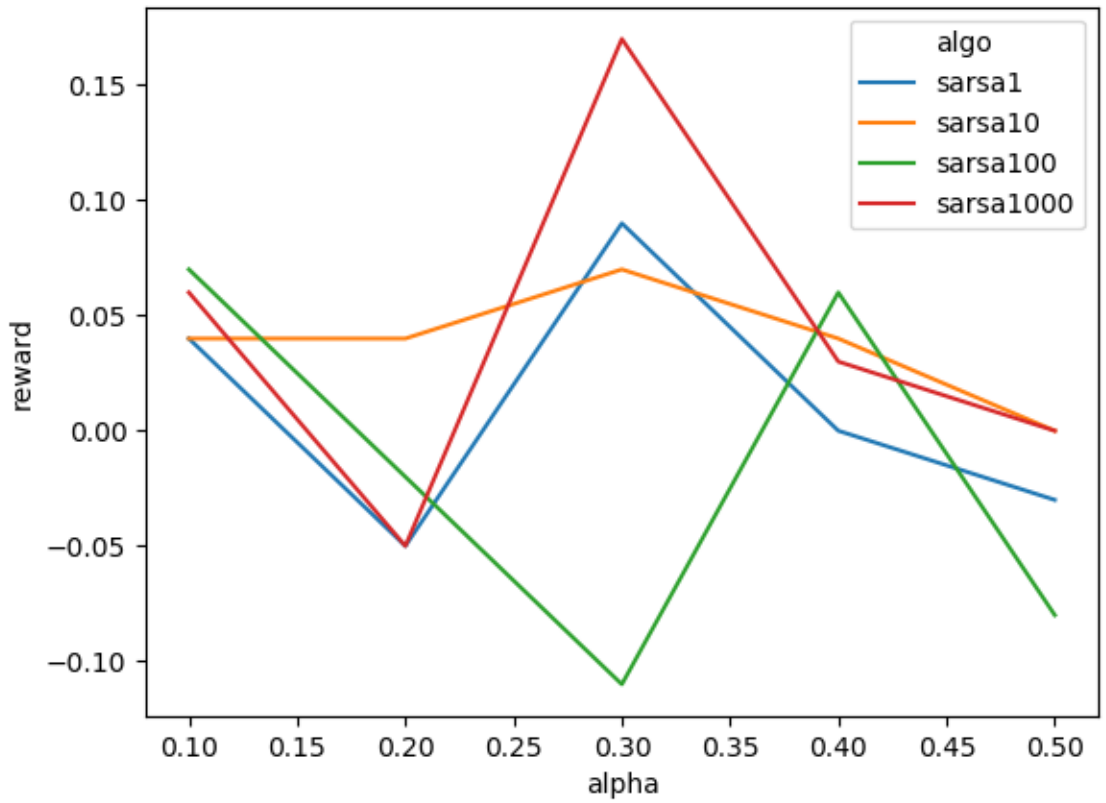
Now if we vary alpha values:

Comparison of performance for SARSA without decay with varying alpha:



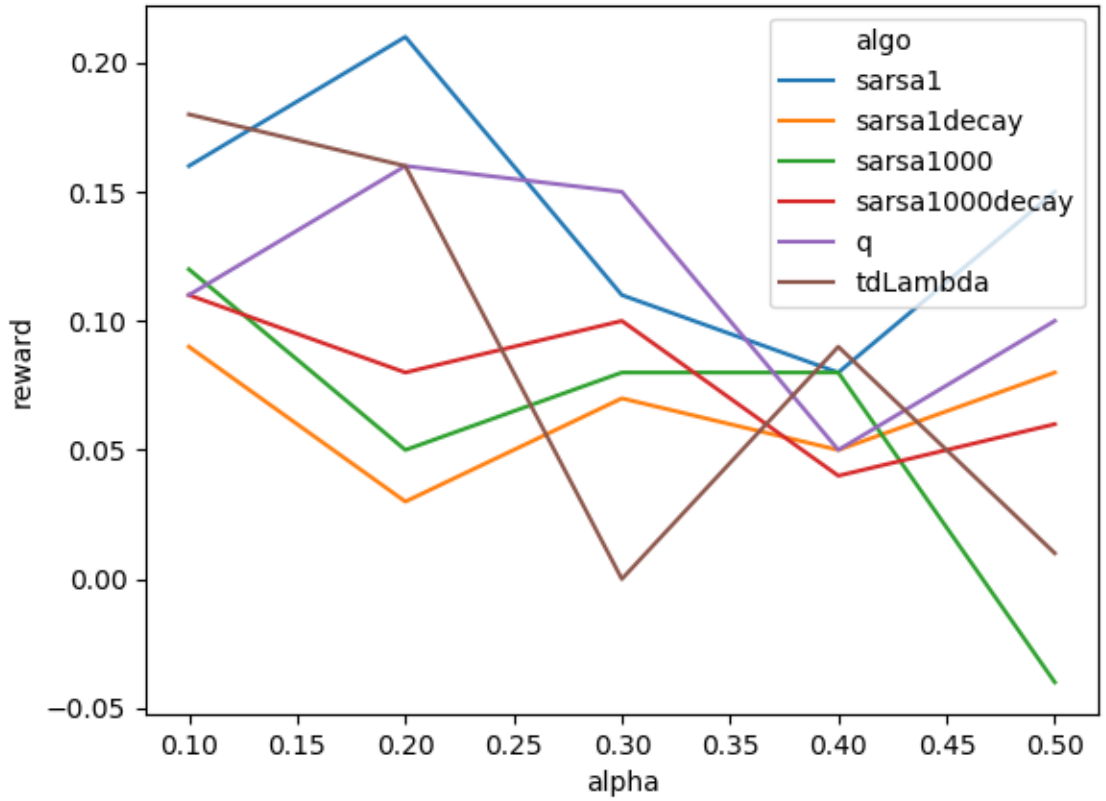
We observe that the performance values of 1 step and 1000 step SARSA decreases with increasing alpha. The general trend is first decrease then increase. This is because initially as alpha increases the variance of the model increases and it gives more weight to recent samples and not enough to current estimate which makes it biased towards recent samples a lot and not generalize. This reduces the performance. If we increase alpha more, performance increases slightly as higher alpha allows the model to update value functions faster and hence reaching convergence sooner.

Comparison of performance for SARSA with decay with varying alpha:



Here there is almost no difference in results, this is perhaps because in 100,000 episodes all algorithms with decay have already converged due to very low epsilon so almost deterministic policy. This gives almost same values for all alphas with some stochastic variance.

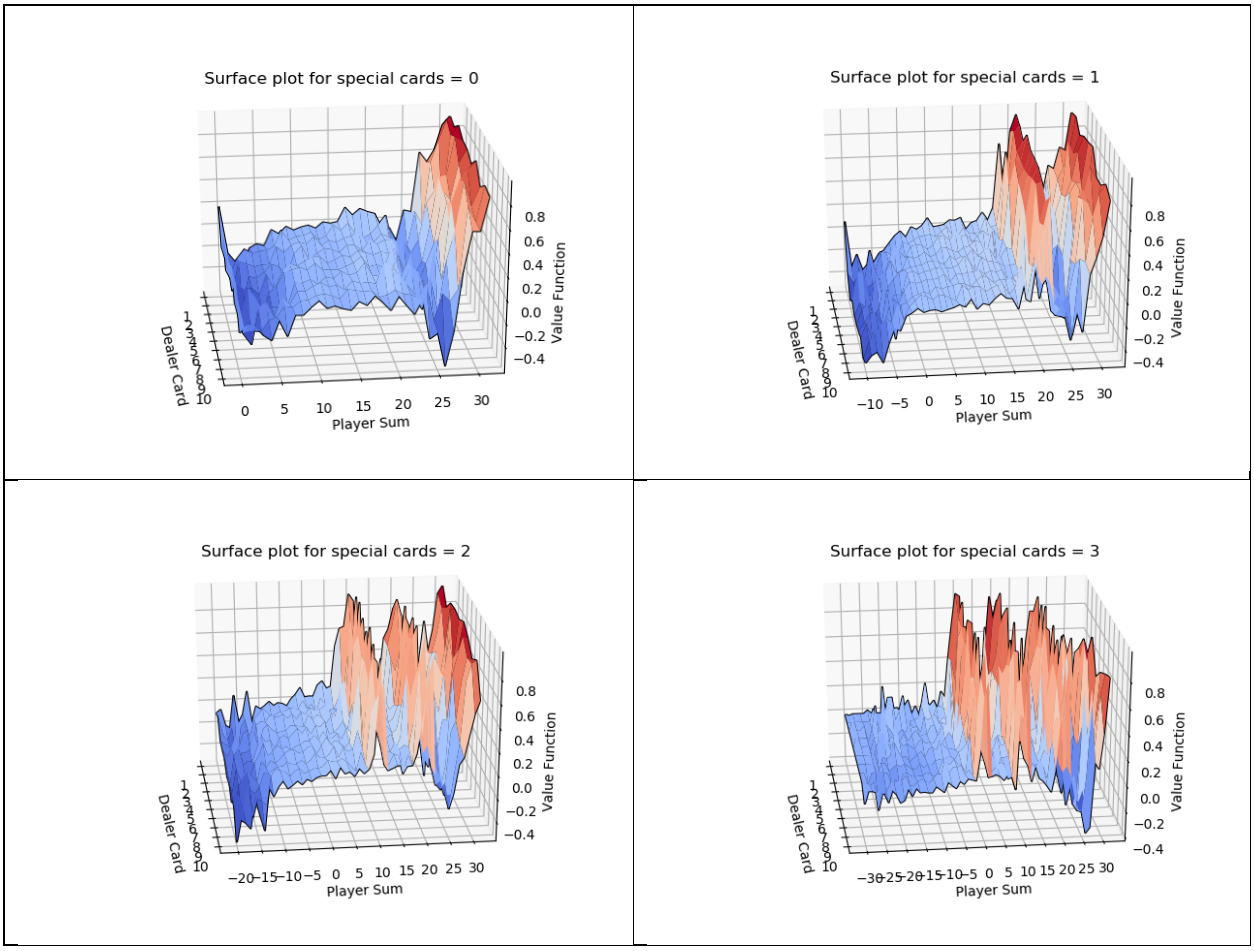
Comparison of performance for SARSA with decay with varying alpha:



Here, almost all algorithms have decreasing performance as alpha increases due to very high variance.

Value function of TD Lambda (0.5):

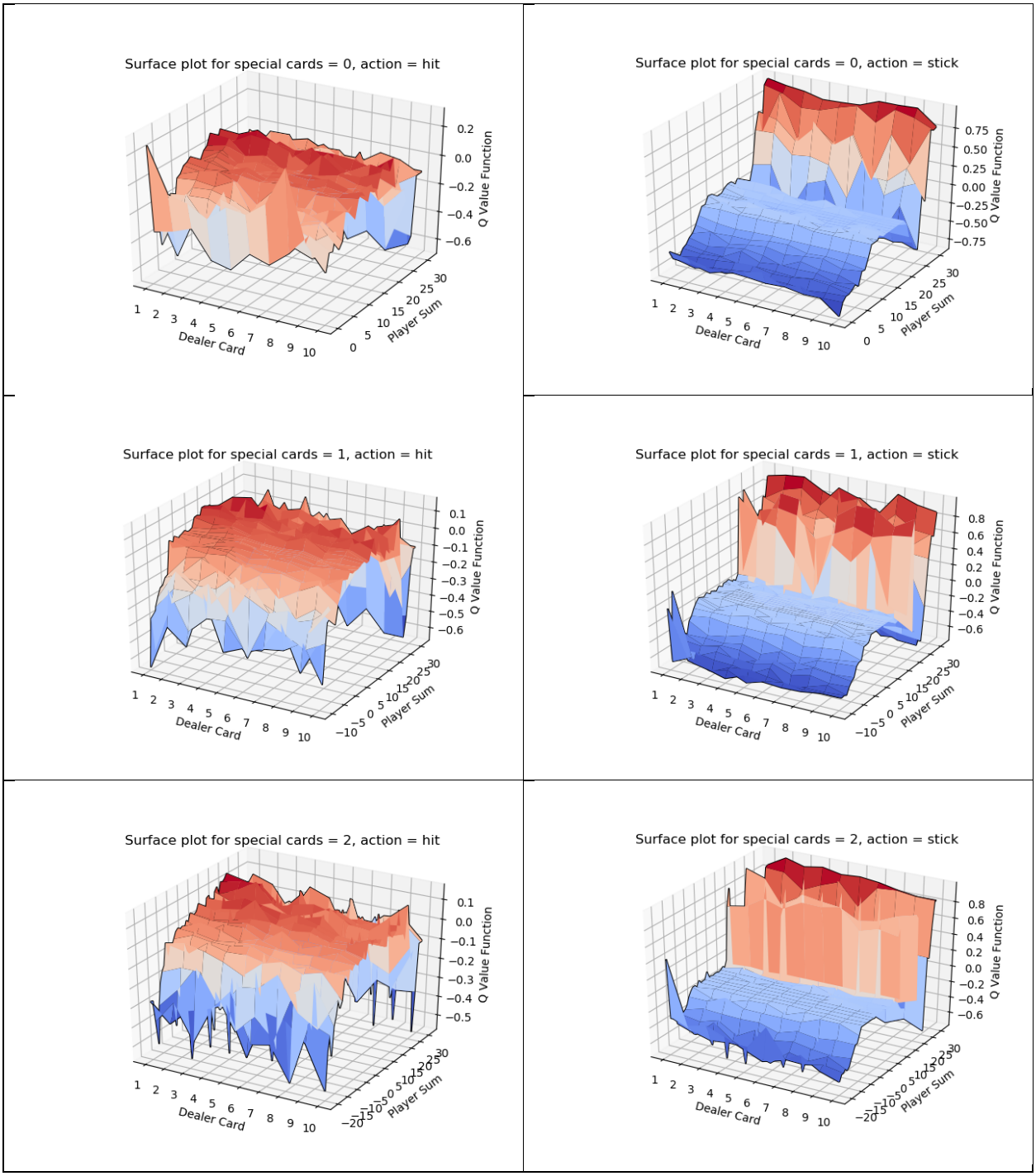
- Episodes: **1 million**
- Alpha: **0.1**
- Epsilon initial value: **0.1**
- Epsilon decay: **True**
- Lambda value: **0.5**

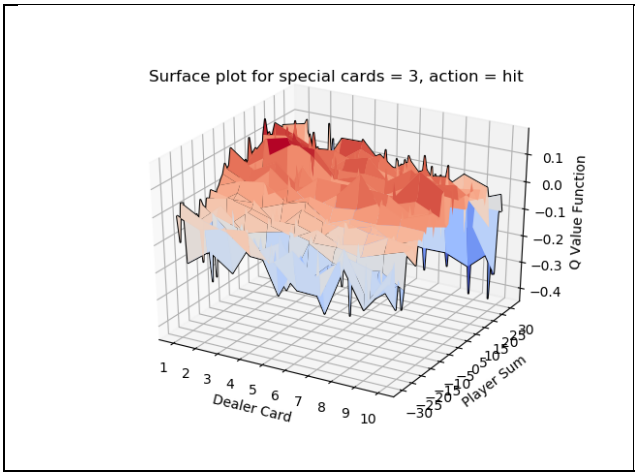


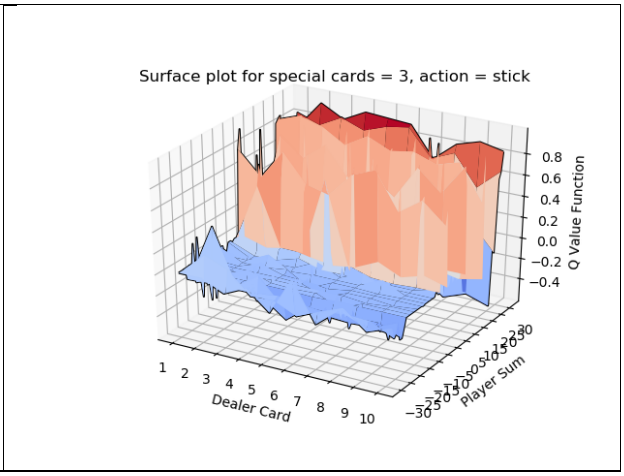
The optimal state value function is similar to the one obtained from the base policy because optimal policy is also very similar. Unlike the basic policy, here there is sudden drop in value function at 25 but not so deep drops for 15 or 5 hard sums. This is because the policy learns that the best idea is not to hit when hard sum <=25. It depends on what dealer card is to decide whether to hit or stick. As shown in the figure below the learnt policy has a blurry line between the hit and stick state clusters unlike a solid distinction irrespective of dealer card. For lower values of dealer card, the TD(0.5) decides that stick is a better option than hit. Also, when dealer card is high then TD(0.5) decides to hit even when it has sum >= 25 as shown by the blue spots in 2 or 3 special card case with low hard sum. This is because it can take this risk without being anxious of getting busted as it can use these cards to get lower sum than 31. Hence, number of special cards also is considered by this algorithms which was not considered by the base policy.

The policy learnt is quite different as shown below:

Q value function for TD(0.5) 1 million episodes, alpha 0.1:

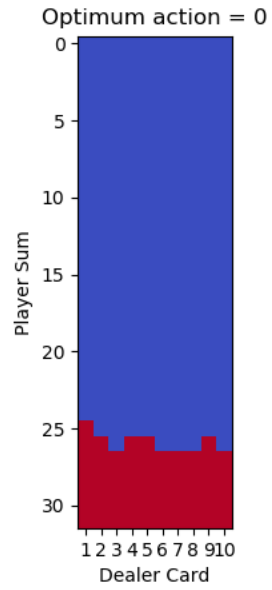
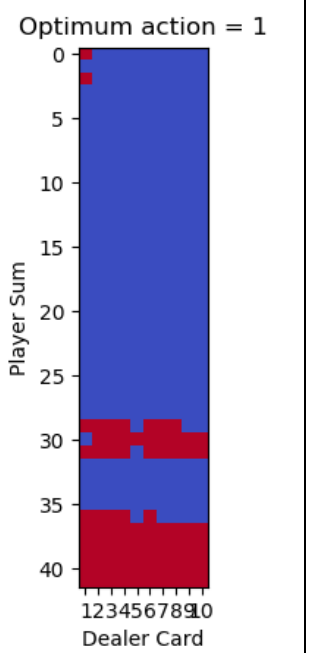
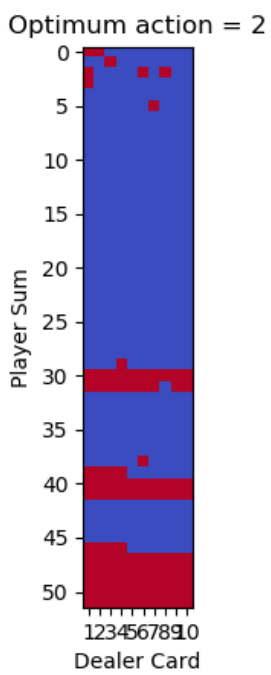
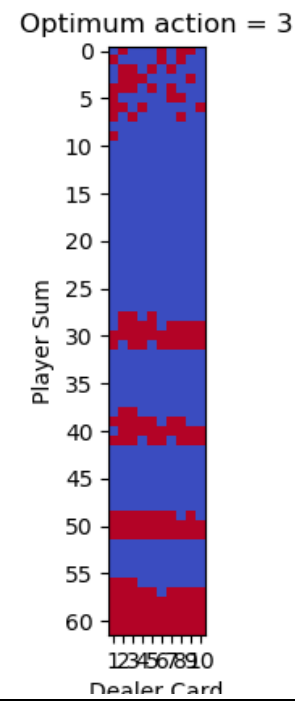






Policy learnt for TD(0.5):

Red = stick, Blue = hit

Special Cards = 0	Special Cards = 1	Special Cards = 2	Special Cards = 3
			

For any queries please contact:

Shreshth Tuli

shreshthtuli@gmail.com

+91-9911671223