
RL Assignment 1 Report

Srijan Sinha, 2016CS10322

The state has been represented based on three parameters :

- Player Hand Value -1 to 31, -1 represents negative sum as well as busted state
- Dealer Hand Value -1 to 10, it is the one card that the dealer gets in the beginning
- Value Type 0-2:
 - 0 represents a hand with no special cards
 - 1 represents a hand with special card where it is used at its lower value
 - 2 represents a hand with special card where it is used at its higher value

Out of these, any state with a negative integer is non actionable.

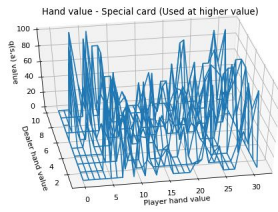
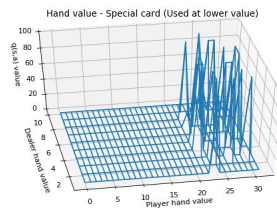
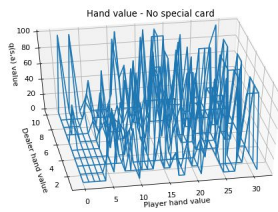
In addition to these, a state -1, -1, -1 is also taken as the start state, where the only action is to "DEAL" the cards.

Policy Evaluation

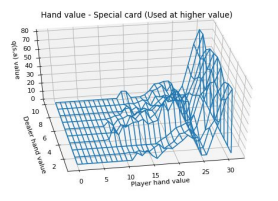
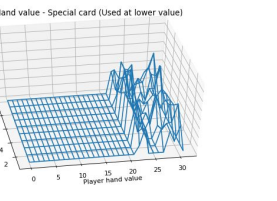
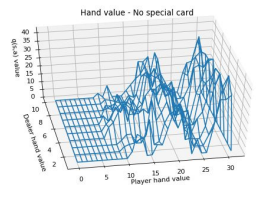
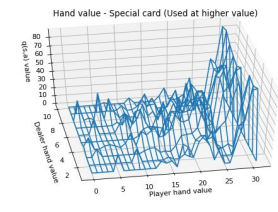
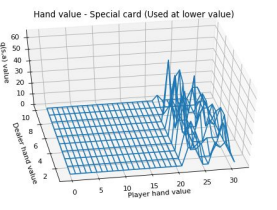
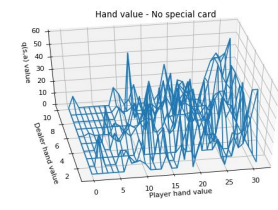
Monte Carlo

First Visit - evaluation over 1000 episodes for different number of runs

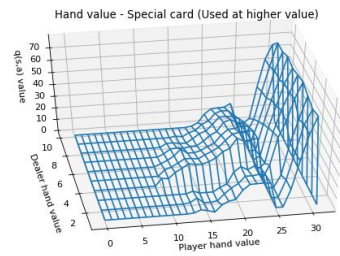
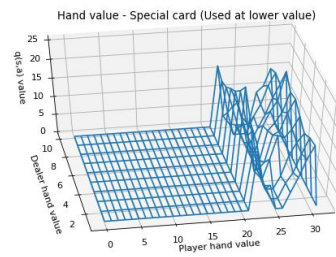
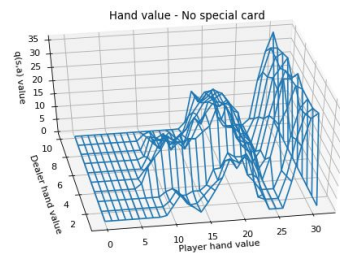
Monte carlo first_visit averaged over 1 runs



Monte carlo first_visit averaged over 10 runs



Monte carlo first_visit averaged over 1000 runs



Every Visit

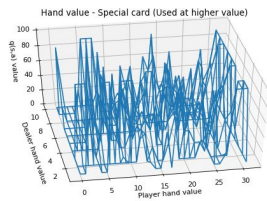
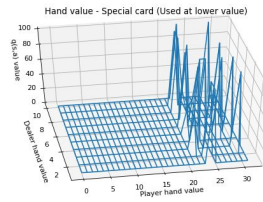
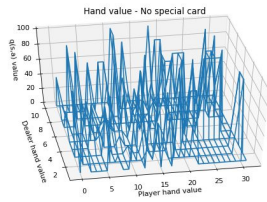


Figure 1: Hand value - No special card

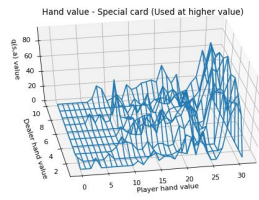
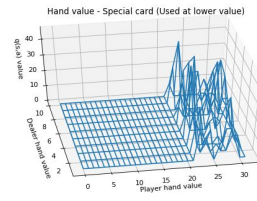
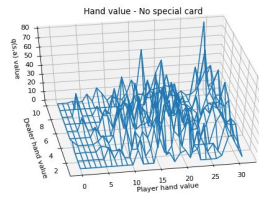
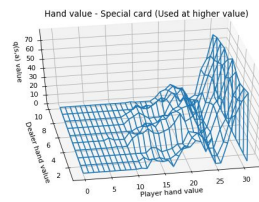
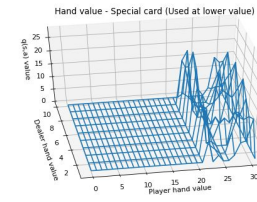
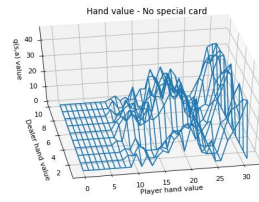
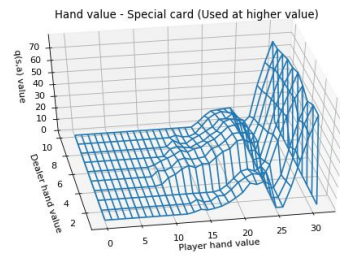
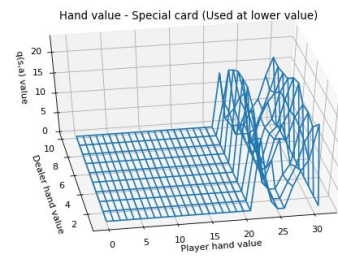
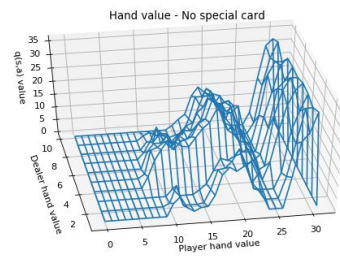


Figure 2: Hand value - No special card

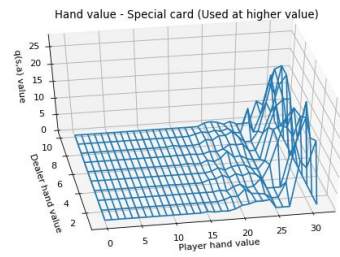
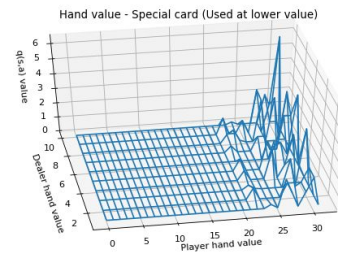
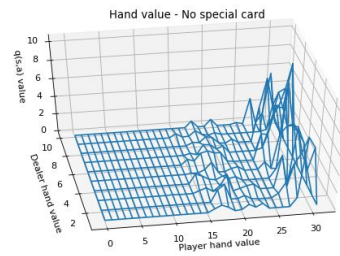


Monte carlo every_visit averaged over 1000 runs

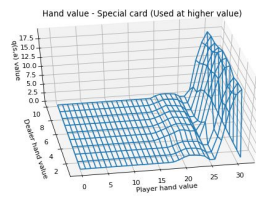
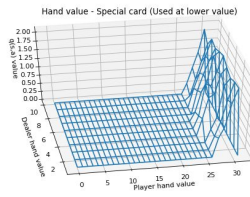
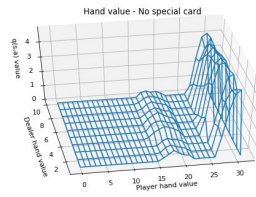


TD(1)

TD(1) averaged over 10 runs

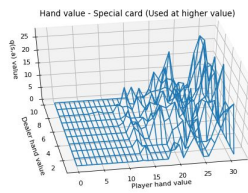
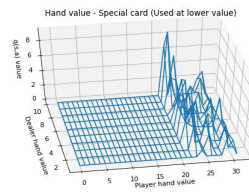
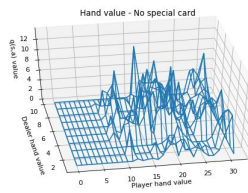


TD(1) averaged over 1000 runs

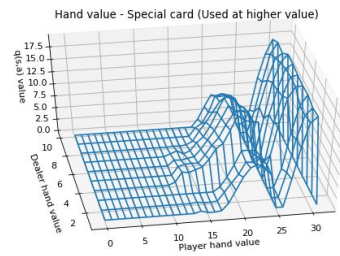
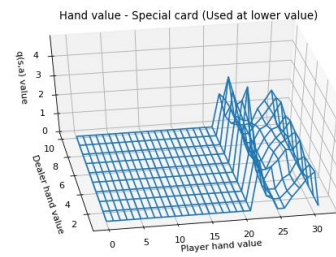
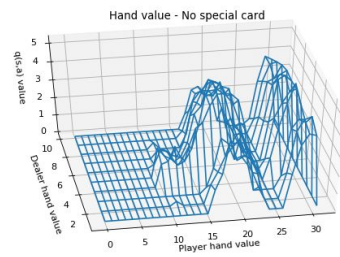


TD(3)

TD(3) averaged over 10 runs

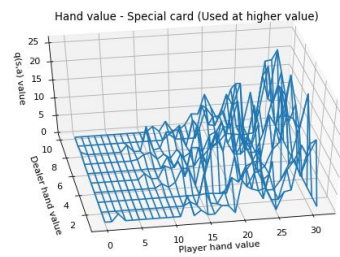
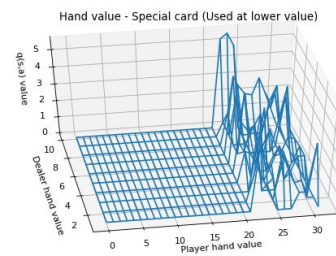
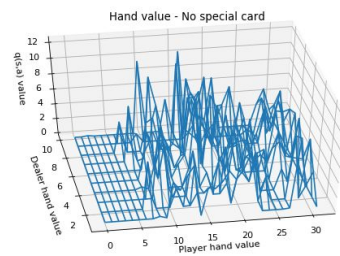


TD(3) averaged over 1000 runs

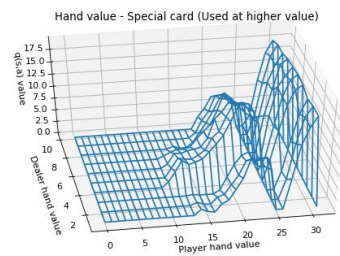
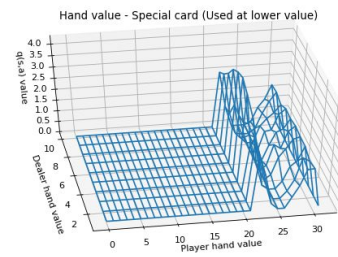
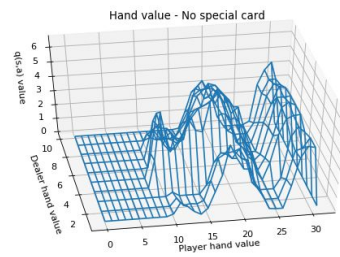


TD(5)

TD(5) averaged over 10 runs

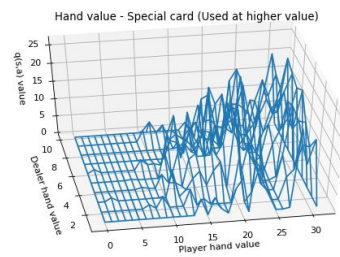
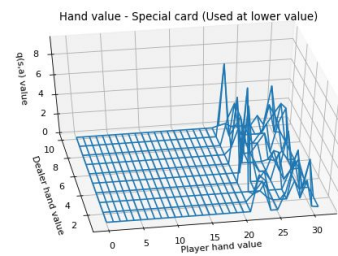
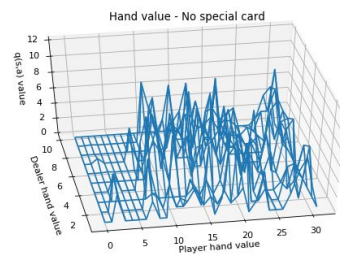


TD(5) averaged over 1000 runs

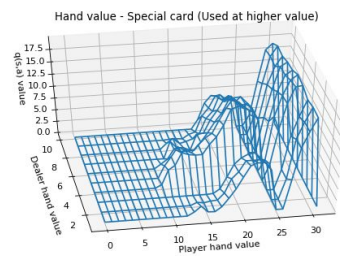
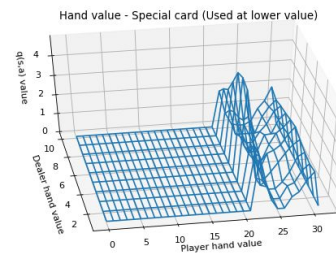
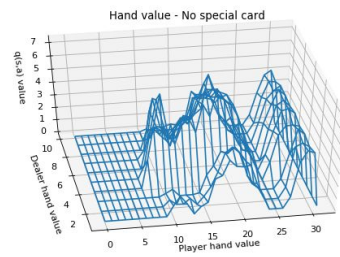


TD(10)

TD(10) averaged over 10 runs

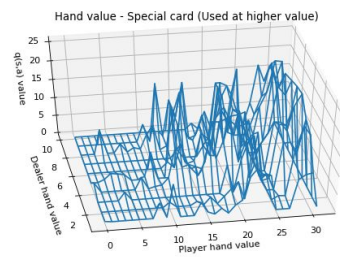
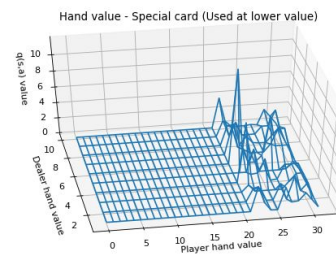
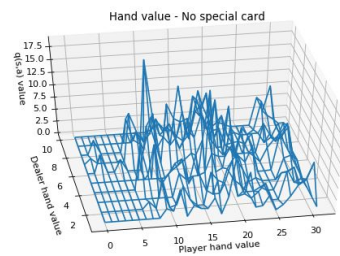


TD(10) averaged over 1000 runs

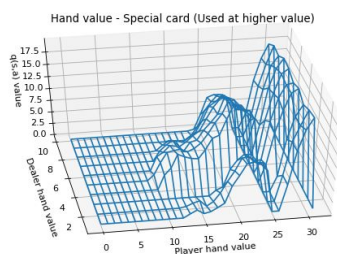
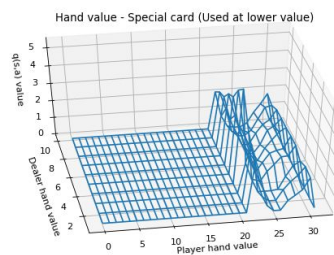
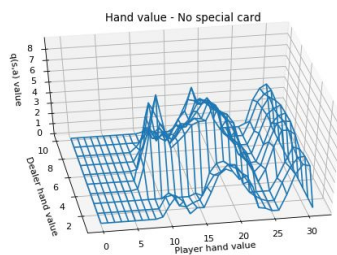


TD(20)

TD(20) averaged over 10 runs

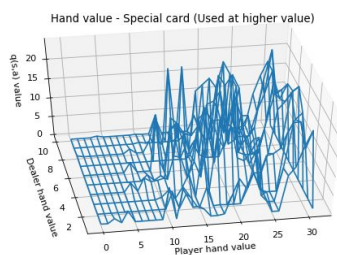
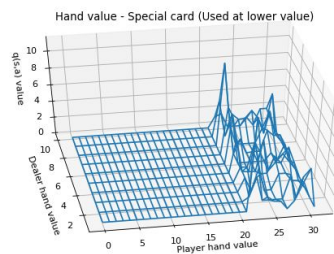
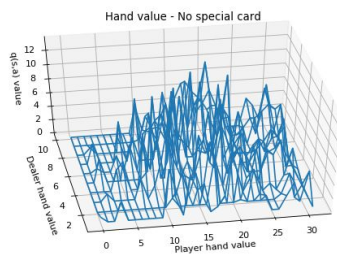


TD(20) averaged over 1000 runs

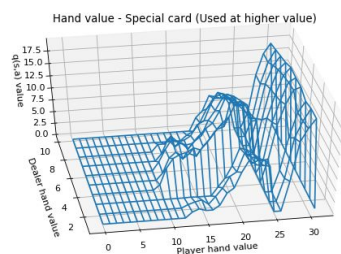
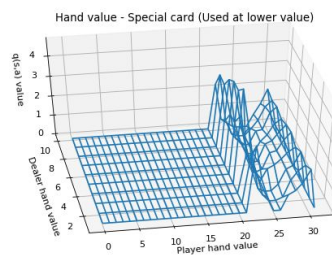
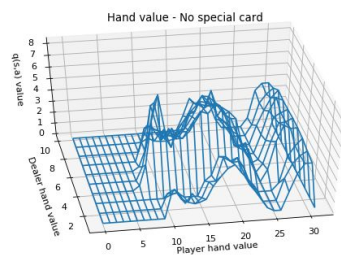


TD(100)

TD(100) averaged over 10 runs

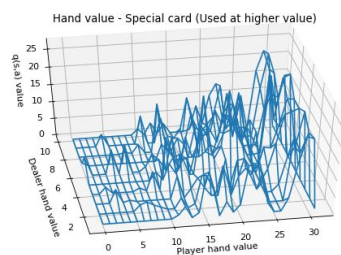
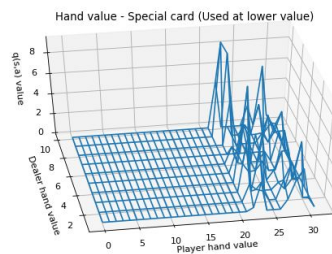
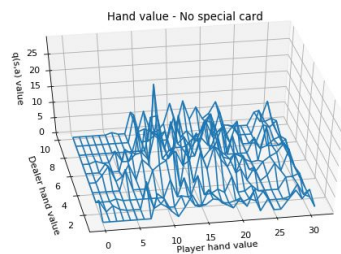


TD(100) averaged over 1000 runs

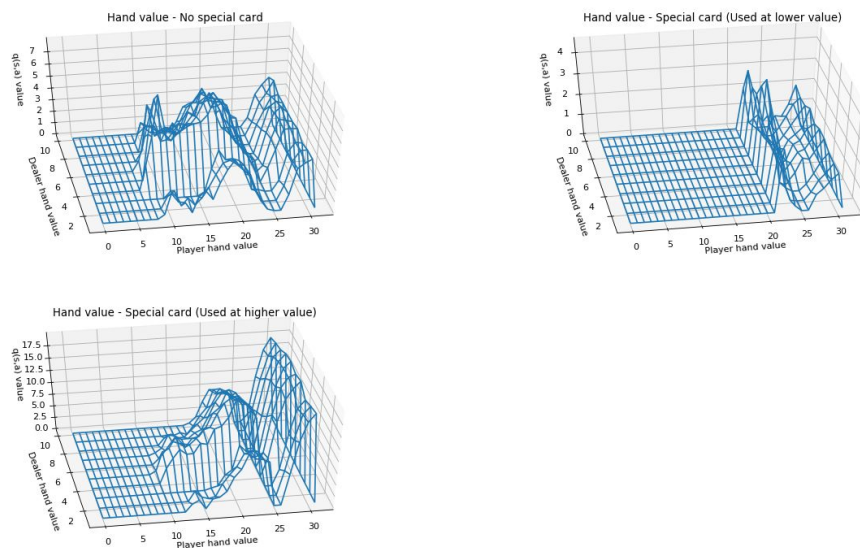


TD(1000)

TD(1000) averaged over 10 runs



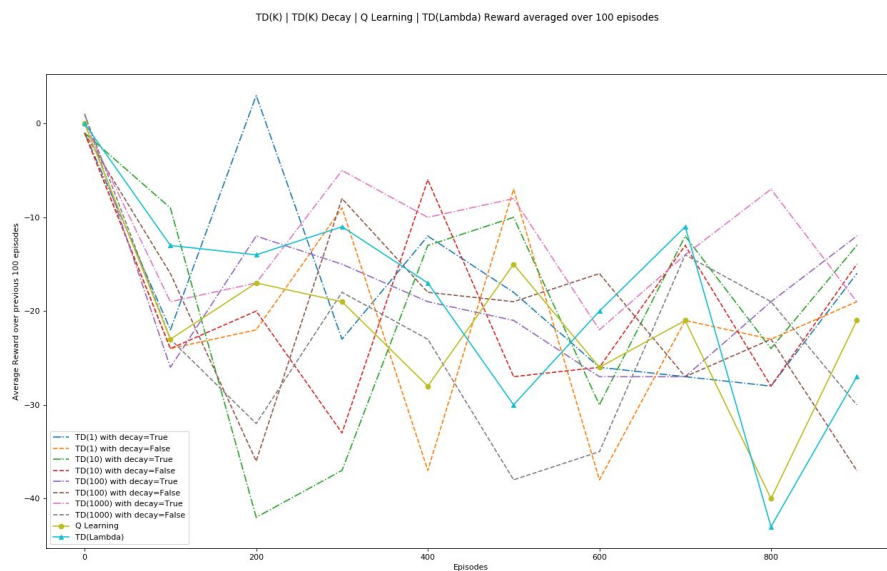
TD(1000) averaged over 1000 runs



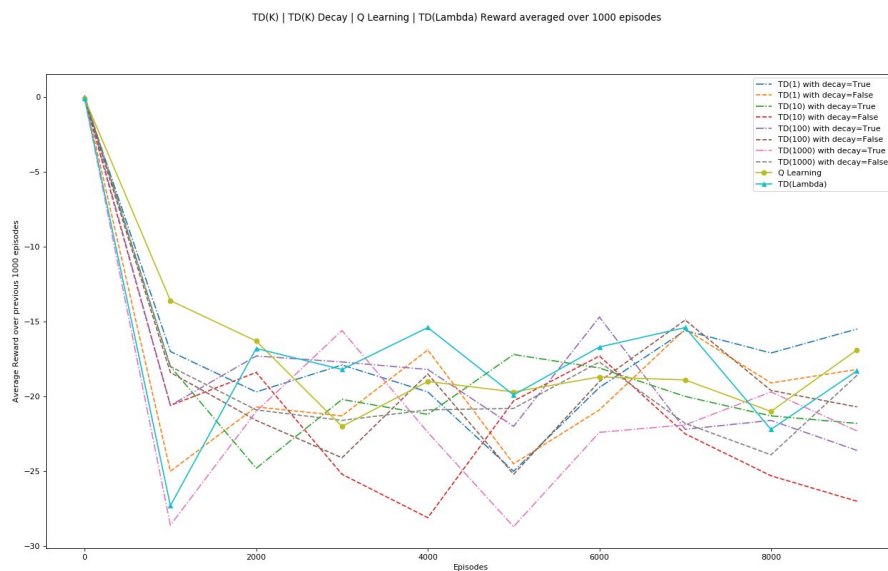
All the algorithms ultimately learn the same value function but every visit learns it the fastest due to more number of updates and TD(K) with $k = 20, 100, 1000$ can be seen as emulating MC (high variance in the beginning) while with $k = 1, 3, 5$ it is more like TD.

Average Reward while Training

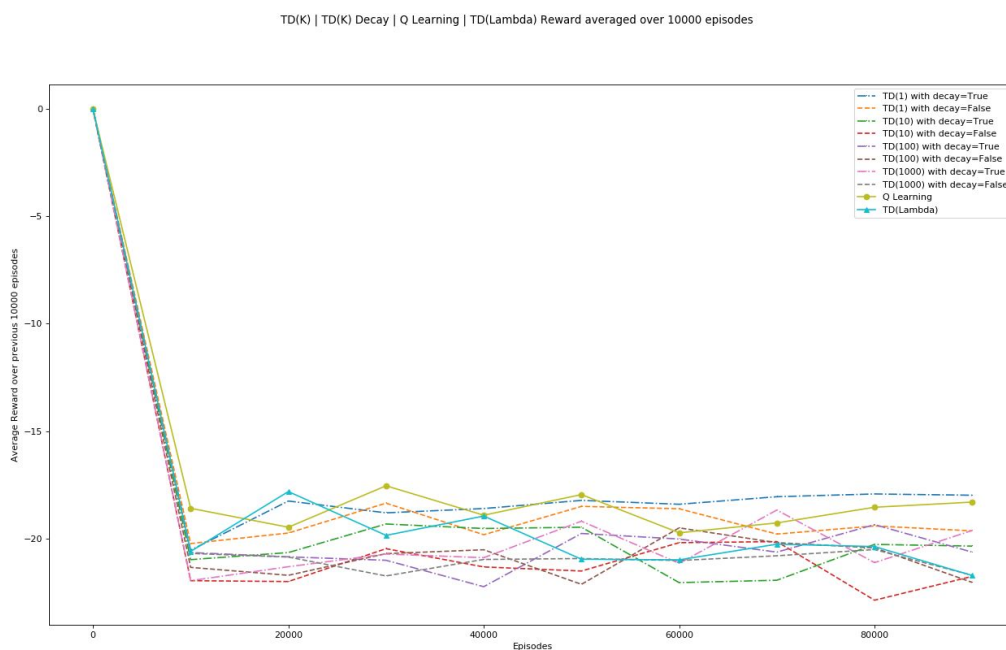
TD(K) vs TD(K) No Decay vs Q Learning vs TD(Lambda)



Trained upto 1000 episodes with average rewards taken every 100 episodes



Trained upto 10k episodes with average rewards taken every 1k episodes



Trained upto 100k episodes with average rewards taken every 10k episodes

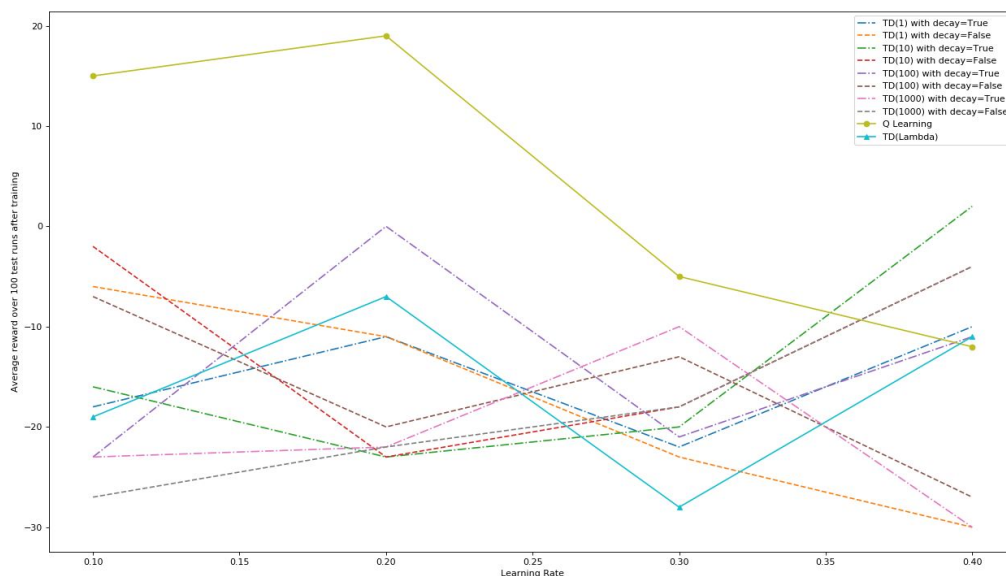
Here we can see that in the lesser training case the rewards are very varying and that is because of two things firstly the algorithm isn't really trained much throughout the 10k episodes and plus the initial distribution of cards has a major say when the rewards are averaged over only 100 episodes.

In 100k case the algorithm is sufficiently trained hence the reward is almost constant throughout in all the algorithms. And also due to 100k being a large number we can assume that the initial distribution of cards will be almost the same in all the reward batches.

Testing the policies after training with different alphas

TD(K) vs TD(K) No Decay vs Q Learning vs TD(Lambda)

I ran the process multiple times to check if the results are coherent

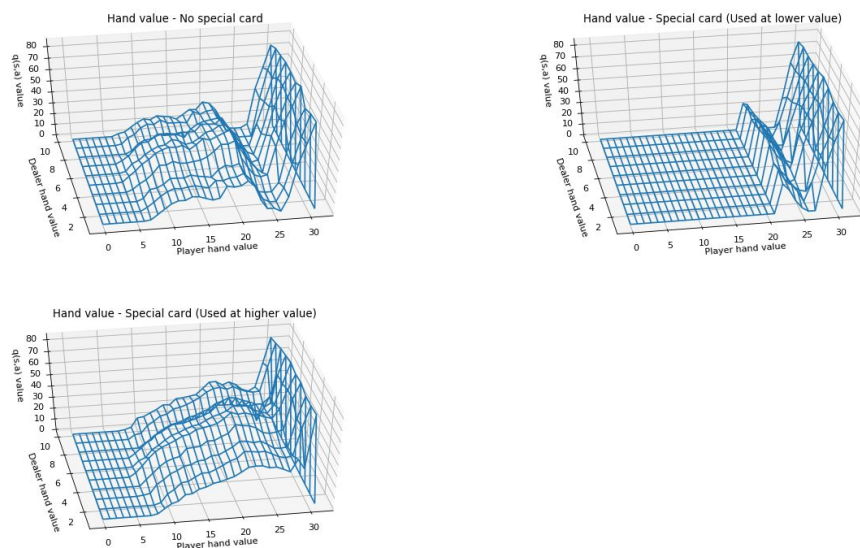


Simulation trained over 100k episodes

Most algorithms (TD with higher k) provide optimum learning at smaller alphas (0.1, 0.2) but some TD algorithms provide best performance with highest values of alpha this because small step TD algorithms have lower variance. $TD(\lambda)$ also provides best performance with highest alpha, since it is a weighted average over $TD(k)$ it will also have lower variance.

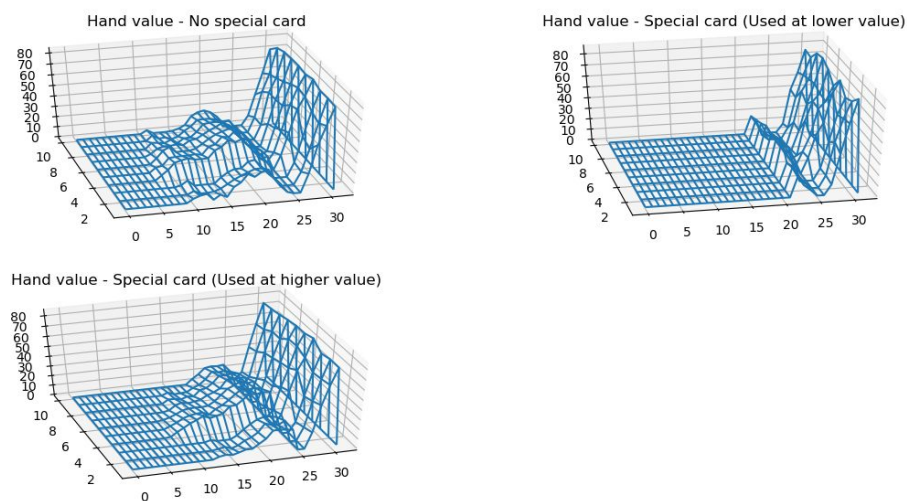
TD(λ)

Value function of policy obtained by $TD(\lambda)$ is as follows :




Trained over 1 million episodes and evaluated using a million episodes (Monte Carlo)

While the original policy's evaluation was:



'Stick above 25' policy's evaluation using MC

As it can be seen TD learns a policy which also optimizes value functions of states where user hand value is around 15 and dealer hand value is 0-2, while the original policy is poor at that. The learnt policy is also better when the hand value is near about 25 while the



actual policy has a sharp dip in value function due to sudden change of action.