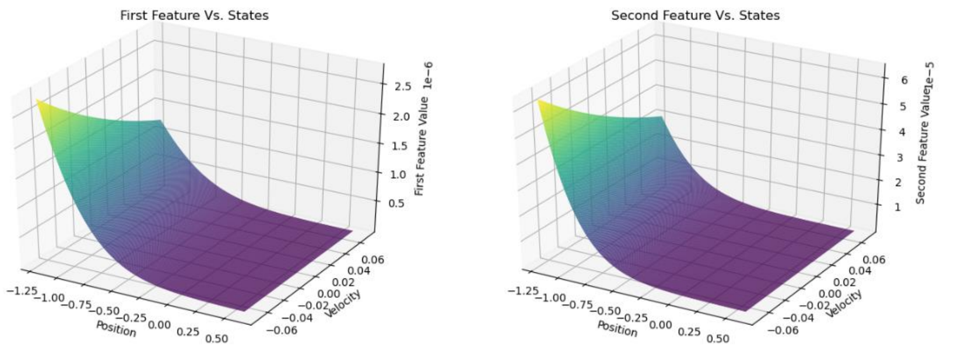
2. <u>Preliminary: The Mountain Car Problem and Feature Engineering</u>

1. The state space is representing the current position of the car and it's velocity, the values of the position are in range: [-1.2,0.6] and the velocity range is [-0.07,0.07]. The action space is {0,1,2} and used 2 - increase velocity, 0- decrease velocity or 1 - no change velocity.
   The reward is 1 if and only if the car reach to the flag and otherwise 0 for any state action pair.
2. In order to plot the features Vs the states we choose the parameter *number_of_kernels_per_dim* to be [10,8] as in LSPI. We sampled 40,000 different state.

   when examining the plots, we can see that both features changes the most in negative positions and negative velocities.
3. The advantages of encoding the state space using RBF's features instead of using the state directly in features is that RBF's first we don’t limit ourselves to the number of features, using RBF we can represent the state with various number of features. Also, because it uses gaussian grid it creates referenced point in space that way we can compare two states. One more advantage is the that we can use RBF to normalize the data.

3. LSPI:

1. LSPI version we saw in class is:

$$\hat{d}_n^k = \frac{1}{n}\sum_{t=1}^{n} \phi(s_t, a_t) r(s_t, a_t)$$

$$\hat{C}_n^k = \frac{1}{n}\sum_{t=1}^{n} \phi(s_t, a_t)(\phi^T(s_t, a_t) - \gamma\phi^T(s_{t+1}, a_{t+1}^*)),$$

$$\theta_k = (\hat{C}_n^k)^{-1}\hat{d}_n^k.$$

where $a_{t+1}^* = \arg\max_a \hat{Q}^{\mu_{k-1}}(s_{t+1}, a) = \arg\max_a \theta_{k-1}^T \phi(s_{t+1}, a)$.

The problems with that version is first when we reached to terminal state the only action available is stay in it, regardless the greedy policy. because in our environment the agent get reward only in terminal state in its goal is to maximize the reward to the greedy approach is to get out the terminal state that the agent could return it and earning another reward.

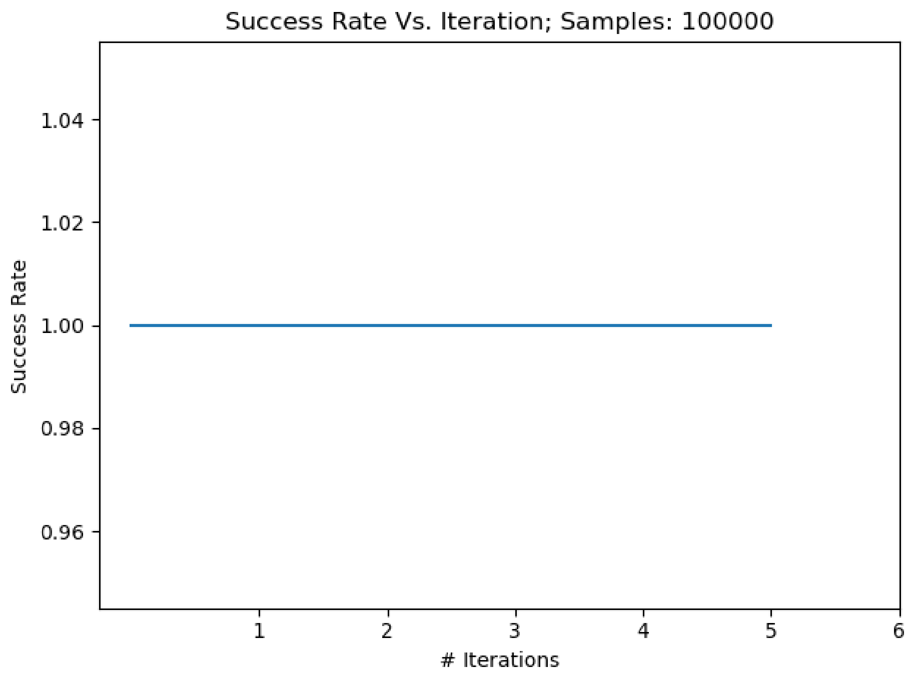
So, for a sample with done flag turn on we use: $\phi(s_t, a_t)\phi^T(s_t, a_t)$ for C

2. The states mean and standard deviation is

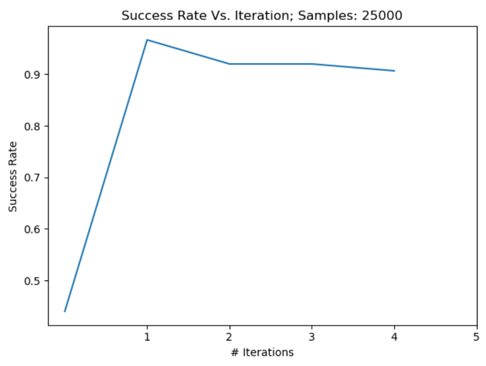
Parameter	Mean	Standard Deviation
Position	-0.299093411	0.51891497
Velocity	0.0000702371796	0.04028207

3. For each action there is vector of weights and for each feature so in the described model each state is represented by 80 features and there are 3 actions therefore 240 weights (in code includes bias so 243).

5. Chosen seed are: 123,234,456, evaluation is based on 300 steps per game

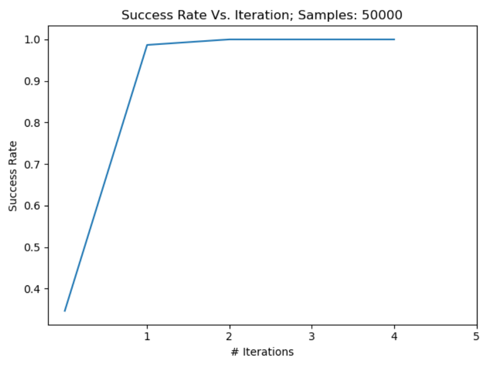


6. For 25,000 samples:

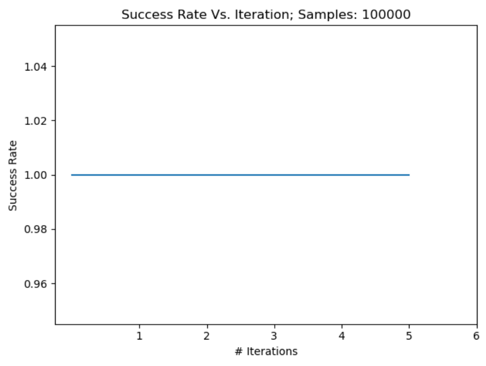


The agent did not complete the learning.

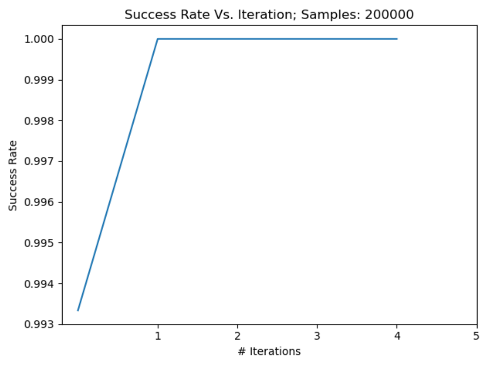
For 50,000 samples:



For 100,000 samples:



For 200,000 samples:

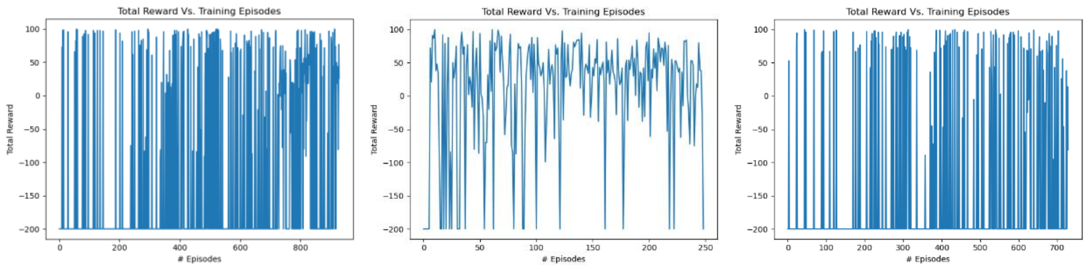


As we can see from the plots, as long we sampled more the success rate is higher and convergence to 1.0 in less iterations. Also as mentioned in the exercise's PDF 100,000 is enough to fast coverage.

4. <u>Q-Learning:</u>

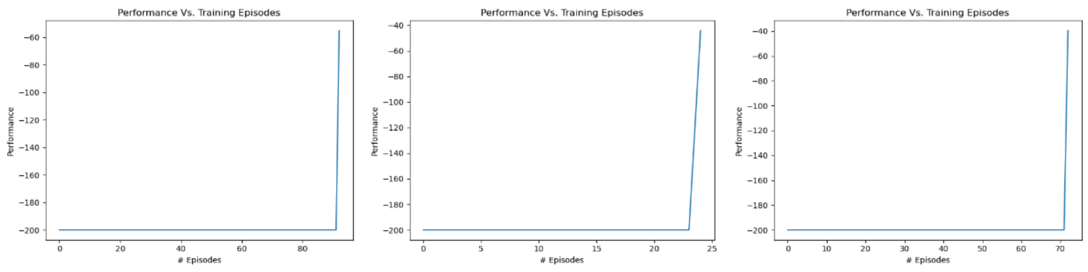
1. In order to consider an episode as successful the car should reach the peak in at most 176 steps, 175 steps it is not on the peak state so the reward is -175 and in the last step it is on the peak so +100. The transformation of the reward function affects the agent to want to reach to the peak in minimum steps because for every action which doesn't get it to the peak it get punished also the high positive reward can propagate to earlier steps using the discount factor better than lower reward.
3. Seeds are 100, 200, 300.

   Total reward Vs. Episode:



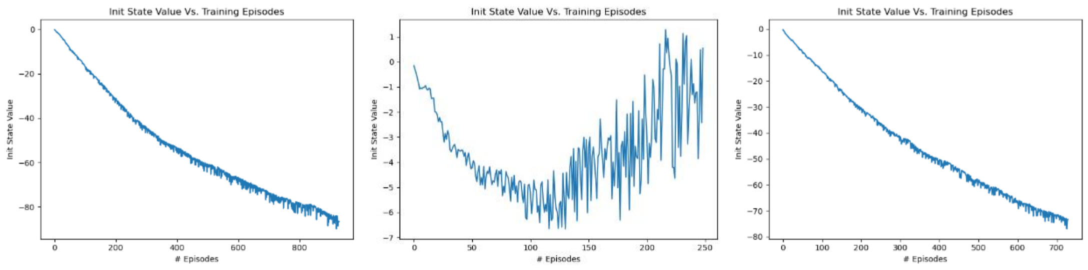
   All the plots indicate that most of the training episodes result with no reaching to the peak.

   Performance Vs. 10 Episodes:



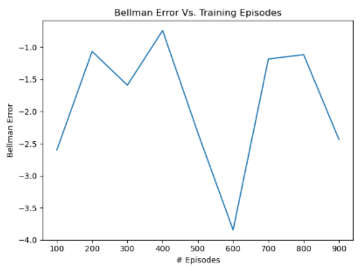
   All the plots indicate there is no gradual progress to obtain a cumulative reward of at most -75, also the cumulative reward is around -50.

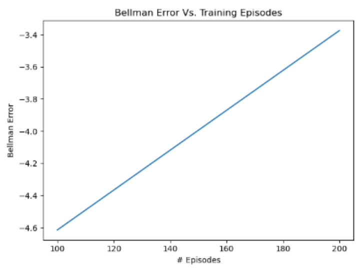
   Init State value Vs. Episodes:

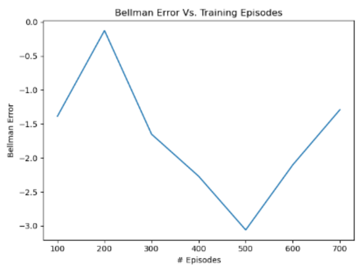


   The plot indicates that the agent might need dozens of steps before reaching the peak.

Bellman Error Vs. 100 Episode:



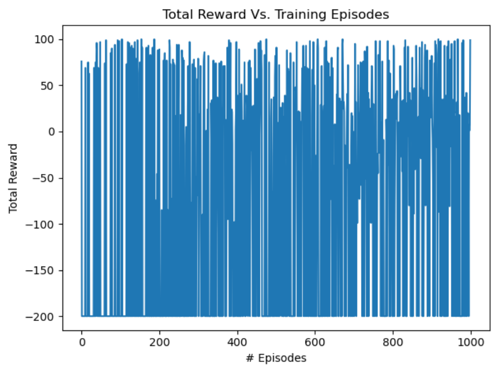




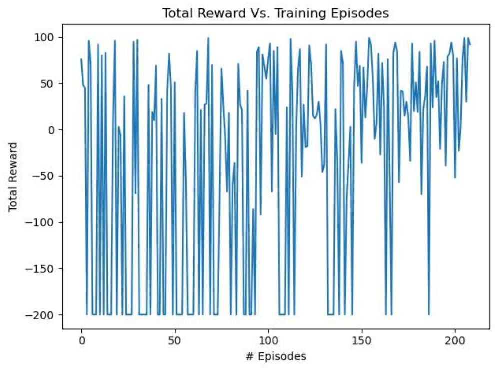
When we see big changes in the bellman error it is indicate of success episodes because the rewards of success episode is much bigger than failure and this is where the learning is reflected.

4. To determine what is the best value for epsilon the training is based on 1,000 episodes maximum.

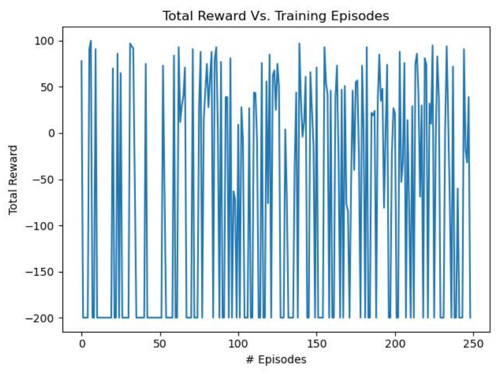
$\varepsilon = 0.01$:



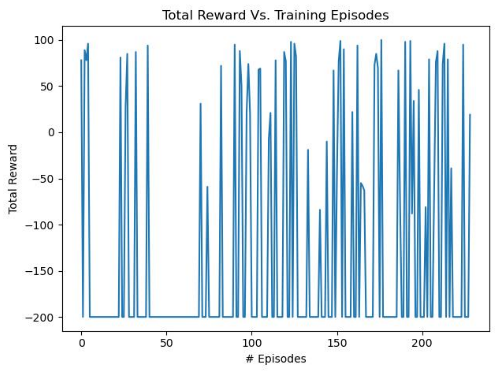
$\varepsilon = 0.3$:



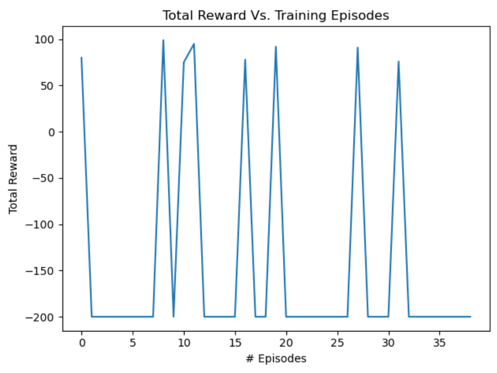
$\varepsilon = 0.5$:



$\varepsilon = 0.75$:



$\varepsilon = 1.0$:



From the plots we can understand that the best epsilon value is around 0.5 because for 0.01 most of the episode the agent didn't reach for the peak and for 0.3,0.5,0.75 it reaches to the peak after around 200 training episodes. Although, for the value of 1.0 it reaches to the peak after 40 training episodes we can't really say that the agent succeed to generalize the problem and finding the best weights for the approximation.