CS 747: Programming Assignment 3

Aman Singh : 190020010

Task 1

The hyper-parameters I used to solve this task are :

```
self.t1_x_states = 90
self.t1_v_states = 35
self.epsilon_T1 = 0.4
self.learning_rate_T1 = 0.2
```

The x-range is divided into `t1_x_states` slots and v-range is divided into `t1_v_states`. So, there are `t1_x_states`*`t1_v_states` states in the tabular version of the task. There are 3 weights, one for each action which hence adds up to the weight vector having 3*`t1_x_states`*`t1_v_states` dimensions.
As can be seen in the figure below, due to the high value of `epsilon_T1` we are not greedily avoiding the sub-optimal action. This in turn causes the reward to remain low for the first 3000 episodes. Even after a winning state is found, the algorithm still continue to explore other paths and hence the training reward remains low. On testing, however, our algorithm always takes the optimal action according to it's weights and ends up averaging a reward of -147.
Small number of states were not chosen as the algorithm was not found to converge to good enough solution. And as the number of states increased, exploration (`epsilon_T1`) had to be increased too, so that elements in the weight vector are appropriately updated.

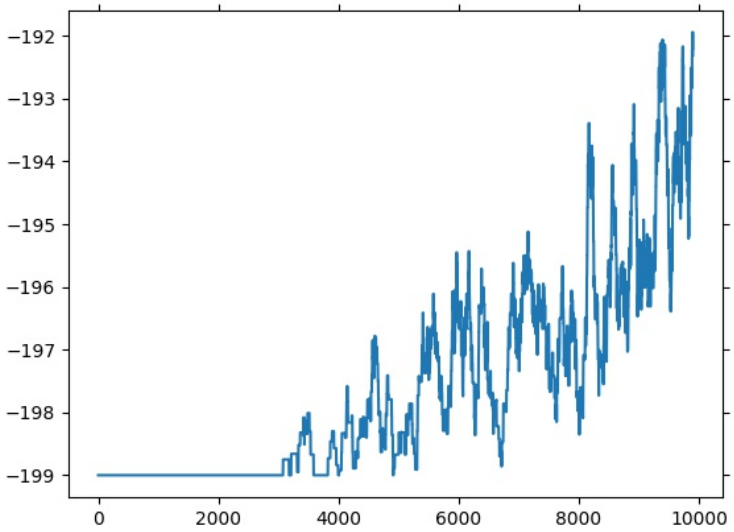

Figure 1: Reward in task 1 training

Task 2

The hyper-parameters I used to solve this task are :

```
self.t2_tiles    = 10
self.t2_x_states = 18
self.t2_v_states = 14
self.epsilon_T2 = 0.01
self.learning_rate_T2 = 0.1
```

In this task tile coding was used for linear function approximation. The amount of tilings used were `t2_tiles` and each tiling had `(t2_v_states+1)*(t2_x_states+1)` 2 dimensional tiles that needed to be remembered and managed. As was explained in the class, each action had it's own bunch of tilings. The tile coding hyperparameters were selected by eliminating variants that did not perform well and to account for the fact that one dimension was distance and other velocity.
As can be seen in the figure, due to the low exploration rate i.e. low value of epsilon_t2, the optimal paths are the only ones that are explored in depth which leads the algorithm to quickly converge to a reward of -104 (in around 1000 episodes). The reward fluctuates a bit due to small amount of exploration, no other strategies to lower reward are found to increase the reward further.

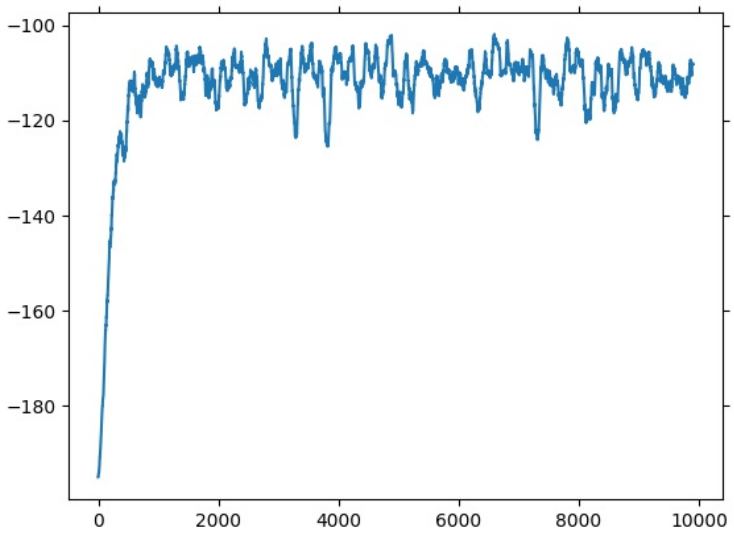

Figure 2: Reward in task 2 training