# Programming Assignment 3

Harshil Naik (190010030)

## *Task 1:*

The weights array is NxNx3x2 dimensional.
Weights are updated at every step according to the Sarsa(0) algorithm.

```
def sarsa_update(self, state, action, reward, new_state, new_action, learning_rate, weights):
    weights[int(state[0]), int(state[1]), int(action)] += learning_rate*(reward + weights[int(new_state[0]), int(new_state[1]), int(new_action)].dot(np.array(new_state)) \
    - weights[int(state[0]), int(state[1]), int(action)].dot(np.array(state)))*(np.array(state))
    return weights
```

Features are extracted by discretizing the environment using a suitable step size.

```
def get_table_features(self, obs):
    # numStates = 50
    delta0 = (self.upper_bounds[0] - self.lower_bounds[0])/self.numStates
    delta1 = (self.upper_bounds[1] - self.lower_bounds[1])/self.numStates
    position = (obs[0] - self.lower_bounds[0])/delta0
    velocity = (obs[1] - self.lower_bounds[1])/delta1
    if position >= self.numStates:
        position = self.numStates - 1
    elif position < 0:
        position = 0
    if velocity >= self.numStates:
        velocity = self.numStates - 1
    elif velocity < 0:
        velocity = 0
    return [position, velocity]
```

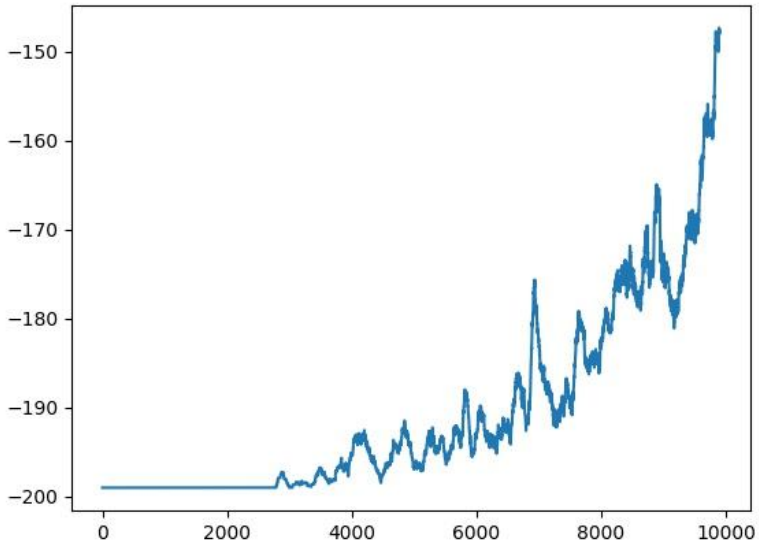
Actions are chosen based on an epsilon greedy approach.

```
def choose_action(self, state, weights, epsilon):
    if np.random.random() < epsilon:
        return self.env.action_space.sample()
    else:
        action = np.argmax(weights[int(state[0]), int(state[1])].dot(state))
        return action
```

**Sarsa(0) weight update :**

$$w^{t+1} \leftarrow w^t + \alpha_{t+1}\{r^t + \gamma w^t \cdot x(s^{t+1}) - w^t \cdot x(s^t)\}x(s^t).$$

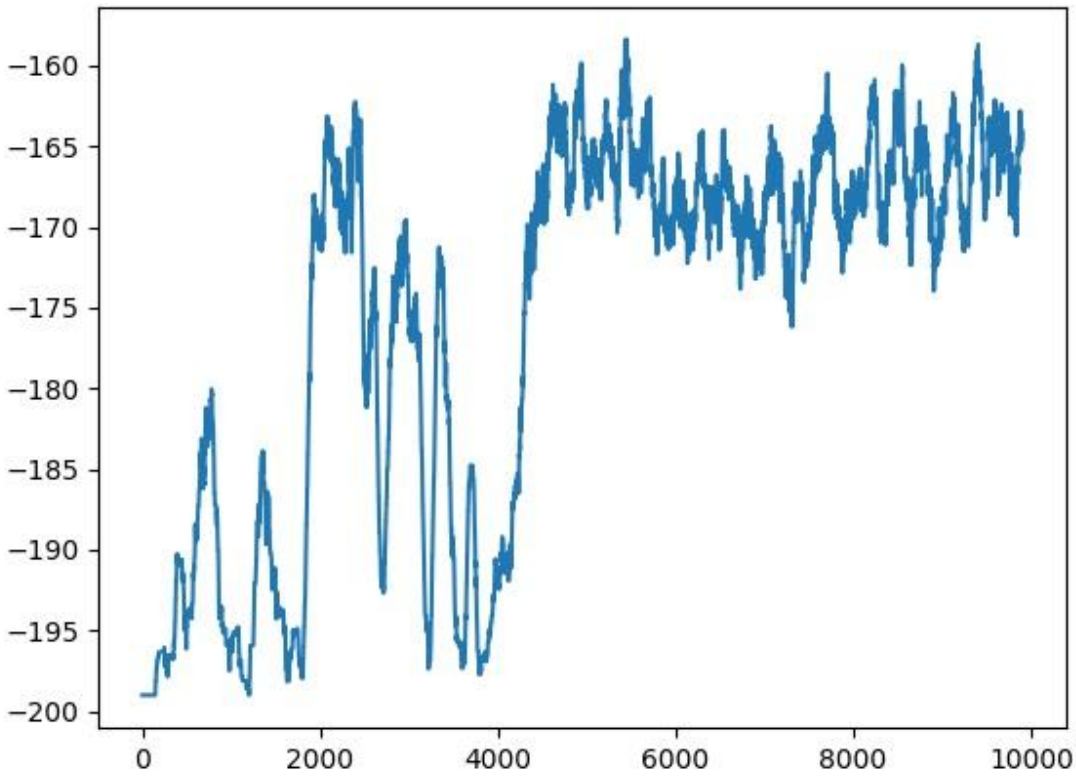
**Observations:**
The results are best for a learning rate of 0.00008, and exploration probability of 0.0001. The rewards start to rise after a few episodes, and rise almost exponentially uptil the last episode.




## *Task 2: Better Features*

To get better features in task 2, we use Tile Coding, with 3 tilings, and 3 divisions in each tiling, that divide the given space equally into 3 parts. The other functions remain the same, such as the choose_action function and the main implementation function.

The learning rate is __ and the exploration rate is ___ to give a rewards of




***References:***

https://harshil3004.gitbook.io/reinforcement-learning/
https://www.geeksforgeeks.org/expected-sarsa-in-reinforcement-learning/