CS747

Assignment 3: Report

190050118
Sudhansh

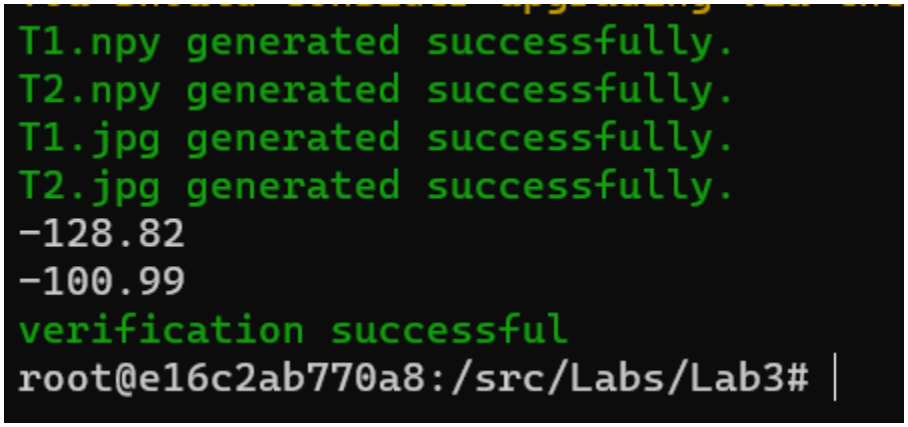
Question 1

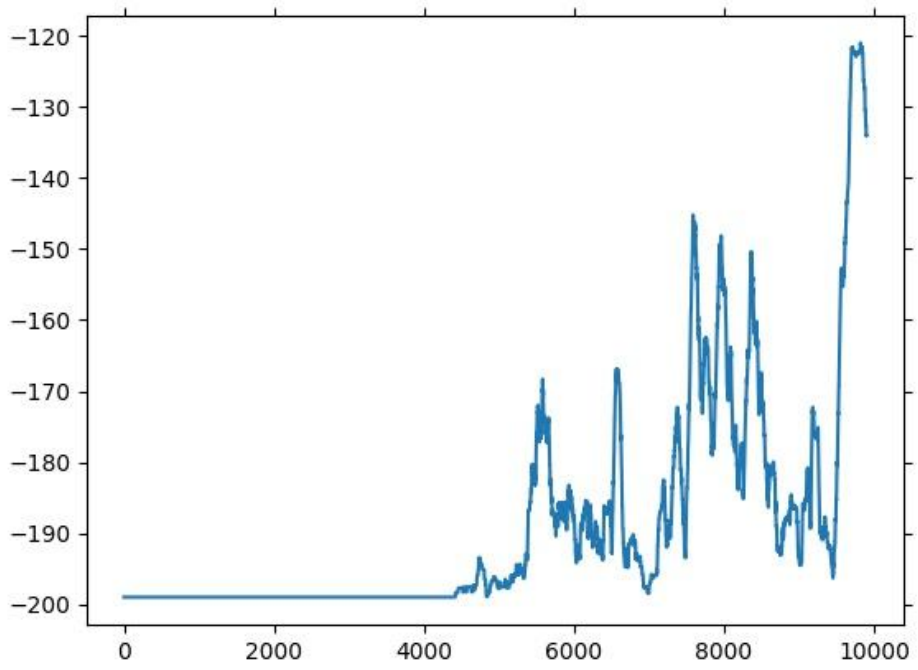
Implementation:

- Firstly, I discretized the input space into 8*8 units in the distance * velocity domain. After this, I expanded the state space to incorporate the three actions for the car. Therefore, in the end, I had 8*8*3 = 192 states. I tried with 10, 20, and 6, but 8 gave the best results.
- Then, I defined the states as a one-hot vector that represented the above-mentioned discretized input space.
- Now, w*x(s) represents the Q value of the respective state and action. The agent is trained using the SARSA update rule.

The main issue with the implementation of the agent was tuning the hyperparameters. After experimenting with various parameters, I found that epsilon in the order of 1e-2 and learning rate in the order of 1e-2 gave the best results. Here are the plots obtained for epsilon = 0.02 and learning rate = 0.01.

As we can see, the reward starts at -199. The agent does not start learning until 4k epochs and this may be due to the seeding and hyperparameters. However, these hyper parameters gave the best results. After 4k, the behavior is stochastic (spiky), with an average of around ~-170. The reward increases sharply in the end. Finally, the agent reaches a state where the average test reward is -128.82.

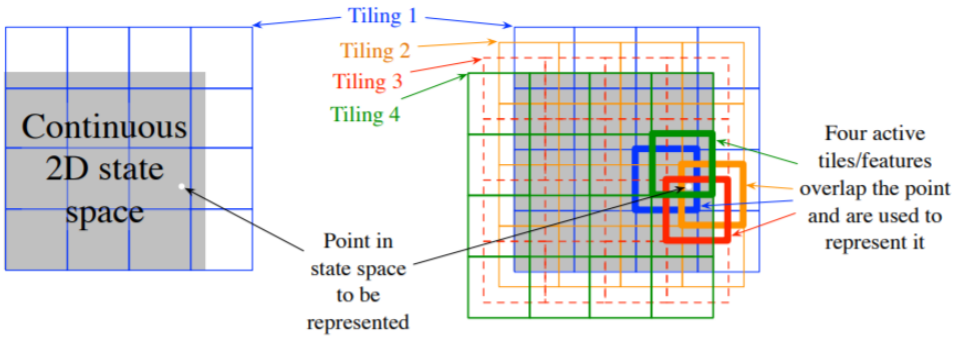




Question 2.

Implementation

- Similar to the above case, we have weights for each action separately. The aim is to represent the Q function using w*x(s), but we implement it via **Tile Coding**.
- I chose eight tilings in both the distance and velocity domain. I implemented conjoined tiling - tiles are shared across features. As in, the 2D space of distance and velocity is segmented into multiple 2D tiles.
- Specifically, I chose the number of tiles to be 9 (approximately in the input domain) along each direction. Basically, I considered a half tile that go outside the input domain.
- Given an input, the state is given by the on-hot vector representing the tiles present on the input. See the following image for a clear understanding.
- I chose the number of tilings to be 8 as it seemed to have given the best results. I tried with 4 and 6 tilings too.



- Then, the Q value of the state-action pair is given by the sum of weights across all the tiles that overlap with the input.

This implementation gives much better results due to more generalization. In fact, we see the improved test reward to be -100.99. Here is the training plot obtained.

We see that there is a sharp increase in the reward in the first 1000 epochs. The training seems much more stable than in the previous case. The output is not as stochastic, and the agent converges quickly! We see the final training reward to converge around ~-115. The learning rate is 0.01, and epsilon is 0.03 here.