

CS 747 - Foundations of Intelligent and Learning Agents

Programming Assignment 3

Prakhar Vijay Diwan, Roll no. 180100083

Note:

Task 1 and task 2 are completed as specified.

Seed values of 0 were given in the code structure, & were left undisturbed and used as is.

I have initialized weights for both the tasks as all 0s.

The hyperparameters used for tasks have been written in the code itself.

1. Task 1: "Tabular" Sarsa

In this task I have discretized the states (x,v) into 100 states using a 10 by 10 grid. So position (x) and velocity (v) had 10 splits uniformly along the range. So, in Tabular Sarsa(0) case, I simply used a binary feature vector for state representation, and here the weight values were directly the $Q(s,a)$ values. The feature and the weight arrays were taken of the same size. The update to weights done was the one corresponding to Q -values update stated in the slides (corresponding to Sarsa(0)). The action was chosen according to ϵ -greedy policy (with constant ϵ as required), which chose random actions with probability ϵ and otherwise chose greedily w.r.t. \hat{Q}^t .

All this functionality was implemented by filling in the 3 functions: *get_table_features*, *sarsa_update* & *choose_action*.

I got the following plot for **training**:

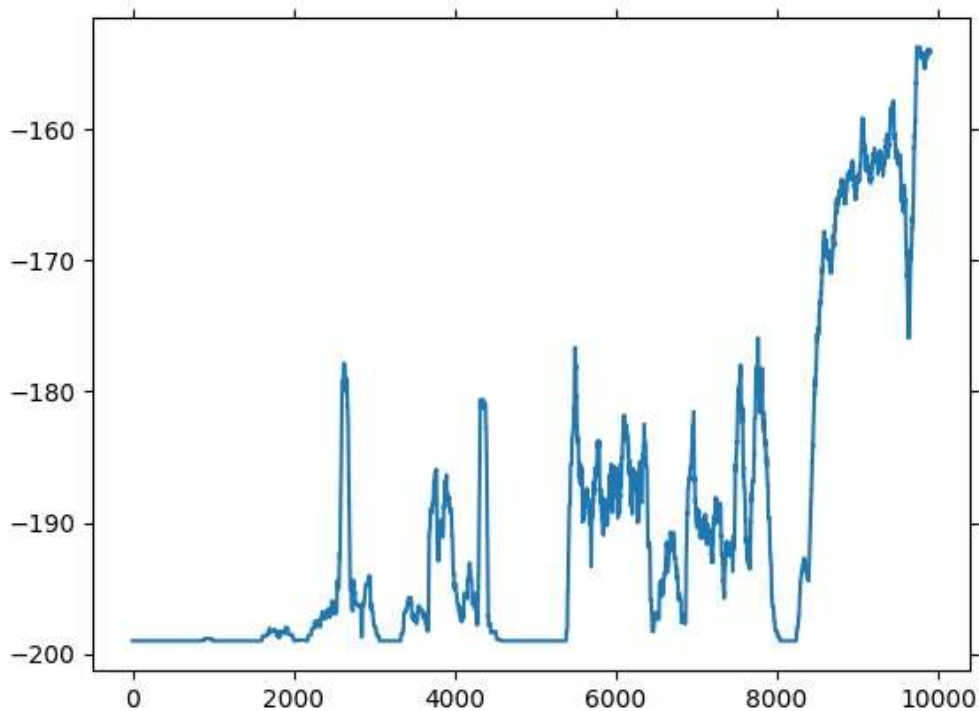


Figure 1: T1.jpg

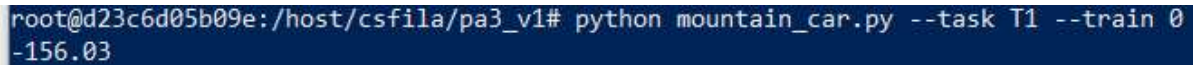
Observations:

From the plot we can see that for some episodes towards the end it reach the desired reward values of above -160.

Tuning of hyperparameters was observed to be highly critical, since even for some slight tweaking long term reward went below -160. I noticed that the ϵ value: if I chose it very low or very high then it led to poor results, same was true for discretization of state space; hence optimal values for them had to be obtained for finishing this task successfully.

Output:

The screenshot of terminal depicting the average reward for 100 tests is shown below:



```
root@d23c6d05b09e:/host/csfila/pa3_v1# python mountain_car.py --task T1 --train 0
-156.03
```

We can see it clearly satisfies the above -160 criteria.

2. Task 2: Sarsa with Linear Function Approximation

In this task, I came up with the feature vector using the tile coding approach. Here I used 10 tilings and same discretization as in task 1. Here I just introduced a dimension in both feature vectors and weight matrix which depicted the tiling number. Accordingly, I modified the binary feature vector obtained by calling *get_better_features*; so accordingly a 1 was given to the tile activated in each tiling. The offset between the different tilings was chosen as per the lecture. So after this I had to generalize the functions *sarsa_update* & *choose_action* for both the tasks. The update done to weights of tilings was same as done in task 1, as per the slide in lectures:

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

So I just added a dimensionality corresponding to tiling in both feature and weight vector, and I was able to generalize for task1 and task2.

I got the following plot for **training**:

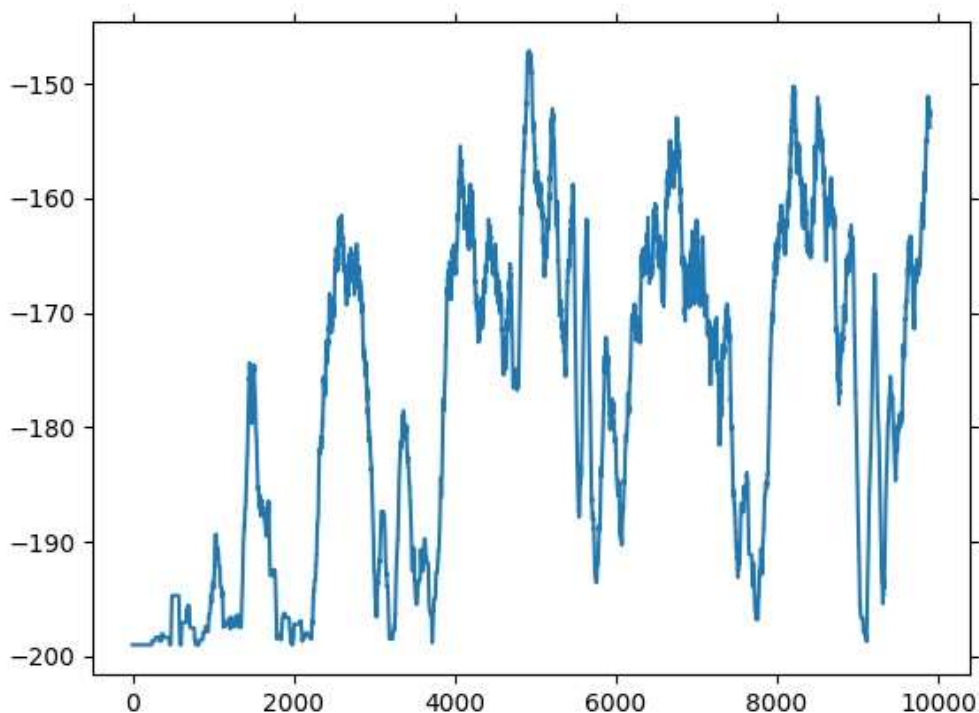


Figure 2: T2.jpg

Observations:

From the plot we can see that for this task, the plots do not reach around -130, this can be due to the fact that here the reward achieved is higher and hence random actions in the middle due to ϵ -greedy policy cost a lot harshly. So the values obtained during training are around -150. But when we run the tests, which have $\epsilon = 0$ in their *choose_action* function, implying that actions are never taken through random choice, which leads to goal (escaping valley) being achieved quickly as we see in the average reward of 100 tests (shown in output snapshot below).

Hyperparameter tuning in this task took a lot longer in this task and was even more critical. This was due to newly added hyperparameter "number of tilings" which had to be optimized as well; along with the greater time required for training in this task. I chose number of tilings as 10 finally.

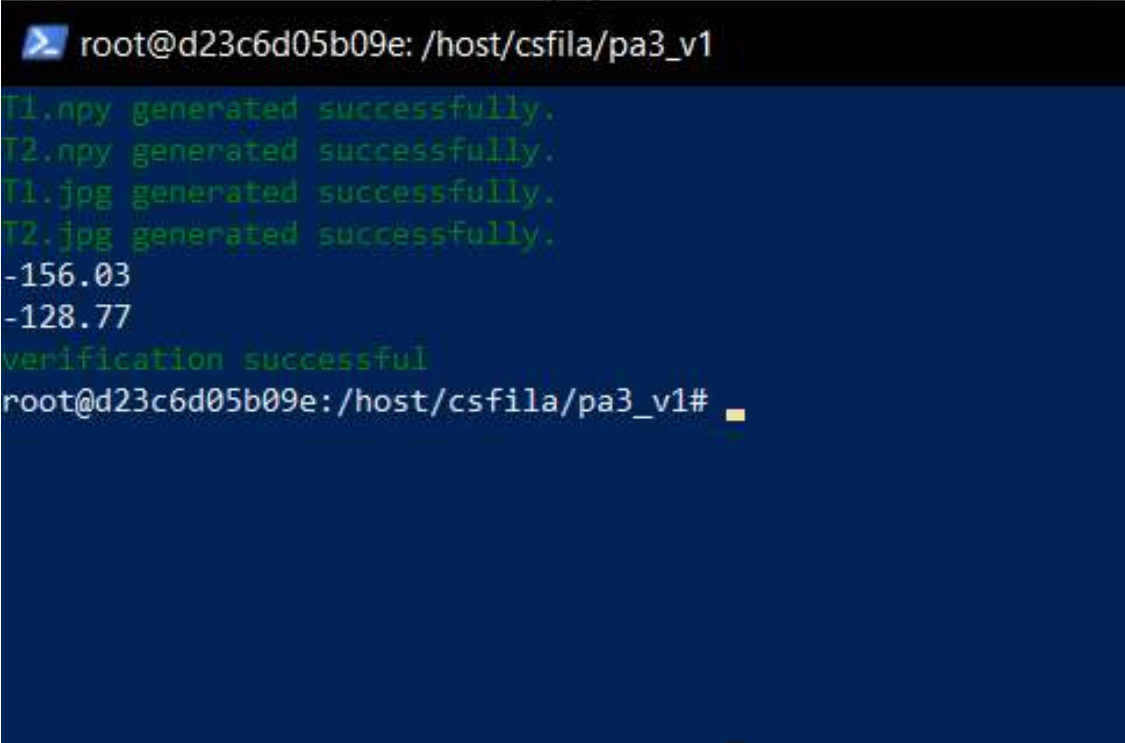
Output:

The screenshot of terminal depicting the average reward for 100 tests is shown below:

```
root@d23c6d05b09e:/host/csfila/pa3_v1# python mountain_car.py --task T2 --train 0  
-128.77
```

Figure 3: Output for Task 2

We can see it clearly satisfies the above -130 criteria.

OUTPUT obtained after running verifyOutput.shA terminal window with a dark blue background and a black title bar. The title bar contains a blue icon of a terminal and the text 'root@d23c6d05b09e: /host/csfila/pa3_v1'. The terminal shows the output of a script: 'T1.npy generated successfully.', 'T2.npy generated successfully.', 'T1.jpg generated successfully.', 'T2.jpg generated successfully.', '-156.03', '-128.77', and 'verification successful'. The prompt 'root@d23c6d05b09e:/host/csfila/pa3_v1#' is followed by a yellow cursor.

```
root@d23c6d05b09e: /host/csfila/pa3_v1  
T1.npy generated successfully.  
T2.npy generated successfully.  
T1.jpg generated successfully.  
T2.jpg generated successfully.  
-156.03  
-128.77  
verification successful  
root@d23c6d05b09e:/host/csfila/pa3_v1#
```

The code for the assignment can be found along with this report.pdf file in the submission.tar.gz file. The other required files are also attached inside it.