Swarada Bharadwaj, 18D170031

CS 747 Assignment 3 - Report

Task 1:

Linear Sarsa is implemented in the following manner:

- The weight vector is independent of action taken, and the state representation is dependent on the state (x,v) as well as action taken (a).
- The state space is discretized into 14 discrete steps for velocity (with 0.01 width), and 18 discrete steps for position (with 0.1 width).
- The feature matrix has dimensions (3, 14*18*3). Each row contains a representation for (x, v, a) (like a flattened 3-D matrix of dimensions (3, 14, 18), while the 3 rows are to be filled with differently activated features depending on the action taken. Each feature corresponding to a particular x, v interval is replicated thrice in each row for each of the 3 actions.
- The feature vector consists of binary features – 1 if a feature is activated, and 0 if it isn't.
- For a given state represented by (x1, v1), the following feature matrix is formed for i=0, 1, 2:
  - In row i, to represent state if action i is taken, the feature corresponding to x1, v1 and action i is activated. Everything else in the row is set to 0. A feature corresponding to x1, v1 is a feature corresponding to the discretized x and v intervals which contain x1 and v1.
- The weight vector is of size 14*18*3, and holds the Q(s,a) values.
- Training is done with gradient descent, as in any other linear approximation of Q.

Task 2:

In the second task, 2-D tile coding has been implemented to form a feature matrix of the same type as in Task 1. The number of tiles along the x axis and v axis are constructed similarly to the discretized intervals of Task 1, but there are 4 tilings, each with an offset from the previous one. The tiles are rectangular, and the offset is different along the x and v axis.

There are 3 sets of tilings to represent each of the 3 actions. Similarly to Task 1, the feature matrix has dimensions (3, 14*18*3). Each row has a feature representation for x,v (essentially, a representation for each tile) and a (each 14*18 vector is replicated in that row 3 times to correspond to the 3 actions).

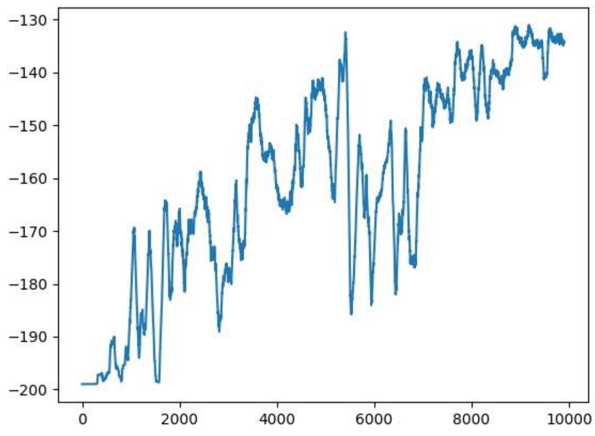
- The feature vector consists of binary features – 1 if a feature is activated, and 0 if it isn't.
- For a given state represented by (x1, v1), the following feature matrix is formed for i = 0,1,2:
  - In row i, to represent state if action i is taken, the feature corresponding to x1, v1 and action i is activated. For each of the 4 tilings, the tile intervals containing x1, v1 are found and the features corresponding to that tile is activated in the part of the vector corresponding to action i. Everything else in the row is set to 0.
- The weight vector is of size 14*18*3, and Q(s,a) is obtained when its dot product is taken with the row corresponding to action a of the feature matrix.
- Training is done with gradient descent, as in any other linear approximation of Q.

Results and Observations:

In all tasks, weights are initialized with zeros and random seed is set to 0.

Task 1:

Final training plot Task 1 - with epsilon = 0.02, learning rate = 0.075:

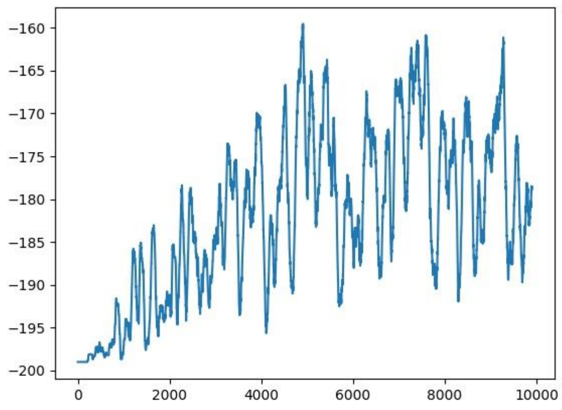


Final reward on testing: -134.57

These are the final parameter values. The process to arrive at them is detailed below:

Initially, an epsilon of 0.2 and learning rate of 0.1 were tried:

Training plot for Task 1 – with epsilon = 0.2, learning rate = 0.1:

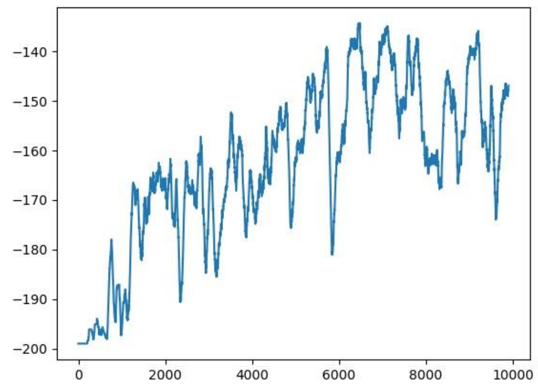


Final reward on testing: -145.57.

The reward in the training plot seems to converge at a point lower than -160, but on testing it gives a reward higher than -160. Additionally, improvement across iterations in the training plot is not uniform, with the plot appearing to have many ups and downs. When epsilon or learning rate were reduced, it resulted in a test reward less than -145.57, although the training plots looked much better and converged at a point above -160.

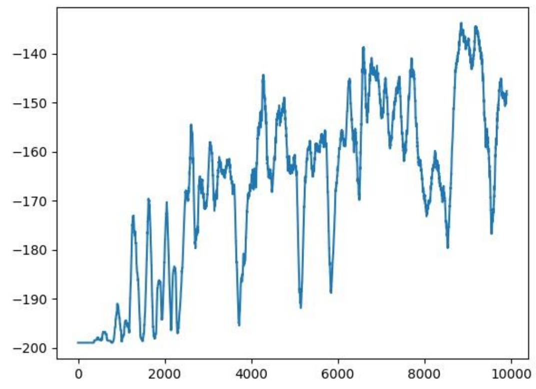
Training plots and test rewards with varying values of epsilon and learning rate:

Epsilon = 0.05, learning rate = 0.1:



Test reward = -162.07. This is not an acceptable test result, but the training plot was improved.

Subsequently, epsilon = 0.02, learning rate = 0.05 was tried:

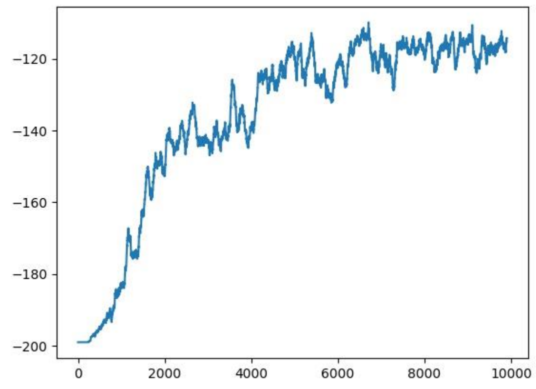


Test reward = -148.55

Now the test reward as well as the training plot were acceptable as epsilon was reduced. With additional trials with epsilon in the range 0.01-0.02 and learning rate in the range of 0.05-0.08, the final parameters were found. Evidently, an epsilon value of 0.05 or greater resulted in more exploration than was necessary to learn across a weight vector of this size.

Task 2:

Final training plot Task 2 - with epsilon = 0.05, learning rate = 0.02:

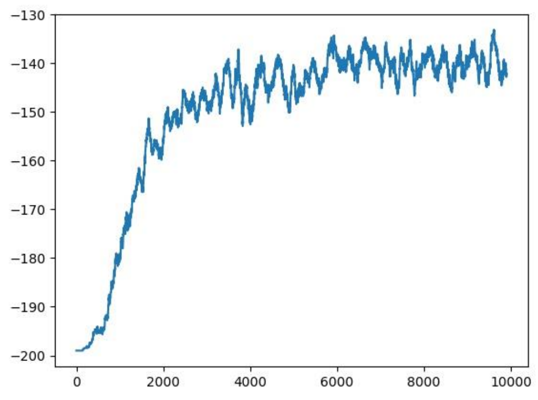


Final reward on testing: -101.35

These are the final parameter values. The process to arrive at them is detailed below:

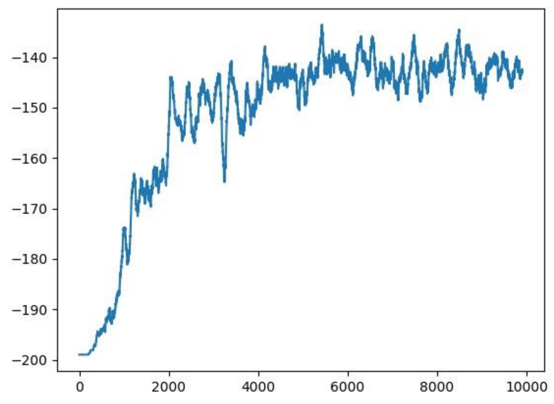
With 5 tilings, the training took over 20 minutes to complete, so the number of tilings were changed to 4, which brought down the training time without compromising on test performance to a great degree.

The following training plot was obtained with 5 tilings, learning rate of 0.02 and epsilon of 0.2:



While the test reward was above -130, the training plot converged at a point which was around -140. In an effort to improve this and the training time, more adjustments were made:

4 tilings, learning rate = 0.025, epsilon = 0.2:



Making changes in the number of tilings (3 tilings per action, and then 2 tilings per action were tried) and learning rate was not improving the look of the training plot. So, epsilon was reduced from 0.2 to 0.05. This resulted in a training plot which converged to a much higher reward, but the plot also stagnated, or became flat, towards the end of the 10000 iterations. So, learning rate was reduced from 0.025 to 0.02 in order to bring about continual improvement. This resulted in the final plot shown at the beginning, and a test reward of -101.35 which was the highest obtained thus far.

The experiment in Task 2 benefitted from a smaller learning rate than that of Task 1, and a larger epsilon – perhaps more exploration was needed in order to learn across the increased number of weights. Tile coding results in much faster and powerful learning even though the tile widths are the same as the interval widths of the discretization in Task 1. The Task 2 training plots converge at much higher values of reward and training also results in more uniform improvement across the 10000 iterations, when compared to Task 1. Task 2 does take more time for training however, taking around 12 minutes while Task 1 takes around 4 minutes.