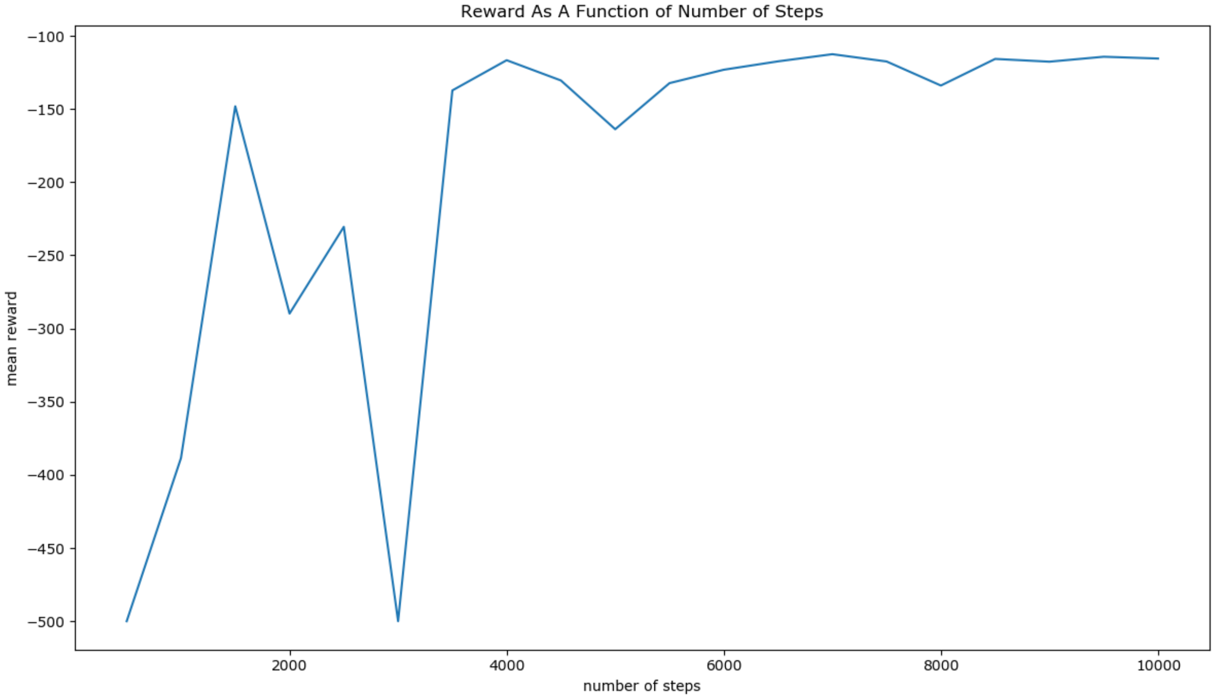
Reinforcement Learning Assignment 3 Report

Students: Itay Osovlanski (311129274)
Tomer Laor (206783813)

The code is solving the Mountain Car issue using Sarsa Lambda and function approximation. The features are 2D radial basis functions.
It runs a simulation using a pre-computed weights, then it computes weights for the current run and outputs a plot of the reward as function on the steps.

Important Note about our features:
We started from "logical" features – but the model didn't converge well.
We've implemented a random search over the possible range of centers for our features.
The features that we are using came out the best from our random search.

Main methods:
sarsa_lambda: This function returns W, the best weights that it found, and policy evaluations along checkpoints.
For each episode we will initialize E, go to the initial state, and pick a first action. Then, for every step we will execute the action, and select a new action (that will be executed in the next step) using epsilon greedy policy. Then we will compute the delta of the error and update W accordingly. We will also update E because we have done a step. If the environment told us that we got to a terminal state, we will finish this episode and go to the next one. At every checkpoint we will call policy_eval function to evaluate the policy.
eps_greedy_policy: generates a random number. If it's below epsilon, do a random action. Else, do the best action given Q.
policy_eval: Evaluate the policy greedily and not epsilon greedily. This is very important – because we do not want to explore new actions while evaluating. The evaluation is the mean of the rewards over 100 runs.
get_features: Returns the vector of the value of features given a state. phi[i] is the value of the i-th feature given a state. Implemented as suggested in Moodle.


Reward As A Function of Number of Steps
mean reward
number of steps