Reinforcement Learning Assignment 2 Report

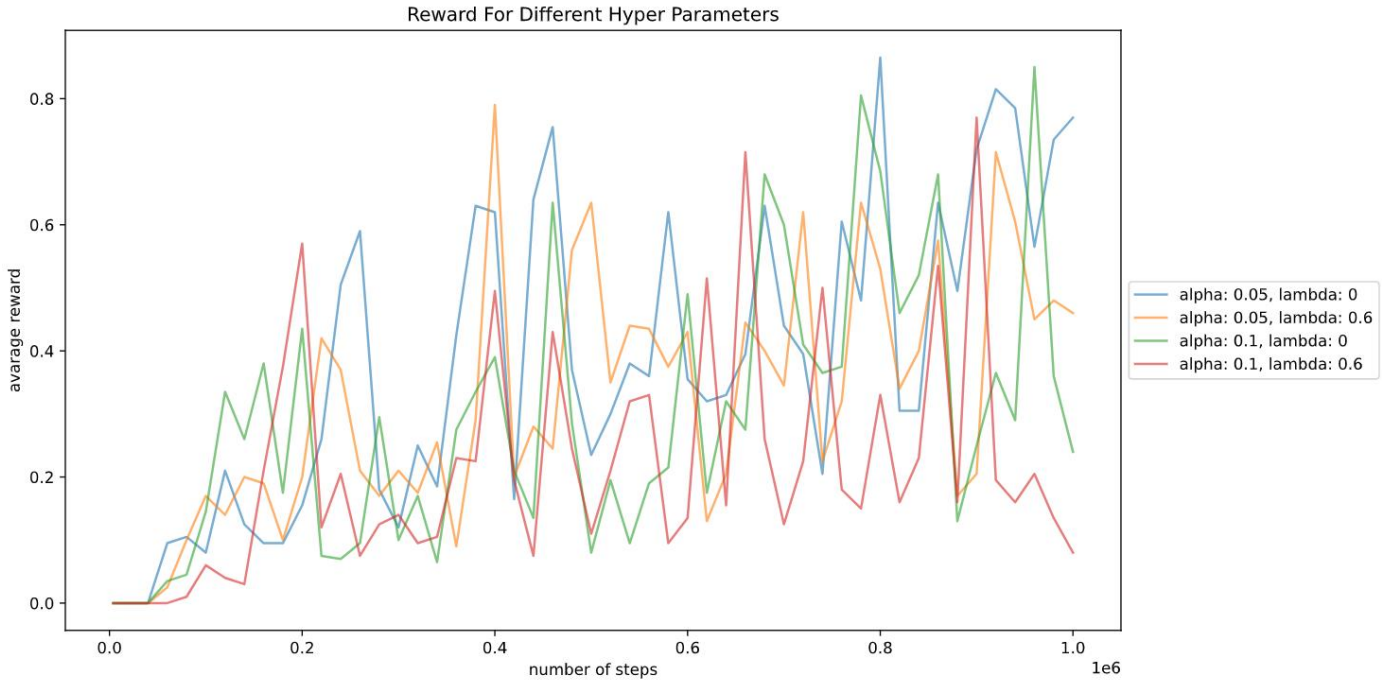
Students: Itay Osovlanski (311129274)
Tomer Laor (206783813)

The code is solving the frozen lake issue using Sarsa Lambda for 2 values of lambda and 2 values of alpha (4 combinations in total). After it computes a policy of hyper parameter combination it runs an agent in the environment for 1 episode under the policy and for each step prints the states of the run. After all the runs have finished, a plot is shown.

Main methods:

sarsa_lambda: This function returns Q, the best policy 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 Q 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. It also decays epsilon in order to get better convergence speed.
At every checkpoint (each 4K steps for under 20K steps, each 20k steps else) 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 200 runs. This method can compute the reward of an episode with the formula:
sum_of_rewards * (gamma ^ number_of_steps_in_run).
It can also compute the reward of an episode with the formula: sum_of_rewards.
It depends on the parameter with_discount.
Plots using 2 methods will be shown. We've done this because we didn't understand if we are required to do the first formula described or just sum the rewards.
show_sim_in_env: Shows the state of the environment for every action that an agent takes under the given policy in the given environment.

The required plot for summing the rewards (without discount the reward):
(This is the default behavior of our code)



The required plot for computing rewards using the formula:
sum_of_rewards * (gamma ^ number_of_steps_in_run)
(To evaluate using this formula, change the parameter EVAL_WITH_DISCOUNT on line 16 to True.