This assignment consisted of implementing and comparing the Sarsa(λ) and the Q-learning algorithm for a reinforcement learning problem in which our agent, which is placed in a grid world environment tries to reach a goal state by taking actions (left, right, up or down) starting from a given start state. The objective of the agent is to maximize its expected cumulative reward. Please refer to the file 'agent.py' for the implementation of the algorithms. The experiments are run on two MDP instances, 0 and 1 and the value of γ is set to 1.0 throughout all the experiments.

For Sarsa(λ), the accumulating traces tend to perform better and therefore, is used for all the experiments. The learning rate (α) is initialized to 0.15 and is decayed exponentially at every episode by a factor of 0.999 i.e. α at episode t is given by $0.15 * (0.999)^t$. Sarsa(λ) is an on-policy algorithm and the policy followed during the learning of the Q-function is given by Greedy(ϵ_t) where ϵ_t is given by $0.5 * (0.95)^t$. The optimal value of λ for the two MDP instances can be inferred from figures 1 and 2. For each value of λ, the Sarsa(λ) algorithm is run 50 times with different random seeds and the average results are plotted here.

I **observed** that increasing λ in the Sarsa algorithm leads to faster convergence until the optimal value of λ, after which the either the convergence is much slower or there is no convergence at all! I also noticed that the optimal value of λ is dependent on the hyperparameter setting we use!

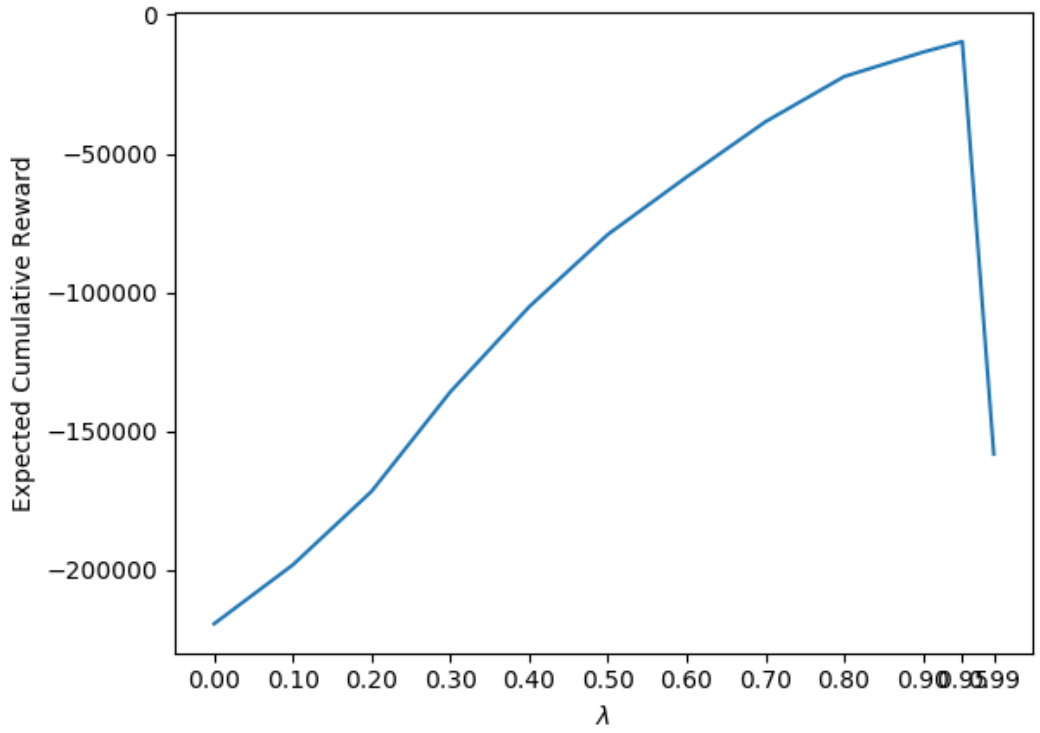

FIGURE 1. Expected cumulative reward over the first 500 episodes of training for Sarsa(λ) for MDP instance 0. Here the **optimal value** of λ is **0.95**.

Q-learning is an off-policy method where the exploration policy used in the experiments performed is given by Greedy(ϵ_t) w.r.t to the Q-function values learned so far. Here, $\alpha = 0.2$ and $\epsilon_t = 0.2 * (0.99)^t$. Also, for each MDP instance, 50 runs of Q-learning were performed, each run comprising of 1600 episodes and the average reward of each episode is plotted vs the episode number.

It can be observed from figures 3 and 4, that both Q-learning and Sarsa(λ) **converge to nearly optimal policies in a small number of episodes.** Also, one interesting thing to note is that the **Sarsa(λ) converges much faster** than the Q-learning algorithm (both are highly tuned).

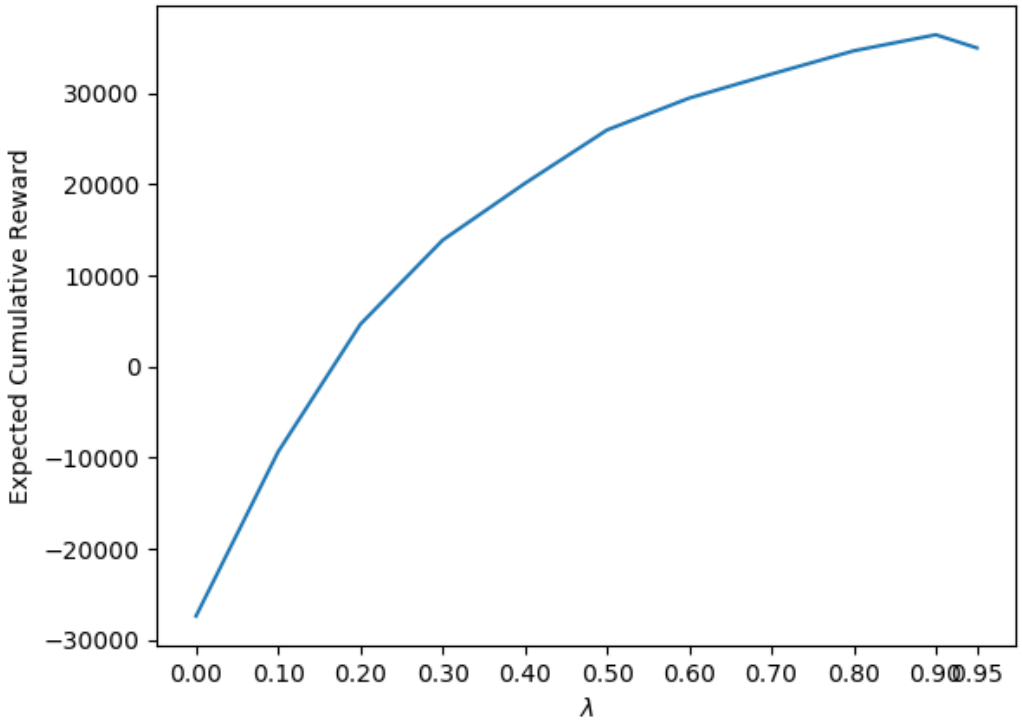

FIGURE 2. Expected cumulative reward over the first 500 episodes of training for Sarsa(λ) for MDP instance 1. Here the optimal value of λ is **0.8**.

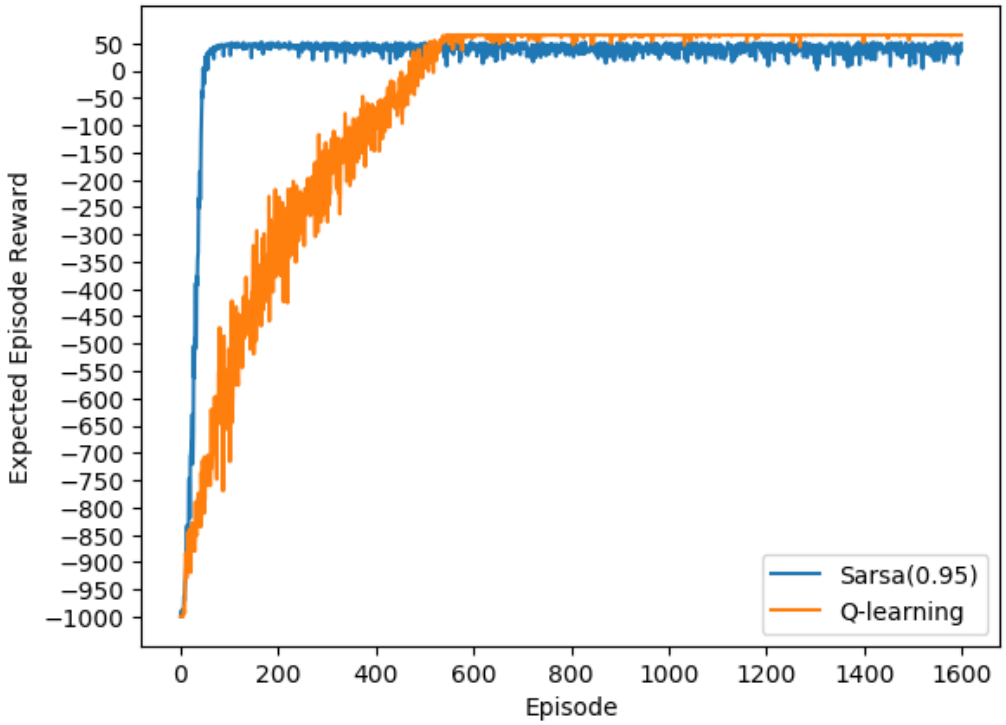

FIGURE 3. Expected episode reward versus episode number for instance 0. Here the optimal value of λ is 0.95 for Sarsa(λ).

All the experiments were conducted using the script 'startexperiment.sh' utilizing the 'start-client.sh' file in the client directory, takes three parameters: the MDP instance, the value of λ and the algorithm to be used. The plots are plotted using the script 'read_results.py'. Both of these scripts are present in the 'results+scripts' directory. The hyperparameter and annealing is found out by manual tuning and trying out different random settings of those hyperparameters.

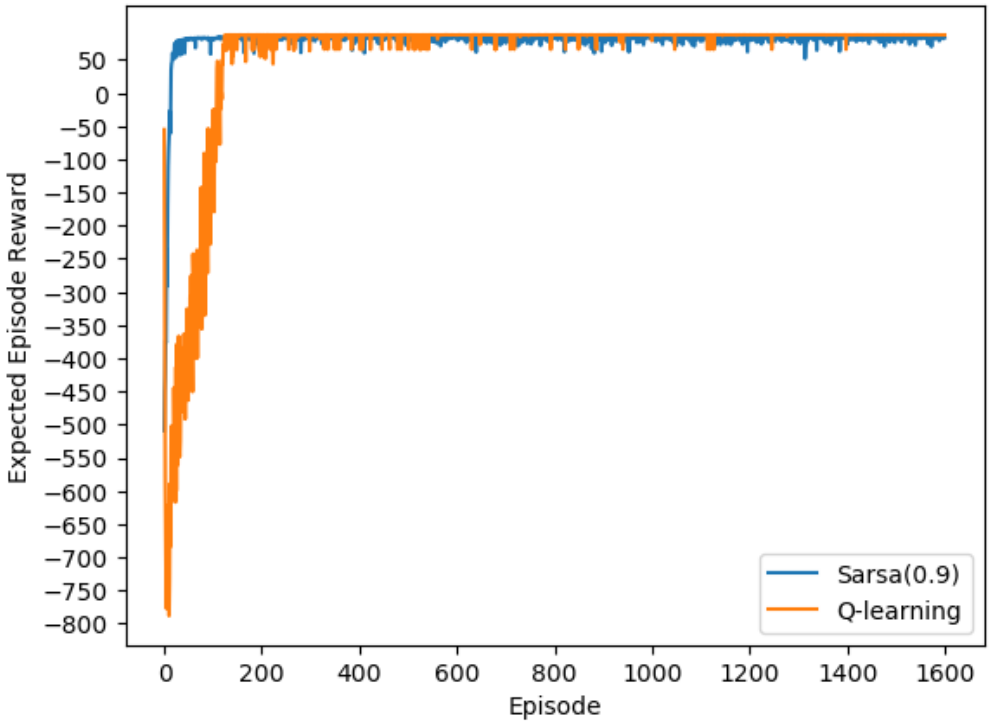

FIGURE 4. Expected episode reward versus episode number for instance 1. The optimal value of λ is 0.9 for Sarsa(λ) for this instance.