COMP767 - Assignment 2

Alexis Tremblay (260922703)* and Joseph D. Viviano (260878678)*

*University of Montreal

March 2019, 10th

1 Prediction and Control: Part A [50 points]

1.1 Experiment Design

To compare the performance of SARSA, Expected SARSA, and Q-learning on the taxi-v2 task, we used a constant discount factor $\gamma = 0.9$. Then, we ran 10 runs, 100 segments each, for all combinations of the 5 learning rates $\alpha = 0.1, 0.3, 0.5, 0.7, 0.9$ and three temperatures $\tau = 0.1, 1.0, 10.0$ tested. Each segment consistent of 10 episodes of of training followed by a single test run where the agent follows the greedy policy. During all greedy action selection, ties were broken randomly. Otherwise exploration was softmax with temperature.

1.2 Hyperparameter Selection

For the following plots, we present the mean return collected during the final segment for all 10 runs. We see a few main trends across all settings. First, training performance was better overall than test performance. High temperatures reduce both training and test performance (due to too much exploitation), which is amplified by high learning rates. For lower temperatures (1 and 0.1), little difference was observed for the higher learning rates. Generally, high learning rates and lower temperatures led to the best results. We believe this is because the environment is deterministic and stationary, therefore, it makes sense for the agent to use all of the experience it receives at every step for the next action selected. High temperatures encourage exploration away from the learned policy, leading to suboptimal results. Similarly, too-low learning rates make suboptimal use of the runs during the segments, which should be fully-considered since no aspect of the environment meaningfully changes episode to episode. When the learning rate is too low, none of the method at any temperature learn to complete the task reliably (during the 10 runs, they fail to complete the task at least once).

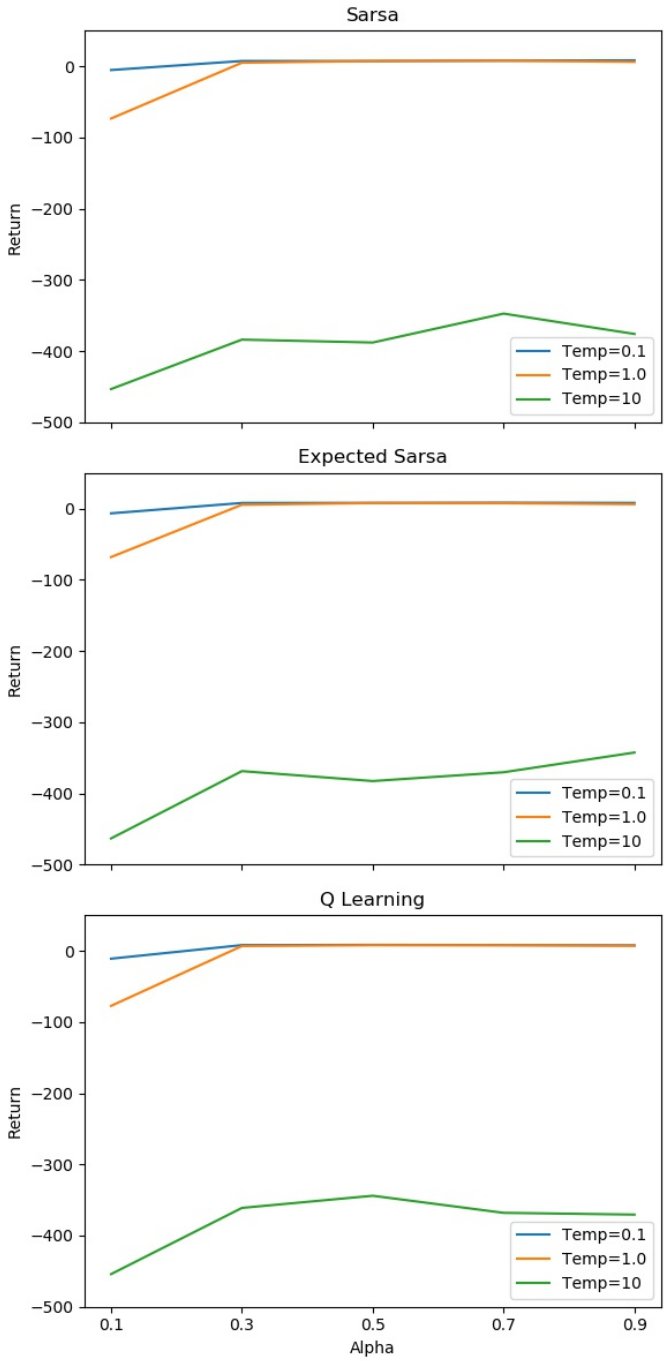

Figure 1: Mean final segment return for all hyperparameters across 10 runs, for all three algorithms.

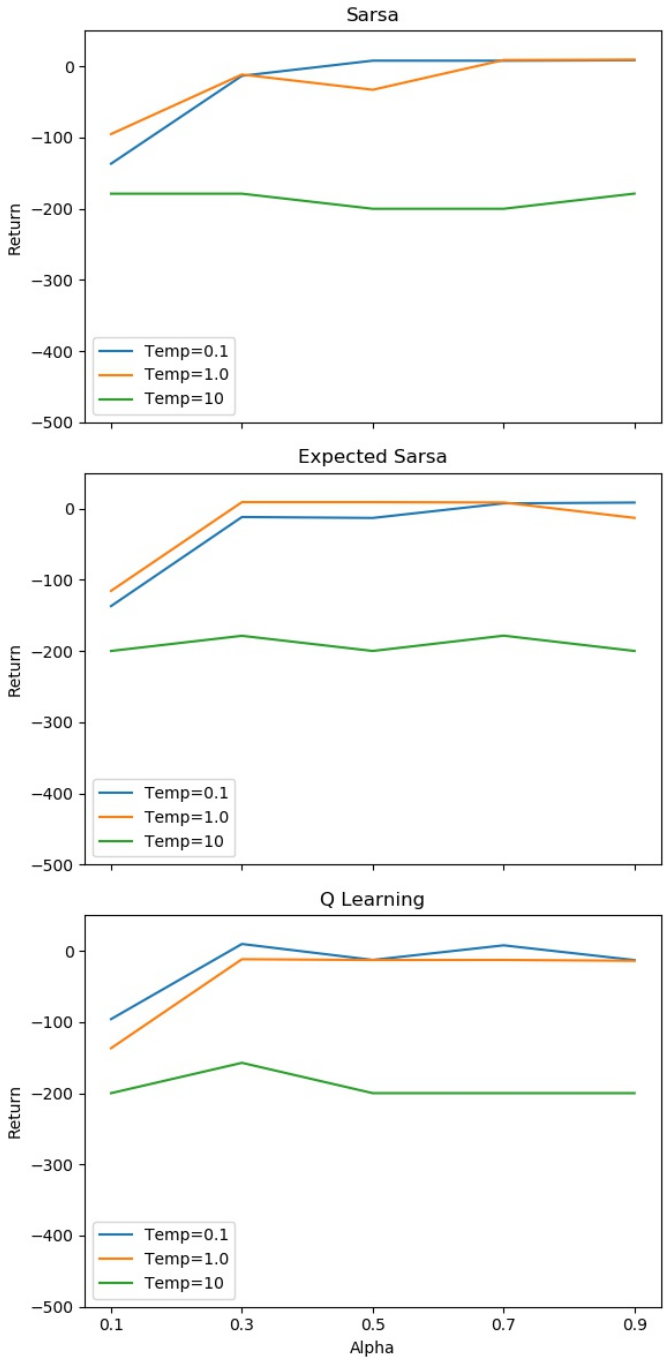

Figure 2: Mean final segment return for all hyperparameters across 10 runs, following the greedy policy, for all three algorithms.

1.3 Learning Curves

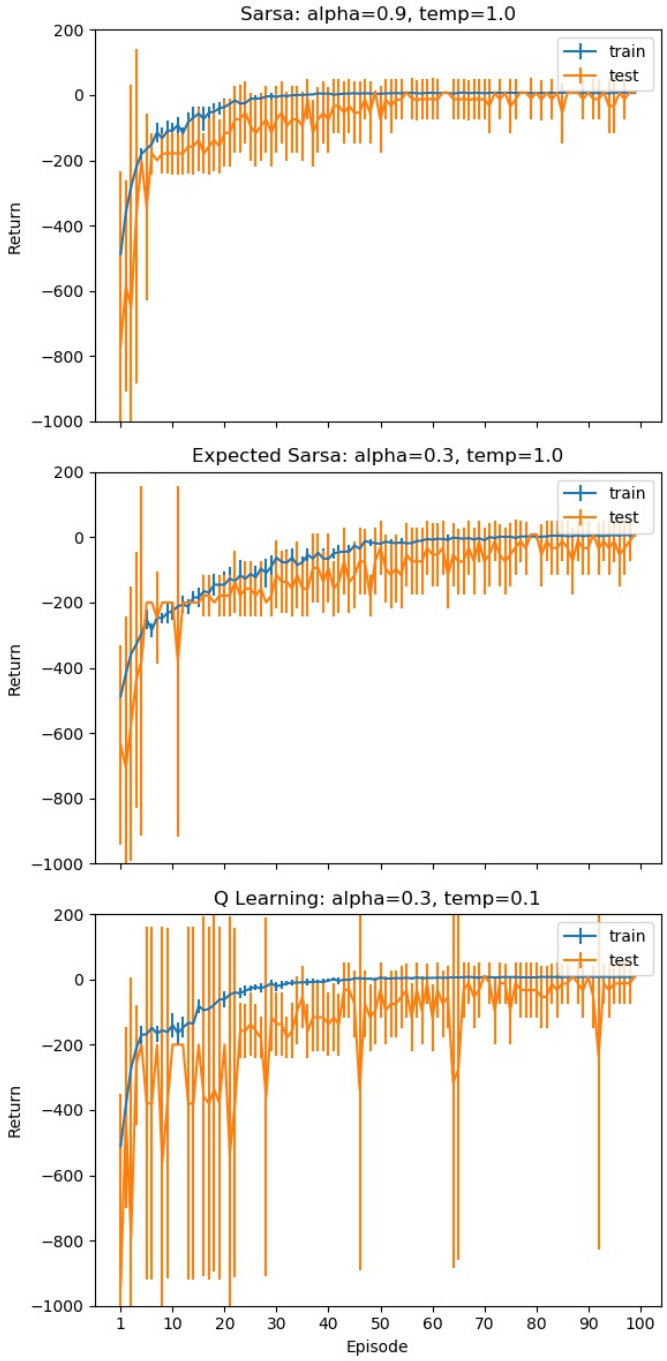

Figure 3: Mean return (±1 standard deviation) for each episode across all 10 runs for all three algorithms.

2 Function Approximation: Part A [50 points]

2.1 Experiment Design

Here, we implemented semi-gradient TD(λ) for estimating value functions to test the effect of both the trace decay rate $\lambda = 0, 0.3, 0.7, 0.9, 1$ and the learning rate $\alpha = 1/4, 1/8, 1/16$. Again γ was fixed at 0.9. We ran 10 runs, each consisting of 200 episodes, 200 steps each. The environment reset, forced into state $(0, 0)$ (i.e., angle=0, velocity=0) at the beginning of each episode, and for all experiments in each run, the seed for numpy's random number generator was fixed to the same value. Each run had a different seed. Actions were sampled from a fixed random policy whereby a torque $t \in [0, 2]$ was uniformly sampled, and applied in the same direction as the current velocity with 90% probability, otherwise it was applied in the opposite direction[1]. At the end of each episode, we recorded the value of state $(0, 0)$, and plot the mean of these values across the 10 runs.

We used function approximation with 5 tilings, each 10×10, each dimension representing angle and velocity respectively. [2]

The results are shown below (error bars show $\pm$ 1 standard deviation). A few things are apparent. First, small λ values take much longer to converge to the value estimate at a given learning rate. Second, higher learning rates lead to faster convergence. All algorithms, given enough iterations, converge to the value -20 for the state $(0, 0)$, but low values of λ with lower learning rates barely converge to this value after 200 episodes. Third, lower values of λ appear to be more susceptible to the specific state of the random number generator (see the bumps around run 100 in the lower λs). This could be seen as the algorithm being more susceptible to the bias of the first few starting states of the episode. In contrast, high values of λ show more variance in the estimate of the value at $(0, 0)$ from run to run, but less dependence on the random number generator or initial few starting states. In the end, the bias of low λ values does seem to diminish and all algorithms seem to converge to the same value estimate of state $(0, 0)$.

Conceptually a λ of 0 is like having no long term memory, which are less able to correct from any bias introduced at the start of an episode (i.e, the random number generator). In constrast, a λ value of 1 is a perfect memory of the past, good and bad. This makes it easier for the algorithm to ignore bias introduced by external forces, but also means that the algorithm is more likely to remember idosyncratic features of past training data, leading to higher variance in the value estimates. These plots suggest that an intermediate solution is desirable to balance the two trade-offs.

[1] So if the velocity was negative, the sampled torque is multiplied by -1. Otherwise multiplied by 1, overall giving a torque sampling $t \in [-2, 2]$

[2] We used the tiling code provided by Sutton at `http://incompleteideas.net/tiles/tiles3.html`

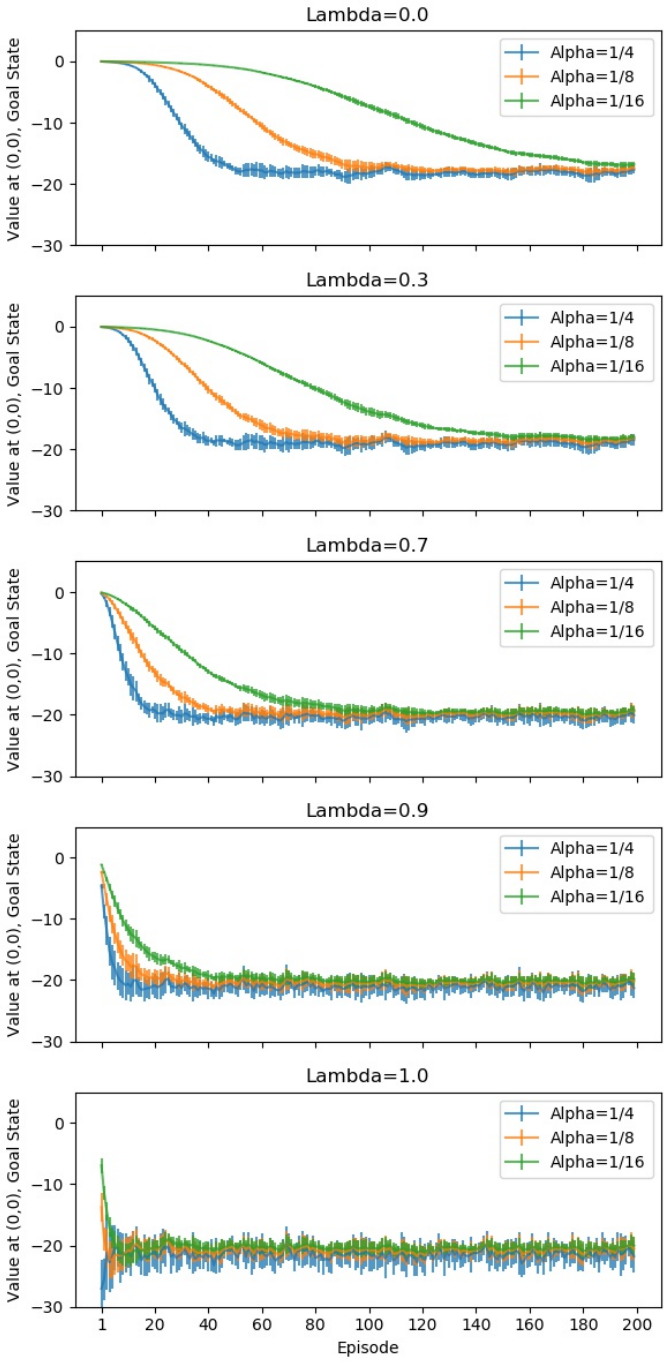

Figure 4: Mean value of position (0, 0) in the state space at the end of each episode across the 10 runs (± 1 standard deviation).