CMPUT 655 Assignment 5

Seth Akins

September 2024

1 TD Questions

1.1 Q-Function Updates

Iterative PI:

$$Q(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \sum_{a'} \pi(a'|s')Q(s',a')]$$

Monte Carlo:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha[G_t - Q(s_t,a_t)]$$

Q-Learning:

$$Q(S_t,A_t) = Q(S_t,A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1},a) - Q(S_t,A_t)]$$

SARSA:

$$Q(S_t,A_t) = Q(S_t,A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1},A_{t+1}) - Q(S_t,A_t)]$$

1.2 Advantages of TD

TD does not require the dynamics function like DP does, and has far lower variance due to the use of bootstrapping values compared with MC.

1.3 Variance

Due to the use of bootstrapping values, TD methods have much lower variance then MC methods, which rely on samples which can vary wildly in each episode.

1.4 Overestimation Bias

Overestimation bias is when certain values are overestimated because we take a maximum over our estimated values to estimate the actual maximum value. In Q-Learning, this occurs when we use the action in state s_{t+1} with the maximum value based on our current estimate of the Q function in the error term of the update.

Double Q-Learning works to prevent this because even if one of the Q values in Q1 is overestimated, we use the Q value in Q2 to update it, which is likely not overestimated. The two estimates are independent and unbiased from each other, so even Q2 is currently overestimating a state-action, we use the value in Q1 to update it, so it will not update based on an overestimate.

SARSA also suffers from overestimation because it typically uses an ϵ-greedy policy, which still involves assigning the maximum chance of picking an action to the highest valued one when doing the update.

1.5 Backup Diagrams

Top Left: Bellman Equation for Q
Bottom Left: Bellman Optimality Equation for Q
Top Right: SARSA
Bottom Right: Q-Learning

2 Exercise 6.1

$$\begin{aligned}
G_t - V(S_t) &= R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1}) \\
&= \delta_t + \gamma(G_{t+1} - V(S_{t+1}) + \Delta_{t+1}) \\
&= \delta_t + \gamma\Delta_{t+1} + \gamma\delta_{t+1} + \gamma^2(G_{t+2} - V(S_{t+2}) + \Delta_{t+2}) \\
&= \delta_t + \gamma\Delta_{t+1} + \gamma\delta_{t+1} + \gamma^2\Delta_{t+2} + \gamma^2\delta_{t+2} + \cdots + \gamma^{T-t-1}\delta_{T-1} + \gamma^{T-t}(G_T - V(S_T) + \Delta_T) \\
&= \delta_t + \gamma\Delta_{t+1} + \gamma\delta_{t+1} + \gamma^2\Delta_{t+2} + \gamma^2\delta_{t+2} + \cdots + \gamma^{T-t-1}\delta_{T-1} + \gamma^{T-t}(0 - 0 + 0) \\
&= \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k + \sum_{k=t}^{T-2} \gamma^{k-t+1}\Delta_{k+1}.
\end{aligned}$$

3 Exercise 6.12

No, Q-Learning is not the same as SARSA if action selection is greedy. This is because even though the update would be the same, SARSA uses the action it chooses in the update as the next action to take in state S', whereas Q-learning chooses a new action greedily after the update.

4 Coding Part 1

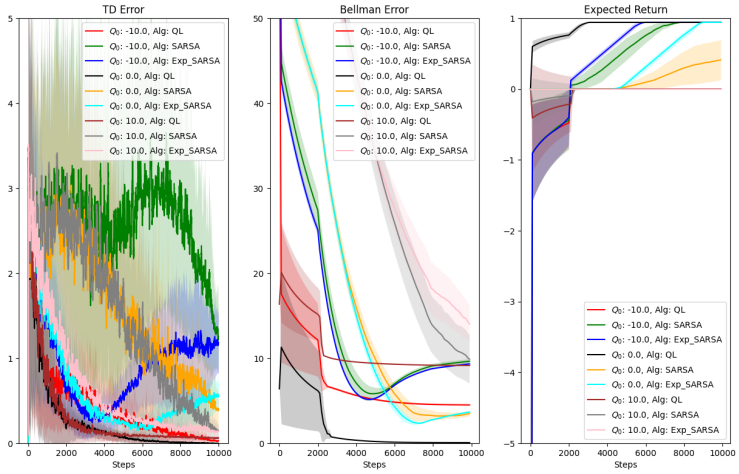

Figure 1: Part 1 Results

For all algorithms, starting with an initial value of zero is better, and leads all algorithms to learn a policy with positive expected returns; however, starting with a value of -10 allows both versions of SARSA to converge quicker, but with higher bellman error.

There is a difference between the TD Error and Bellman Error plots, as they measure different things. This is because the TD error is measuring the difference between our old estimate of the state-action pair with the new one, whereas the Bellman error is the difference between the optimal Q function based on our policy and our current estimate of the Q function.

For some of the algorithms, the expected return converges to near one, which would indicate the algorithm has learned the optimal policy; however, there does not appear to be a relationship between low TD and Bellman errors and converging to the optimal policy. SARSA has quite high TD error and plateaus at a higher bellman error, but is able to learn the a more optimal policy then Q-learning at $Q_0 = -10$, even though it has both lowest TD error and bellman error.

The policy can act optimally, even though it has not obtained the optimal Q-function, as evident by the runs of SARSA, which converge to the optimal expected return, but not to zero Bellman error. This is because as long as for each state, the optimal state-action value has the highest estimate, even if only by a small margin, our greedy policy based on our Q function estimate will

always choose that action.

Expected SARSA always performs at least as good, and sometimes better than either of the other two algorithms, regardless of initialization. No algorithm manages to achieve a non-zero reward for $Q_0 = 10$, but expected SARSA reaches the optimal policy for both other initial values.

To define performance, I would rely on the expected return, because it represents how well we expected the agent to do in the task we give it based on the greedy policy from our estimated action-value function. It gives us the best idea if the agent is learning a good policy, which we cannot tell from the TD error or the Bellman error.

5 Coding Part 2

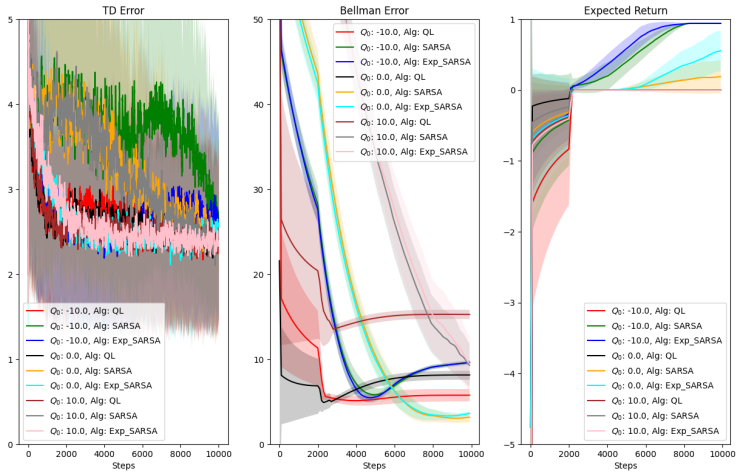

Figure 2: Part 2 Results

The plot for the TD error is far higher and more varied, and far fewer initializations converge to the optimal policy compared to without stochastic errors. The ones that do also take much longer, although their Bellman error trend is changed less. No instances of Q-Learning converge to the optimal policy with stochastic rewards, likely because they keep greedily moving in whatever direction they happen to receive the highest reward for.

6 Coding Part 3

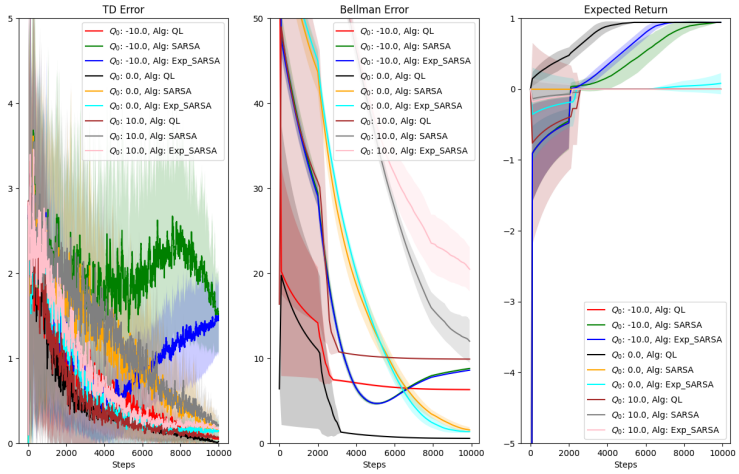

Figure 3: Part 3 Results

Compared to part 1, both the TD errors and Bellman errors are generally lower for those which converge to the optimal policy. This means that the Q function estimate they learn is closer to the actual true value for their policy, and that each update has less variance; however, it takes much longer to converge, with both expected SARSA and SARSA not converging in 10,000 steps for $Q_0 = 0$.