CMPUT 655 Assignment 2

Seth Akins

September 2024

1 Bellman Equation

The Bellman Equation for the value function, $v_\pi(s)$ is as follows:

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

The Bellman Equation for the action-value function, $q_\pi(s, a)$ is as follows:

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

1.1 Proof of the Bellman Equation for the Value Function

We start with the definition of the value function, which is defined as the expected return for starting in state s and following the policy π:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s]$$

From there, we can expand G_t to $R_{t+1} + \gamma G_{t+1}$ because G_t is the sum of our discounted return or discounted rewards, so we are just pulling out the first reward from the sum:

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s]$$

By the definition of an MDP, we have a finite set of possible values for our random variables for actions, states, and rewards; therefore, by the definition of expectation, we can convert the outer expectation into a sum of the potential outcomes, weighted by the probabilities of each one occurring, given we are in state s and following policy π. By definition, the expected value of G_{t+1} depends on the state s' and all future rewards, so we still need an expectation on it. As such, our equation becomes

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma \mathbb{E}_\pi[G_{t+1}|S_t = s']]$$

Finally, we can see that the term being multiplied by γ is the same term we started with, just at time $t+1$. This term is the expected return of state s' following policy π, which is the value function for $v(s')$ by definition. As such, we arrive at the Bellman Equation for the value function:

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

1.2 Difficulty of Estimating the Bellman Equations

The Bellman Equations are quite difficult to correctly estimate. The sum terms in each of them can become quite large as the set of possible states and actions increases, making the calculations quite difficult. The recursive nature also may involve a large number of calculations once the MDP become more complex, as many states and actions must be visited before arriving at a terminal state; however, it is still far easier then a brute force computation, which is why they are very important.

1.3 Requirements for Applying the Bellman Equations

In order to apply the Bellman Equations, we need to know the set of possible states we can be in, the set of possible actions we can take, the set of possible rewards, and the dynamics function of the MDP.

1.4 Computational Resource Assumptions

As mentioned in the difficulty of estimating the bellman equations, for complex problems the set of actions and states for an MDP can become quite large. In addition, larger problems will require substantial recursive calculations. As such, the amount of time and space required to compute the bellman equations can become large very quickly. It becomes nearly impossible in problems which have high branching factors, such as chess, as there are a large number of actions which can be taken from each state.

2 Exercise 3.11

By the definition of expectation:

$$\mathbb{E}_\pi[R_{t+1}|S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)r$$

3 Exercise 3.12

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

$$v_\pi = \sum_a \pi(a|s) q_\pi(a, s)$$

4 Exercise 3.13

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

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

5 Exercise 3.16

Adding a constant to the rewards of an episodic task could potentially have an effect. In an episodic task, we typically have $\gamma = 1$, which means our rewards are not discounted. If we have a maze running task, we want the agent to complete the maze as quick as possible. As such, we will likely give the agent no rewards, or negative rewards for each action taken, and only a positive reward when reaching the terminal state.

Imagine we gave the agent a reward of -1 for each action it takes and 5 for reaching the terminal state. If we add a constant $+1$ to each of these, the agent will no longer care about how long it takes to find the exit. That is because, with no discount, the total reward for reaching the terminal state at time $t = 10$ is the same as reaching it at $t = 1000$. Our agent will likely become lazy and not really be maze-running anymore.

An even worse scenario could occur if we added $+5$ to each of the rewards. Then, the agent would maximize the reward by walking around the maze for an infinite amount of time, and would never leave.

Clearly, adding a constant can drastically change the task. There could be a case where this doesn't happen, such as adding -1 to both of these tasks. That would keep the signs the same, so the agent would still prioritize reaching the end, as that is the only positive reward. As such, it depends on the relationship between the rewards we have, and the size of the constant we add.

6 Coding - Implementation

I have included the code in the zip of my submission. The link to the code in my GitHub repository is also located here.

7 Coding - Plotting

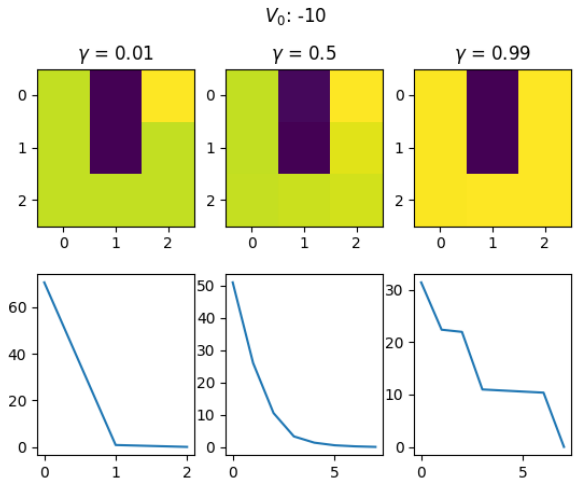

Figure 1: Plot of the Value Function with Initial Value of Minus 10

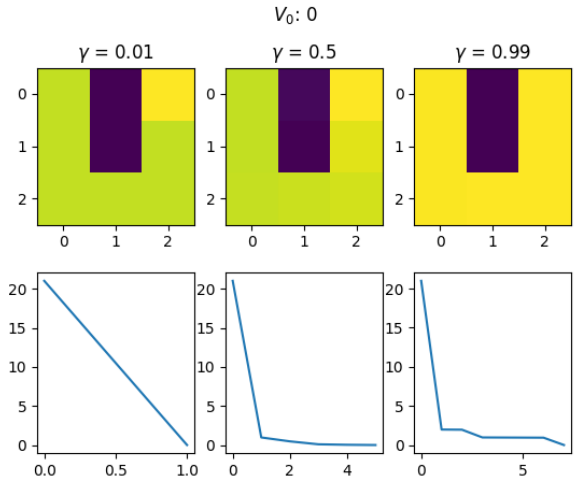

Figure 2: Plot of the Value Function with Initial Value of Zero

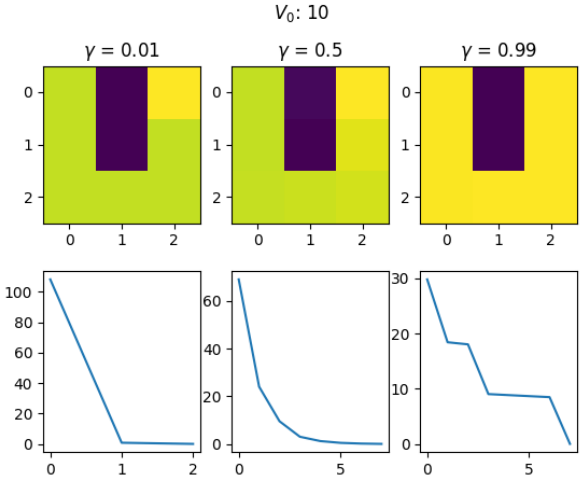

Figure 3: Plot of the Value Function with Initial Value of Ten

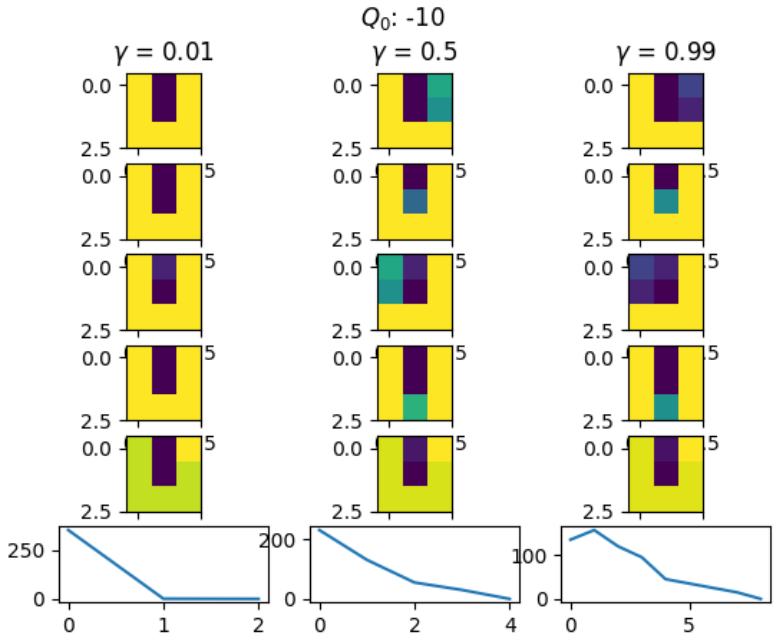

Figure 4: Plots for the Action Value Function with Initial Value of Minus Ten

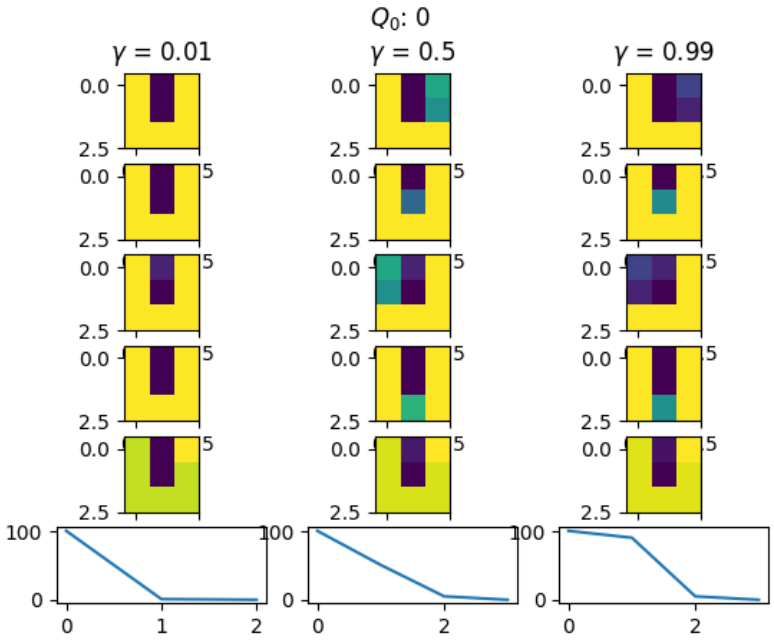

Figure 5: Plot for the Action Value Function with Initial Value of Zero

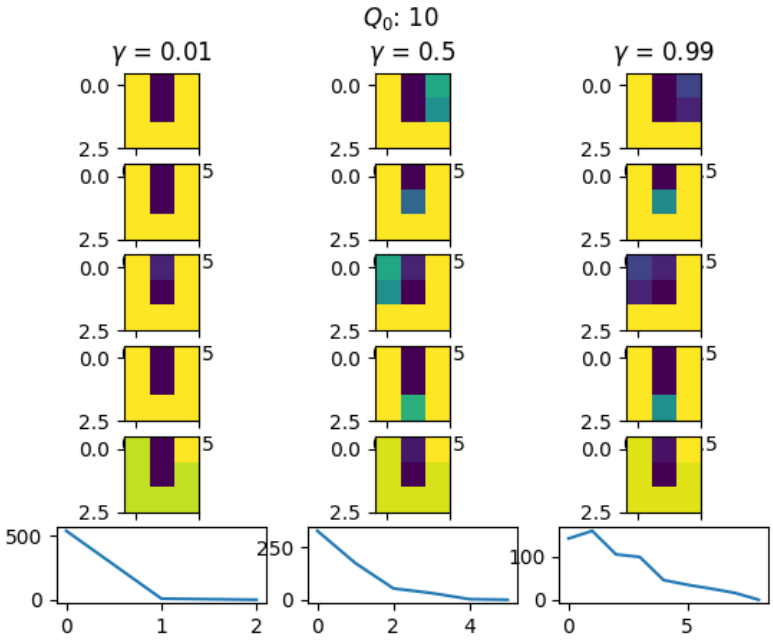

Figure 6: Plot for the Action Value Function with Initial Value of Ten

8 Coding - Discussion

8.1 Impact of Gamma

In the graphs, it can be seen that gamma heavily impacts what the final value of each state or state-action pair is. The closer the value of gamma is to one, the more of the future reward is preserved in the value of each state or action.

For example, on all three of the graphs for the value function, when gamma is very high, the value of states along the optimal path to the terminal state all have values close to 1, as the reward is not discounted much. Compared to when gamma is very low, the tiles, other than the tile with the terminal state, almost all have a value of 0 because nearly nothing of the final reward is measured in the value of the state.

This is also observed in the action-value function graphs, where if gamma is low, even when you make a bad move into the states worth -10, this is hardly indicated in the value of the state-action pair. Conversely, when gamma is high, and the agent moves into the states with reward -10, this is reflected substantially more in the value of the state-action pair.

Additionally, larger values of gamma seem to cause the rate of convergence to be slower, so more updates are required.

The gamma value could matter a lot for larger grid worlds, as if it is quite high, as the problem gets larger, it could take substantially longer to converge to our true value and action-value functions; however, if it is too small, then we would have such sparse valuations (mostly equal to 0) that the agent may struggle to make good choices.

8.2 Impact of Value Initialization

The value of initialization seems to have little to no effect on the value final value of the states and state-action pairs, which makes sense from the definition of the Bellman equations.

However, the initial value does cause a large difference in the values of the Bellman error. When the value was 0, the initial Bellman error was substantially smaller than for positive and negative 10. This caused the functions to converge to their true values quicker, as they are closer to their true values for most of the states and state-action pairs, so less steps are needed for convergence. Also, the reward for taking most actions in this MDP is zero, so by starting at zero, for most values we are much closer to what they will become after the first Bellman update.

For all the graphs, the slowest convergence occurred when gamma was 0.99 and the initial value was non-zero. This is because most of the initial values are far from their true value, and the only marginal discount on future rewards meant that more of the reward amount propagated to each state, requiring more iterations for convergence.

In general, as most states and state-action pairs were valued closer to zero than positive or negative 10, initializing the values to zero lead to faster convergence.

8.3 Bellman Error Trend

As mentioned above, both the initial value and the value of gamma heavily affect the bellman error trend. When gamma is small and the initial value was zero, most of the initial values are closer to the true values. As such, the bellman error quickly reaches a near zero value, with minimal changes before convergence. When gamma is large and the inital value was non-zero, the converse is observed for the same reason.

In general, most graphs exhibited a large initial drop in Bellman Error in the first iteration, and then much smaller drops in subsequent iterations (especially when $\gamma! = 0.99$). This makes sense as other than in the case when the initial value was zero, most states and state-action values are far from their true value as the reward for taking most actions in this MDP is zero.