CMPUT 655 Assignment 3

Seth Akins

September 2024

1 Theory Questions

1.1 Policy Evaluation

The iterative policy evaluation is as follows:

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

The update of the value function is expected because there are many possible future states and rewards as S_{t+1} is a random variable with a probability distribution for each possible next state occurring depending on the previous state.

Yes you can, still do the expected update, as you are just sampling values from the distribution, and for each step your backup diagram is just linear instead of branching into multiple different possible states. You would then just have to do take many samples in order for it to converge, which would be much more inefficient (assuming we know the dynamics).

In order to guarantee convergence, we need to assume that states, actions, and rewards are finite, the transition and reward function (the dynamics) must be stationary, and the discount factor must be less than one.

With the assumption that the dynamics are entirely known, we can solve the problem of policy evaluation as a system of linear equations instead of the iterative method, but this is more difficult because there number of unknown variables and equations is equal to the number of possible states.

Using the iterative method, we only need to maintain an estimate of the value of each state, instead of maintaining the same number of many lengthy equations for the value of each state. One disadvantage is that it may take more time for the function to converge to converge depending on the initial values, the size of the state, etc.

1.2 Exercise 4.3

1.2.1 Equation 4.3

$$q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$

1.2.2 Equation 4.4

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

1.2.3 Equation 4.5

$$q_{k+1}(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \sum_{a'} \pi(a'|s')q_k(a',s')]$$

1.3 Generalized Policy Iteration

Generalized Policy Iteration is the process of applying policy evaluation and policy improvement with any ordering of the two. Typically, we alternate between them with some set number of steps, so instead of alternating between converging to v_{pi} and our greedy π based on our value function, we take smaller steps between them, and arrive at both the optimal policy and value function, just like in policy iteration.

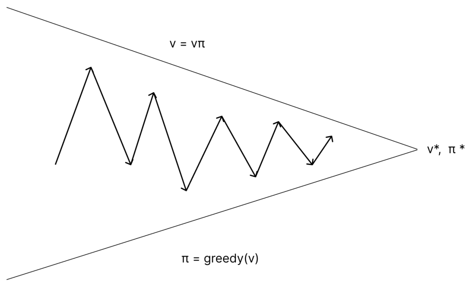

Figure 1: GPI Diagram

2 Coding Questions

2.1 Part 1

2.1.1 Implementation

All the code for this assignment is located here.

2.1.2 Results

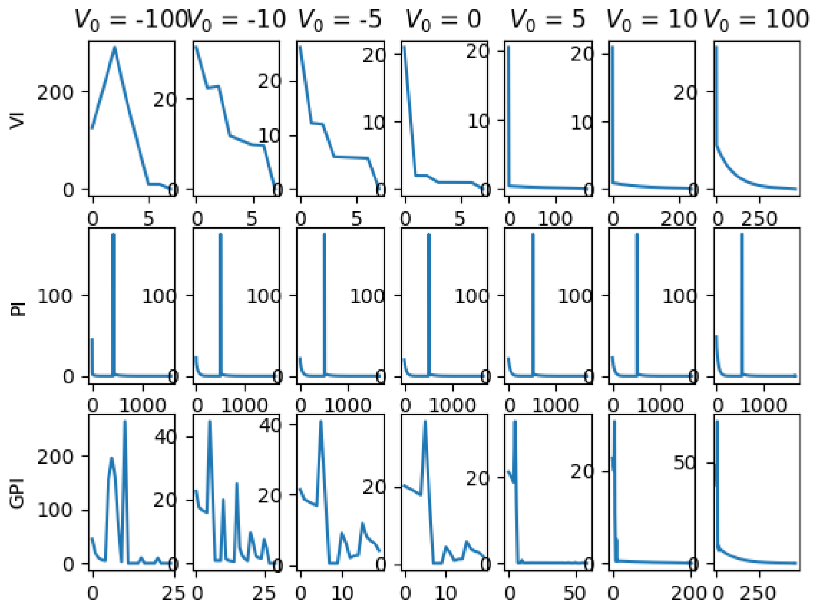

Figure 2: Bellman Error Plots for VI, PI, and GPI

	VI	PI	GPI
Mean	129	1647.43	118.57
S.D.	162.71	36.15	156.04

Table 1: Mean and Standard Deviation for Bellman Errors for Value Function

2.1.3 Discussion

On average, according to my table, general policy iteration was the quickest to learn the optimal policy.

The Bellman error does not steadily decrease, but spikes at some point in all of the graphs. This is because whenever we do a policy improvement step, the value function is no longer accurate for the new policy. As such, when we then do a policy evaluation step, there is a significant change in the value of the value function, which corresponds to the spike in Bellman errors we see in the graph.

For PI, the optimal policy and value function took significant time to converge, which is seen in the number of iterations shown on the graph and in the table, but was much more consistent with a far smaller standard deviation than the other two methods.

On average, GPI is better than the other algorithms in terms of converging quicker; however, the standard deviation is quite high, and for larger initial values of V_0, it takes significantly longer for them it converge then for small values of V_0. Value iteration also converges substantially quicker for non-positive values of V_0, GPI just does better in the worst cases, leading it to higher performance on average.

I would argue that this small grid world is not the best place to take advantage of GPI. Due to the simple optimal policy and small number of states, the advantage of taking smaller steps with GPI is not really worth it compared to the greedy approach used by value iteration. In a more complex MDP with a larger number of possible states and actions, the greedy approach by VI would likely be worse, as it could greedily head in a non-optimal direction and take significant time to head back towards an optimal policy. GPI could work better by taking more frequent, smaller steps in the correct direction for convergence.

2.2 Part 2

I was not able to finish the code for value improvement using the q function, but the other plots and data is here.

2.2.1 Plots and Table

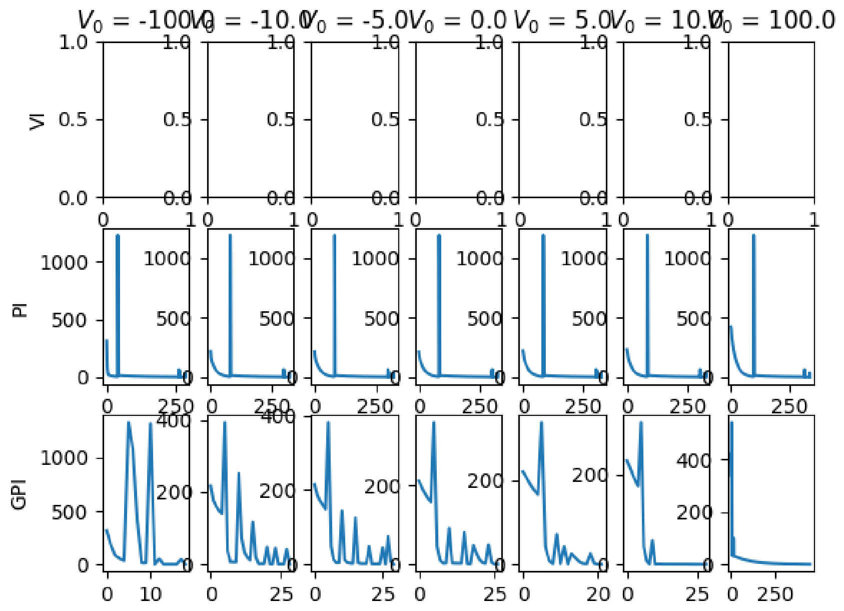

Figure 3: Bellman Error Plots for Q Function

	PI	GPI
Mean	321.29	86.86
S.D.	15.43	149.12

Table 2: Mean and Standard Deviation for Bellman Errors for Action-Value Function

2.2.2 Discussion

The number of policy evaluations is overall, much more efficient using the Q function instead of the V function. All of them take less steps to converge, and the mean convergence is lower for the algorithms.

The initialized value seems to matter less for q then for v. It still makes an impact, especially when we use $V_0 = 100$, but in general, the amount of steps to converge is smaller. The Bellman trends decrease quicker as a result of this.

2.3 Conclusion

2.3.1 Initialization Sensitivity

In general, PI is quite insensitive to value initialization, whereas VI and GPI are quite sensitive. In terms of number of policy evaluations, GPI is preferred on average, but clearly this can depend on the case and result in less performance depending on v_0. It tends to give us better worst case performance, but worse best case performance compared to VI. This makes sense due to the greedy approach of VI, compared to the more balanced approach of smaller steps with GPI.

2.3.2 Better Algorithm for Learning Q vs V

Both GPI and PI seem to do better when learning q compared to v, taking far less iterations on average, as seen in the lower mean values for the number of iterations (I would guess this is the same for VI, but I can't see).

2.3.3 Initialization: Does it Matter

The short answer is yes.

The longer answer is, as mentioned above, both GPI and VI are much more sensitive to the value of v_0 and q_0 than PI; however, they both still converge much quicker on average than PI.

Regardless of the algorithm and function we learn, positive values of v_0 and q_0 take longer to learn. This is likely because there is only one positive reward for one action in this MDP, and it only has a value of one. In order to converge the functions and policy to their optimal values at or below one, many steps are required when the values are initialized above one.

The value function seems to be in general more sensitive to initialization values then the action-value function. The action-value function is effected, but takes much fewer iterations to convergence, whereas the value function really struggles with large positive values of V_0.