CMPUT 655
Assignment 4

September 22, 2024

Dikshant
1877098

---

1. **Monte Carlo**

   (a) MC can be seen as a form of GPI, i.e., a loop of policy evaluations and policy improvements. In at most two sentences, what is the difference compared to Dynamic Programming?

   (b) Given a trajectory $\{s_t, a_t, r_t\}_{t=1...T}$ write the recursive loop to calculate the MC estimate of the value function for each state in the trajectory (hint: it's in the pseudocode).

   (c) In one sentence, say what it means that "MC does not bootstrap".

   (d) In one sentence, when is first-visit MC equal to every-visit MC? (hint: think about the blackjack example in the book)

   (e) In at most two sentences, define the two requirements for MC to converge. Are they practical?

   (f) List at least two drawbacks and one advantage of MC.

   (g) In at most two sentences, say when we need importance sampling.

Ans:

(a) GPI learns optimal behavior directly through interactions with the environment by estimating value functions from sample experiences, without requiring knowledge of the environment's dynamics. In contrast, DP methods rely on a complete model of the environment, including all transition probabilities and rewards, to perform updates.

(b)

$$G = R_{t+1} + \gamma G \quad (\text{for } t = T-1, T-2, \ldots, 0).$$

(c) MC doesn't bootstrap means they do not update their value estimates on the basis of the value estimates of successor states.

(d) First-visit MC equals every-visit MC when each state-action pair is visited only once per episode.

(e) For MC to converge, we need (1) sufficient exploration to ensure every state-action pair is visited infinitely often, and (2) the learning rate to decay over time (e.g., through decreasing step sizes). These conditions are often impractical, especially ensuring infinite exploration in real-world settings.

(f) Drawbacks:
• Delayed Feedback: MC methods only update estimates after an entire episode is completed. This can result in slower learning because it requires waiting until the end of an episode before making any updates.
• High Variance: MC methods can suffer from high variance in their estimates, especially in stochastic environments, as they rely on full-episode returns, which can vary significantly between episodes.
Advantage:
• Model-Free: MC methods do not require knowledge of the environment's dynamics (transition probabilities and rewards). This makes them suitable for tasks where the environment model is unknown or too complex to model.

(g) Importance sampling is needed when evaluating or improving a target policy using data generated from a different behavior policy. It corrects for the discrepancy between the two policies, ensuring unbiased estimates in off-policy learning.

2. *Consider an MDP with a single non-terminal state and a single action that transitions back to the nonterminal state with probability p and transitions to the terminal state with probability $1-p$. Let the reward be +1 on all transitions, and let γ=1. Suppose you observe one episode that lasts 10 steps, with a return of 10. What are the first-visit and every-visit estimators of the value of the non-terminal state?*

Ans: The first-visit MC estimate of the value of the non-terminal state is:

$$V_{\text{first-visit}}(s) = 10$$

The every-visit MC estimate is the average of all the returns:

$$V_{\text{every-visit}}(s) = \frac{10+9+8+7+6+5+4+3+2+1}{10} = \frac{55}{10} = 5.5$$

3. *In the boxed algorithm for off-policy MC control, you may have been expecting the W update to have involved the importance-sampling ratio $\pi(A_t|S_t)/b(A_t|S_t)$, but instead it involves $1/b(A_t|S_t)$. Why is this nevertheless correct?*

> **Ans:** Because our target policy, $\pi(S_t)$, is a deterministic, greedy one, we are only observing trajectories where $\pi(A_t|S_t) = 1$

4. *(a)* ***Part 1***
   Run on-policy MC. You'll have a total of 9 settings. For each setting, plot the mean of the Bellman error over steps with 95% confidence interval across the 50 seeds. Use `error_shade_plot()` *and the provided loop .How do plots compare to Part 1? Did IS help or not? Why? and the provided loop. Discuss your results:*

   - *Is there a trend among the hyperparameters (episodes per iteration, decay)?*
   - *What are the implications of collecting more episodes per iteration? Think about 1) accuracy of the estimates, 2) number of updates.*
   - *What are the implications of a faster decay? Is it always better to keep exploring for longer? Or is the problem easy enough that we decay faster and still learn the optimal policy? Does the number of episodes per iteration matter with respect to decay?*

   Be concise and answer each bullet point in 1-2 sentences.

   (b) ***Part 2***
   Repeat the experiment but this time have stochastic rewards. In your experiment loop, make the environment with the argument `reward_noise_std=3.0` *to add Gaussian noise with 0 mean and 3.0 standard deviation, and run it again. Note that the matrix used to compute the true Q-function is still correct, because it denotes the expectation over rewards (and the Gaussian noise has 0 mean). Discuss your results:*

   - *How do plots compare to Part 1? Is the trend different now? If yes, why?*
   - *Do you think that having a random initial state (rather than fixed in the top-left corner of the grid) would help? Would that be equivalent to "exploring starts"?*
   - *Stochastic rewards highlight a problem of MC estimates: which one?*

   Be concise and answer each bullet point in 1-2 sentences.

   (c) ***Part 3***
   Repeat both experiments but this time use weighted importance sampling (IS). As target policy, use an -greedy policy with = 0.01 (this will prevent the importance ratio from going to 0, since every action will have a small probability of being selected). Discuss your results:

- *How do plots compare to Part 1? Did IS help or not? Why?*

Be concise and answer each bullet point in 1-2 sentences.
Note: when computing the true Q-function for the Bellman error, remember that now your target policy is not the behavior policy!

Ans:

(a) **Part 1**

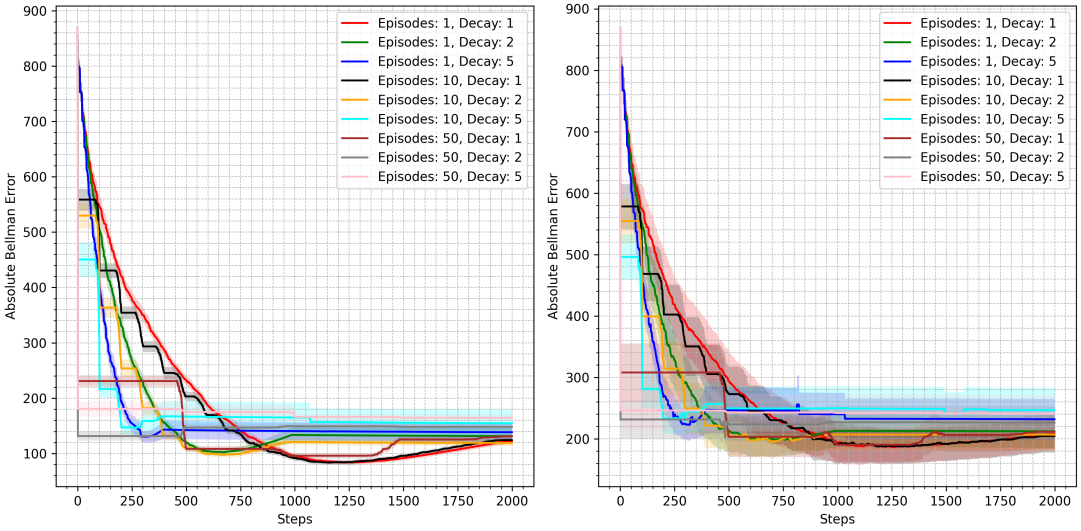

Figure 1: On-Policy Monte Carlo

- **Trend among the hyperparameters**:
  - Episodes per iteration: As the number of episodes per iteration increases, the Bellman error tends to reduce more rapidly. This is because accumulating more episodes before updating the Q-function provides a better estimate of the return for each state-action pair, improving the accuracy of the policy evaluation.
  - $\epsilon - decay$: A faster decay decreases exploration more quickly, meaning the policy will converge to a greedy policy faster.
- **Implications of collecting from episodes per iteration**:
  - Accuracy of the estimates: More episodes per iteration increase the accuracy of the estimates and it reduces the variance in the updates, leading to more quick learning.

- Number of updates: While collecting more episodes before updating the Q-function leads to more accurate estimates, it also reduces the frequency of updates.

• **Implication of faster ϵ decay**: Faster decay reduces exploration early on, which can speed up convergence but risks missing better policies if exploration is insufficient.
Is it always better to keep exploring for longer?: No, if the problem is simple, faster decay can still lead to the optimal policy. In more complex environments, longer exploration helps prevent suboptimal convergence.
Or is the problem easy enough that we decay ϵ faster and still learn the optimal policy?: In our case, the environment was simple so it's able to converge quickly.
Does the number of episodes per iteration matter with respect to decay: Yes, with more episodes per iteration, even faster decay works well as updates are based on better estimates. With fewer episodes, slower decay is needed to ensure sufficient exploration.

(b) **Part 2**

• **Comparison to Part 1**: The plots with stochastic rewards show more fluctuation in the Bellman error, and convergence is slightly slower.
• **Random Initial State**: Having a random initial state would encourage better exploration of the state space, helping to reduce bias introduced by starting in the top-left corner each time. This would be equivalent to "exploring starts," potentially improving policy evaluation by allowing the agent to experience a wider variety of states.
• **Problem of MC estimates**: If the rewards are noisy, MC estimates of the return can have high variance, which slows learning and increases the uncertainty in the value estimates.

(c) **Part 3**

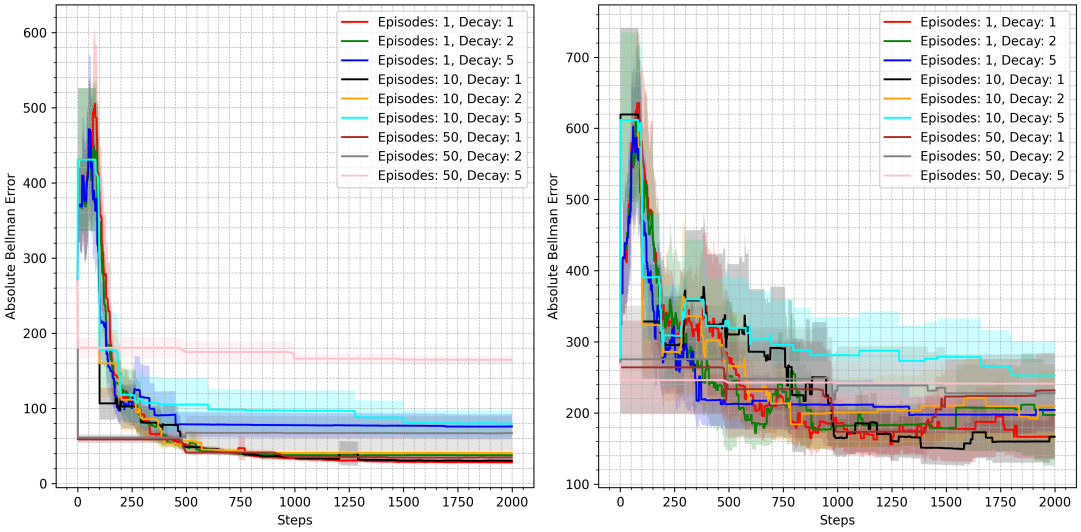

Figure 2: Off-Policy Monte Carlo (IS)

- **Advantages of importance sampling**: The plots with IS show more stable Bellman error trends compared to Part 1 without IS. IS helped because it reduces bias by adjusting the weight of returns based on the likelihood of selecting the same actions under both the behavior and target policies.
- **Extra finding**: Since the target policy (ϵ-greedy with ϵ=0.01) ensures non-zero probabilities for all actions, the variance of estimates is there, and it is particularly higher with higher noise.

Code can be accessed on github from here: Monte Carlo