Assignment 3: Q-Learning and Actor-Critic Algorithms

Due October 18, 11:59 pm

1 Multistep Q-Learning

Consider the N-step variant of Q-learning described in lecture. We learn $Q_{\phi_{k+1}}$ with the following updates:

$$y_{j,t} \leftarrow \left(\sum_{t'=t}^{t+N-1} \gamma^{t'-t} r_{j,t'} \right) + \gamma^N \max_{\mathbf{a}_{j,t+N}} Q_{\phi_k}\left(\mathbf{s}_{j,t+N}, \mathbf{a}_{j,t+N}\right) \tag{1}$$

$$\phi_{k+1} \leftarrow \arg\min_{\phi \in \Phi} \sum_{j,t} \left(y_{j,t} - Q_\phi(\mathbf{s}_{j,t}, \mathbf{a}_{j,t})\right)^2 \tag{2}$$

In these equations, j indicates an index in the replay buffer of trajectories $\mathcal{D}_k$. We first roll out a batch of B trajectories to update $\mathcal{D}_k$ and compute the target values in (1). We then fit $Q_{\phi_{k+1}}$ to these target values with (2). After estimating $Q_{\phi_{k+1}}$, we can then update the policy through an argmax:

$$\pi_{k+1}\left(\mathbf{a}_t \mid \mathbf{s}_t\right) \leftarrow \begin{cases} 1 \text{ if } \mathbf{a}_t = \arg\max_{\mathbf{a}_t} Q_{\phi_{k+1}}\left(\mathbf{s}_t, \mathbf{a}_t\right) \\ 0 \text{ otherwise.} \end{cases} \tag{3}$$

We repeat the steps in eqs. (1) to (3) K times to improve the policy. In this question, you will analyze some properties of this algorithm, which is summarized in Algorithm 1.

Algorithm 1 Multistep Q-Learning

Require: iterations K, batch size B
1: initialize random policy π_0, sample $\phi_0 \sim \Phi$
2: **for** $k = 0 \dots K-1$ **do**
3: Update $\mathcal{D}_{k+1}$ with B new rollouts from π_k
4: compute targets with (1)
5: $Q_{\phi_{k+1}} \leftarrow$ update with (2)
6: $\pi_{k+1} \leftarrow$ update with (3)
7: **end for**
8: **return** π_K

1.1 TD-Learning Bias (2 points)

We say an estimator $f_\mathcal{D}$ of f constructed using data $\mathcal{D}$ sampled from process P is *unbiased* when $\mathbb{E}_{\mathcal{D}\sim P}[f_\mathcal{D}(x) - f(x)] = 0$ at each x.

Assume $\hat{Q}$ is a noisy (but unbiased) estimate for Q. Is the Bellman backup $\mathcal{B}\hat{Q} = r(s,a) + \gamma \max_{a'} \hat{Q}(s', a')$ an unbiased estimate of $\mathcal{B}Q$?

☐ Yes

■ No

1.2 Tabular Learning (6 points total)

At each iteration of the algorithm above after the update from eq. (2), Q_{ϕ_k} can be viewed as an estimate of the true optimal Q^*. Consider the following statements:

I. $Q_{\phi_{k+1}}$ is an unbiased estimate of the Q function of the last policy, Q^{π_k}.

II. As $k \to \infty$ for some fixed B, Q_{ϕ_k} is an unbiased estimate of Q^*, i.e., $\lim_{k\to\infty} \mathbb{E}\left[Q_{\phi_k}(s,a) - Q^*(s,a)\right] = 0$.

Answer Key

III. In the limit of infinite iterations and data we recover the optimal Q^*, i.e., $\lim_{k,B\to\infty} \mathbb{E}\left[\|Q_{\phi_k} - Q^*\|_\infty\right] = 0$.

We make the additional assumptions:

- The state and action spaces are finite.
- Every batch contains at least one experience for each action taken in each state.
- In the tabular setting, Q_{ϕ_k} can express any function, i.e., $\{Q_{\phi_k} : \phi \in \Phi\} = \mathbb{R}^{S \times A}$.

When updating the buffer $\mathcal{D}_k$ with B new trajectories in line 3 of Algorithm 1, we say:

- When learning *on-policy*, $\mathcal{D}_k$ is set to contain only the set of B new rollouts of π (so $|\mathcal{D}_k| = B$). Thus, we only train on rollouts from the current policy.
- When learning *off-policy*, we use a fixed dataset $\mathcal{D}_k = \mathcal{D}$ of B trajectories from another policy π'.

Indicate which of the statements **I-III** always hold in the following cases. No justification is required.

	I.	**II.**	**III.**
1. $N = 1$ and ...			
(a) on-policy in tabular setting	☐	☐	■
(b) off-policy in tabular setting	☐	☐	■
2. $N > 1$ and ...			
(a) on-policy in tabular setting	☐	☐	■
(b) off-policy in tabular setting	☐	☐	☐
3. In the limit as $N \to \infty$ and ...			
(a) on-policy in tabular setting	■	■	■
(b) off-policy in tabular setting	☐	☐	☐

1.3 Variance of Q Estimate (2 points)

Which of the three cases ($N = 1$, $N > 1$, $N \to \infty$) would you expect to have the highest-variance estimate of Q for fixed dataset size B in the limit of infinite iterations k? Lowest-variance?

Highest variance:

☐ $N = 1$

☐ $N > 1$

■ $N \to \infty$

Lowest variance:

■ $N = 1$

☐ $N > 1$

☐ $N \to \infty$

1.4 Function Approximation (2 points)

Now say we want to represent Q via function approximation rather than with a tabular representation. Assume that for any deterministic policy π (including the optimal policy π^*), function approximation can represent the true Q^π exactly. Which of the following statements are true?

☐ When $N = 1$, $Q_{\phi_{k+1}}$ is an unbiased estimate of the Q-function of the last policy Q^{π_k}.

☐ When $N = 1$ and in the limit as $B \to \infty$, $k \to \infty$, Q_{ϕ_k} converges to Q^*.

☐ When $N > 1$ (but finite) and in the limit as $B \to \infty$, $k \to \infty$, Q_{ϕ_k} converges to Q^*.

■ When $N \to \infty$ and in the limit as $B \to \infty$, $k \to \infty$, Q_{ϕ_k} converges to Q^*.

Answer Key

1.5 Multistep Importance Sampling (5 points)

We can use importance sampling to make the N-step update work off-policy with trajectories drawn from an arbitrary policy. Rewrite (2) to correctly approximate a Q_{ϕ_k} that improves upon π when it is trained on data $\mathcal{D}$ consisting of B rollouts of some other policy $\pi'(\mathbf{a}_t \mid \mathbf{s}_t)$.

Do we need to change (2) when $N = 1$? What about as $N \to \infty$?

You may assume that π' always assigns positive mass to each action. [Hint: re-weight each term in the sum using a ratio of likelihoods from the policies π and π'.]

The new update should be:

$$\phi_{k+1} \leftarrow \arg\min_{\phi \in \Phi} \sum_{j,t} \Big(\prod_{i=1}^{N-1} \frac{\pi_k(\mathbf{a}_{j,t+i} \mid \mathbf{s}_{j,t+i})}{\pi'(\mathbf{a}_{j,t+i} \mid \mathbf{s}_{j,t+i})} \Big) \big(y_{j,t} - Q_\phi(\mathbf{s}_{j,t}, \mathbf{a}_{j,t}) \big)^2.$$

When $N = 1$, the additional product term vanishes and so no change is needed. As $N \to \infty$, the product term remains necessary.

2 Deep Q-Learning

2.1 Introduction

Part 1 of this assignment requires you to implement and evaluate Q-learning for playing Atari games. The Q-learning algorithm was covered in lecture, and you will be provided with starter code. This assignment will be faster to run on a GPU, though it is possible to complete on a CPU as well. Note that we use convolutional neural network architectures in this assignment. Therefore, we recommend using the Colab option if you do not have a GPU available to you. Please start early!

2.2 File overview

The starter code for this assignment can be found at

`https://github.com/berkeleydeeprlcourse/homework_fall2023/tree/main/hw3`

You will implement a DQN agent in `cs285/agents/dqn_agent.py` and `cs285/scripts/run_hw3_dqn.py`. In addition to those two files, you should start by reading the following files thoroughly:

- `cs285/env_configs/dqn_basic.py`: builds networks and generates configuration for the basic DQN problems (cartpole, lunar lander).
- `cs285/env_configs/dqn_atari.py`: builds networks and generates configuration for the Atari DQN problems.
- `cs285/infrastructure/replay_buffer.py`: implementation of replay buffer. You don't need to know how the memory efficient replay buffer works, but you should try to understand what each method does (particularly the difference between `insert`, which is called after a frame, and `on_reset`, which inserts the first observation from a trajectory) and how it differs from the regular replay buffer.
- `cs285/infrastructure/atari_wrappers.py`: contains some wrappers specific to the Atari environments. These wrappers can be key to getting challenging Atari environments to work!

There are two new package requirements (`gym[atari]` and `pip install gym[accept-rom-license]`) beyond what was used in the first two assignments; make sure to install these with `pip install -r requirements.txt` if you're re-using your Python environment from last assignment.