CS 449/549 - Homework 4

Ahmed Said Dönmez

Soft Actor-Critic and Hindsight Experience Replay

In this section I discuss the Soft Actor-Critic (SAC) algorithm and the Hindsight Experience Replay (HER) algorithm. I first provide an overview of the SAC algorithm and then discuss the HER algorithm.

Soft Actor-Critic

Soft Actor-Critic (SAC) is an off-policy algorithm that learns a stochastic policy in an environment with continuous action spaces. The algorithm both tries to maximizes the expeted reward while also keeping the entropy above a threshold. Usually without such constraint, the optimal policy is deterministic. Therefore, adding entropy constraint encourages exploration. The Soft Actor-Critic algorithm is similar to the actor-critic algorihtm, but its objectives are regularized by the entropy of the policy. The algrithm uses two Q-functions to reduce the positive bias. Also, it uses two target Q-networks whose weights are expinentially moving average of the original Q-networks. The algorihtm has the following loss functions for policy and Q-networks:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\mathbb{E}_{a_t \sim \pi_\phi} \left[\alpha \log(\pi_\phi(a_t \mid s_t)) - Q_\theta(s_t, a_t) \right] \right] \tag{1}$$

$$J_Q(\theta) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q_\theta(s_t, a_t) - \left(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p} \left[V_{\bar{\theta}}(s_{t+1}) \right] \right) \right)^2 \right] \tag{2}$$

The term α in the policy loss function is the temperature parameter that encourages exploration is updated using gradient ascent. The update rule is given by:

$$\alpha \leftarrow \alpha - \lambda \nabla_\alpha \mathcal{L}(\alpha) \tag{3}$$

where the loss function $J(\alpha)$ is given by:

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_t} \left[-\alpha \log \pi_t(a_t \mid s_t) - \alpha \bar{\mathcal{H}} \right]. \tag{4}$$

The algorihtm also uses a replay-buffer of size 10^6 so that past samples can also be used in the training to make the training more efficient. In this case, we use Hindsight Experience Replay (HER) as the replay buffer.

Hindsight Experience Replay

Hindsight Experience Replay (HER) is an algorithm that improves the sample efficiency of reinforcement learning algorithms by reusing past experiences. The algorithm works by generating additional goals for the agent to achieve, even if the original goal was not achieved. This allows the agent to learn from its failures and improve its performance. The algorithm is based on the idea of hindsight, which involves changing the goal of an episode after it has been completed. The algorithm works by storing past experiences in a replay buffer and sampling from the buffer to generate additional goals. The algorithm then replays the episode with the new goals and Q-function and policy is updated accordingly. The algorithm uses the following key components:

- After an episode ends, the transitions in an episode are saved to the buffer as usual.
- The algorithm also generates additional goals by sampling future achieved goals and using them as new goals. This allows the agent to learn from what it achieved additional to the actual goal.

Implementation

In this section we discuss the implementation of the SAC and HER algorithms.

SAC and HER Implementation

The SAC algorithm is implemented using the following key components:

- **Actor Network:** The actor network is a neural network that takes the state and goal as the combined state as input and outputs the mean and standard deviation of the action distribution. It also outputs the log probability of the action. Since we are using SAC, the policy has gaussian distribution and includes noise. The algorithm also tries to maximize the entropy of the policy.
- **Critic Networks:** The critic network is a neural network that takes the state and goal as the combined state and action as input and outputs the Q-value of the combined state-action pair. The critic network is trained using the soft Q-learning algorithm, which involves learning a Q-function that estimates the expected return of a state-action pair. There are two critics in the algorithm to reduce the positive bias.
- **Target Networks:** The target networks are used to stabilize the training of the Q-function. Their architecture is the same as critic networks. The target networks are updated using the exponential moving average of the original critic networks. There are two target networks similar to critic networks.
- **Temperature Parameter:** The temperature parameter is updated over time as well. The α parameter aims to maximize the entropy. The higher the α the higher the entropy and the explroation. It is varied during training.
- **Optimization:** The actor, critics and temperature parameter are optimized using the Adam optimizer. The optimizer is used to minimize the loss functions of the actor, critics and temperature parameter.
- **Replay Buffer:** The replay buffer is used to store past experiences and sample from them to train the Q-function and policy. The replay buffer is implemented as a circular buffer with a fixed size.
- **Hindsight Experience Replay:** After each episode, 4 random future states for a step in an episode is selected as achieved goals and they are added to the buffer after calculation of the reward. This way, the agent learns from its failures and improves its performance.

Results

I have implemented two algorithms in the task of FetchReach-V3 environment of the gymnasium-robotics library. I have trained the agent for 50 epochs, where each epoch consisted of 50 episodes and each episode consisted of 50 time steps. After each epoch of training, I have randomly selected a seed and tested the policy on that seed for 20 episodes. I have done this

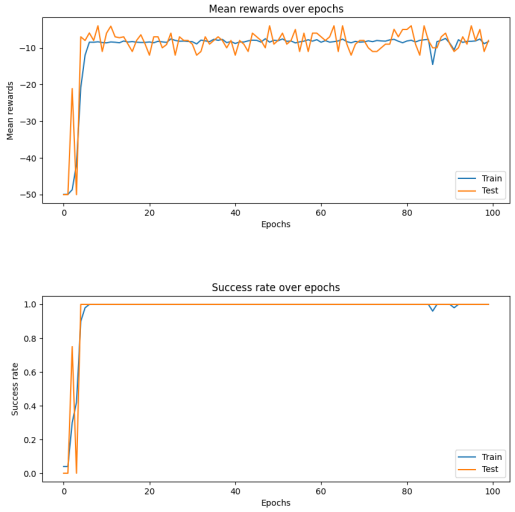

Figure 1: Training and Testing of SAC Algorithm over 50 epochs. Mean rewarsds per episode (above) and success rate (below) are shown.

testing for 50 different seeds due to having 50 epochs. The results of the training and testing over time are shown in the following figures.

The algorithm performed really well and converged after a few epochs as it can be seen in Figure . The rewards also increased quite alot and the robot was able to reach the target within a few steps only most of the time, and stay there until the end of the eposiode. The success rate was 100% after a few epochs for training and test cases.

Remark: Since the agent could not reach the target in many cases in the original environment due to some bug, I have changed reach.py file from the library of gymnasium-robotics so that the goal point is closer to the robot. This way, the goal was always within the configuration space of the robot.

Analysis

The resulting policy is quite stable and almost perform with perfect accuracy as we see in Figure . Also, for testing on 6 different seeds, the success rate was perfect for all the tests, and the robot was quickly reaching the goal. For this simple task, although edge-cases should be tested as well, the policy seems quite reliable.

Discussion About Randomness

In the implementation, I have used the following random seeds for the environment: ”4, 8, 15, 16, 23, 42”. Since the policy converged to a very optimal one with very low standard deviation, the results were almost the same for all the seeds. Also, the performance for many other different random seeds are perfect over 50 epochs. This shows that the policy is almost optimal and it is almost deterministic. Therefore, randomness has little effect on the perforamnce.