Efficient Exploration via Actor Critic Ensemble

Sihao Chen

Motivation and Background

Rainbow (Hessel et al.) proposed simultaneously combining improvements to DQN. Some tricks they tried:

- Target networks
- Prioritized experience replay
- Multi-step learning
- Noisy nets

I tried to adapt some of these techniques (along with others) to DDPG.

Quick Recap

Regular DDPG:

- Performs a Bellman backup:
  $y = r + \gamma \max_{a'} Q_\theta(s', a')$
- Q function loss:
  $\mathbb{E}[(Q_\theta(s,a) - y)^2]$
- Policy loss:
  $-\mathbb{E}[Q_\theta(s, \pi(s))]$

Ensemble DDPG

- Create K_1 independently initialized policy networks $\{\pi_{\phi_k}\}_{k=1}^{K_1}$
- Create K_2 independently initialized Q networks $\{Q_{\theta_i}\}_{i=1}^{K_2}$
- Pick actions with random noise using an ensemble function (introduced later)
- Priority is set to $p_i = |\delta| + \lambda \|\nabla_a Q(s_i, a_i|\theta^Q)\| + \epsilon$ [1]
- Delayed policy update and clipped gaussian noise like in TD3

[1] M. Vecerik *et al.*, “Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards,” *arXiv:1707.08817 [cs]*, Oct. 2018.

Ensemble Functions

- Need to use Q and policy networks to select an action
- One strategy is to use upper confidence bounds (UCB) [1]:
- Pick actions according to:

  $$\tilde{k}_t = \operatorname{argmax}_{k=1,2,\cdots,K_1} \{\mu(s_t, \pi_{\phi_k}(s_t)) + \sigma(s_t, \pi_{\phi_k}(s_t))\}$$
  $$a_t = \pi_{\phi_{\tilde{k}_t}}(s_t),$$

  where the mean and standard deviation are carried out over all Q network outputs
- Goal is to increase exploration by taking actions “we aren’t yet sure about”
- Another working strategy is choosing the action with the greatest minimum value of all Q-networks, minimizing the risk of the “worst case”.

[1] R. Y. Chen, S. Sidor, P. Abbeel, and J. Schulman, “UCB Exploration via Q-Ensembles,” *arXiv:1706.01502 [cs, stat]*, Nov. 2017.

Putting it all together

Algorithm 1 Full Ensemble DDPG

Input: number of policy networks K_1, number of Q networks K_2, total steps T, batch size N, discount γ, policy del d, polyak update factor τ, ensemble functions f_1, f_2

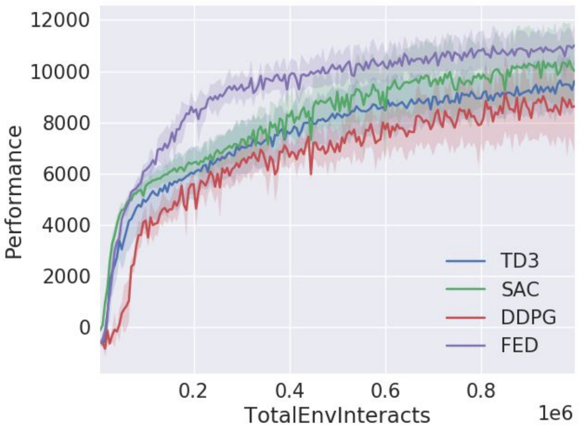
Initialize K_1 copies of independently initialized policy networks $\{\pi_{\phi_i}\}_{i=1}^{K_1}$
Initialize K_2 copies of independently initialized Q networks $\{Q_{\theta_j}\}_{j=1}^{K_2}$
Define $\Phi := \{\phi_i\}_{i=1}^{K_1}$, $\Theta := \{\theta_j\}_{j=1}^{K_2}$
Initialize target networks $\Phi' \leftarrow \Phi, \Theta' \leftarrow \Theta$
Initialize replay buffer $\mathcal{B}$
for step $t = 1, \cdots, T$ **do**
    Pick an action with random noise $a_t \leftarrow f_1(s_t : \Phi, \xi_t)$
    Take action a_t, receive state s_{t+1} and reward r_t from environment
    Add (s_t, a_t, r_t, s_{t+1}) to replay buffer $\mathcal{B}$
    Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
    Pick an action $\tilde{a} \leftarrow f_2(s', \Phi')$
    $y \leftarrow r + \gamma \min_j Q_{\theta'_j}(s', \tilde{a})$
    Update critics $\theta_j \leftarrow \mathrm{argmin}_{\theta_j} N^{-1} \sum (y - Q_{\theta_j}(s,a))^2, j = 1, \cdots, K_2$
    if t mod d **then**
        Update policies by the deterministic policy gradient:
        $\nabla_{\phi_i} J(\phi_i) = N^{-1} \sum \nabla_a \min_j Q_{\theta_j}(s,a)|_{a=\pi_{\phi_i}(s)} \nabla_{\phi_i} \pi_{\phi_i}(s), i = 1, \cdots, K_1$
        Update target networks:
        $\Theta' \leftarrow \tau\Theta + (1-\tau)\Theta'$
        $\Phi' \leftarrow \tau\Phi + (1-\tau)\Phi'$
    end if
end for

Algorithm 1 TD3

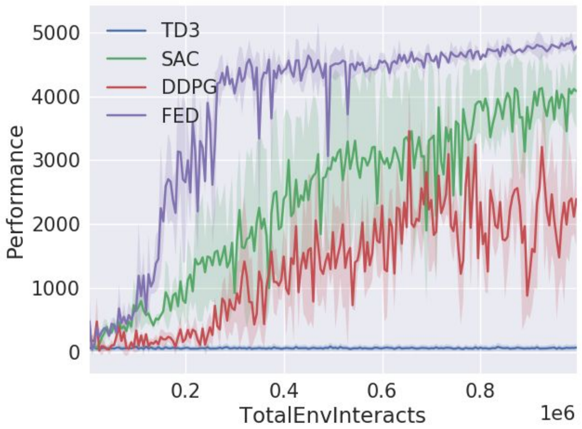
Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta'_1 \leftarrow \theta_1$, $\theta'_2 \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
    Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'
    Store transition tuple (s, a, r, s') in $\mathcal{B}$

    Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
    $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \mathrm{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
    $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \tilde{a})$
    Update critics $\theta_i \leftarrow \mathrm{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s,a))^2$
    if t mod d **then**
        Update ϕ by the deterministic policy gradient:
        $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
        Update target networks:
        $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$
        $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$
    end if
end for

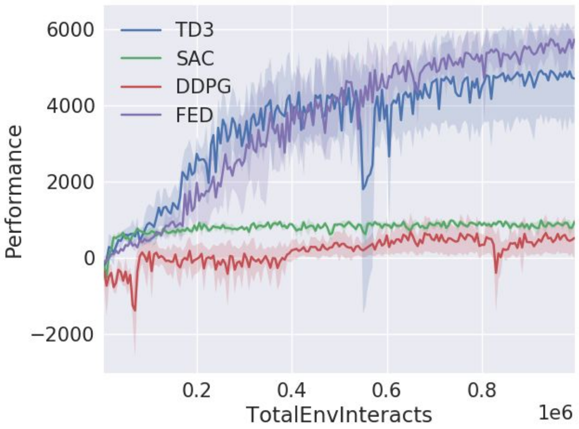
Results



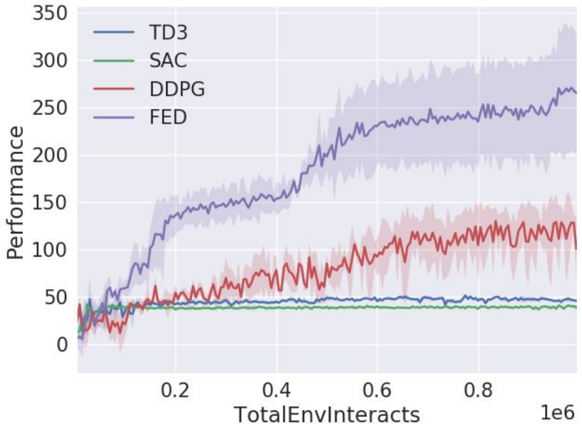
HalfCheetah-v2



Walker2d-v2



Ant-v2



Swimmer-v2

Discussion

Noisy Networks does not work well.

Parameter Space Noise is slow, since we need to re-assign the weights of all the networks multiple times when tuning the standard deviation of noise.