Addressing Function Approximation Error in Actor-Critic Methods

Twin Delayed Deep Deterministic Policy Gradient

JunhyunPark 2021-04-19

Summary

overestimation problem in Actor-Critic methods, TD3

- **Overestimation** of value function according to $\max_{a'} Q(s', a')$ in TD target.

- DPG -> DDPG -> **Twin Delayed** DDPG (TD3) + Regularization
  - **Twin** critic network: Q1, Q2 -> Clipped Double Q-Learning: $\min_{i=1,2} Q_i(s', \tilde{a})$
  - **Delayed**: update target critics & actor network every T updates of Q1, Q2
  - Noisy Action Regularization: $\tilde{a} = a + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma)$

TD3 (paper, NIPS '18)
https://arxiv.org/pdf/1802.09477.pdf

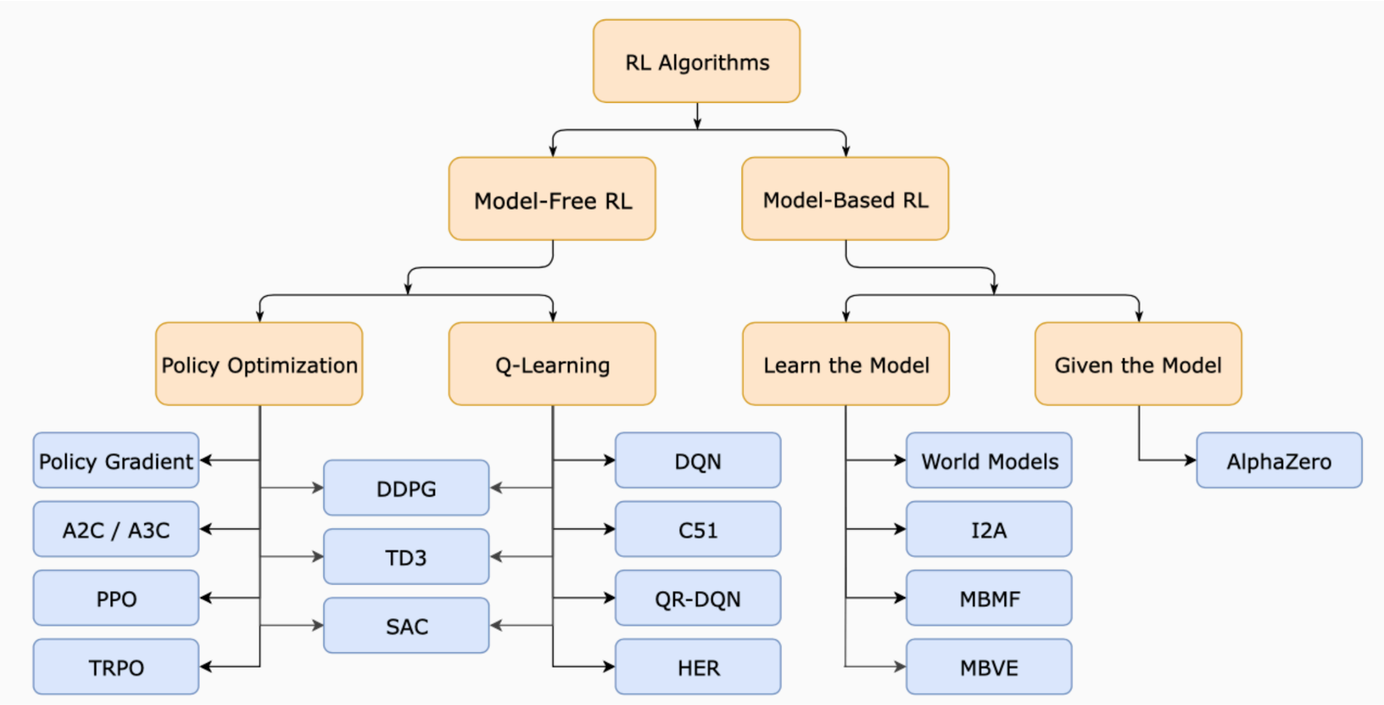
TD3 (blog)
https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html#td3

RL (blog)
https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

Preliminary

RL > DRL > Model-Free > Actor Critic

- Actor Critic = Value based + Policy based
  - Value based (ex) Q-learning, SARSA
    - model value function $Q_w(a|s)$
    - learn Q first, then we earn policy
    - + Deep Learning -> DQN
  - Policy based
    - model policy function $\pi_\theta(a|s)$
    - + Deep Learning -> Policy Gradient
  - DDPG = DQN + Policy Gradient with deterministic policy $\mu_\theta(s)$
    - model $Q_w(a|s)$ & $\pi_\theta(a|s)$ or $\mu_\theta(s)$



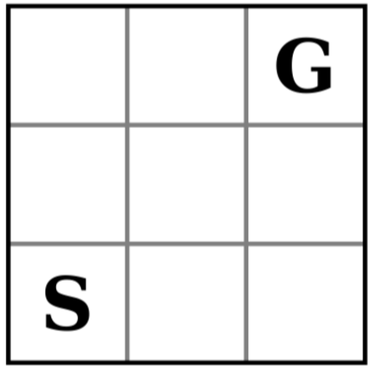
OpenAI DeepRL spinning up (documentation)
https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html

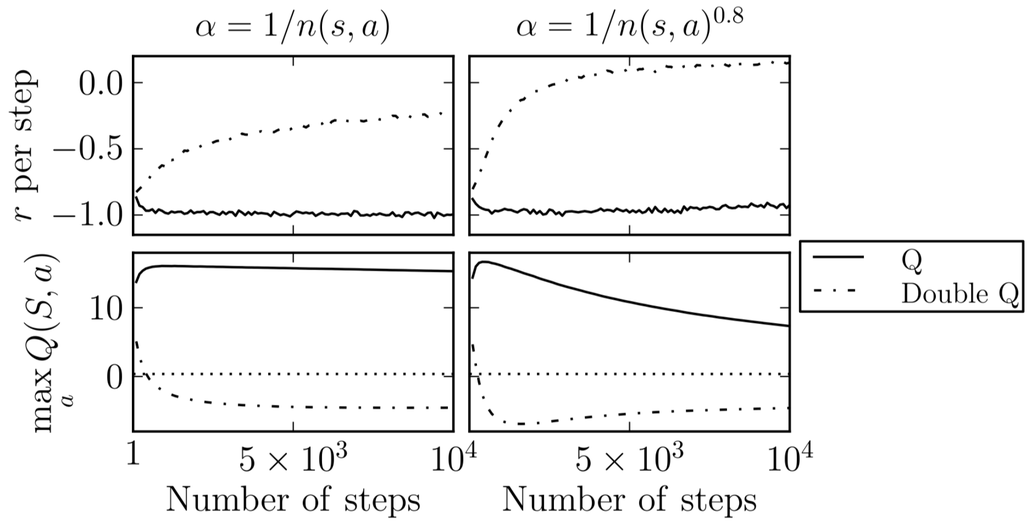
Overestimation Problem

in Q-learning setting: Double Q-learning

- generally good performance, but sometimes doesn't work
- problem of max estimator in Q-learning update formula

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right)$$





Double Q-Learning (paper, NIPS '10)
https://papers.nips.cc/paper/2010/hash/091d584fced301b442654dd8c23b3fc9-Abstract.html

Solution

Double Estimator

Algorithm 1 Double Q-learning

Initialize Q^A,Q^B,s
repeat
  Choose a, based on $Q^A(s,\cdot)$ and $Q^B(s,\cdot)$, observe r, s'
  Choose (e.g. random) either UPDATE(A) or UPDATE(B)
  if UPDATE(A) **then**
    Define $a^* = \arg\max_a Q^A(s',a)$
    $Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)\left(r + \gamma Q^B(s',a^*) - Q^A(s,a)\right)$
  else if UPDATE(B) **then**
    Define $b^* = \arg\max_a Q^B(s',a)$
    $Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)(r + \gamma Q^A(s',b^*) - Q^B(s,a))$
  end if
  $s \leftarrow s'$
until end

Double DQN (paper, NIPS ’10)
https://arxiv.org/abs/1509.06461

Overestimation Problem

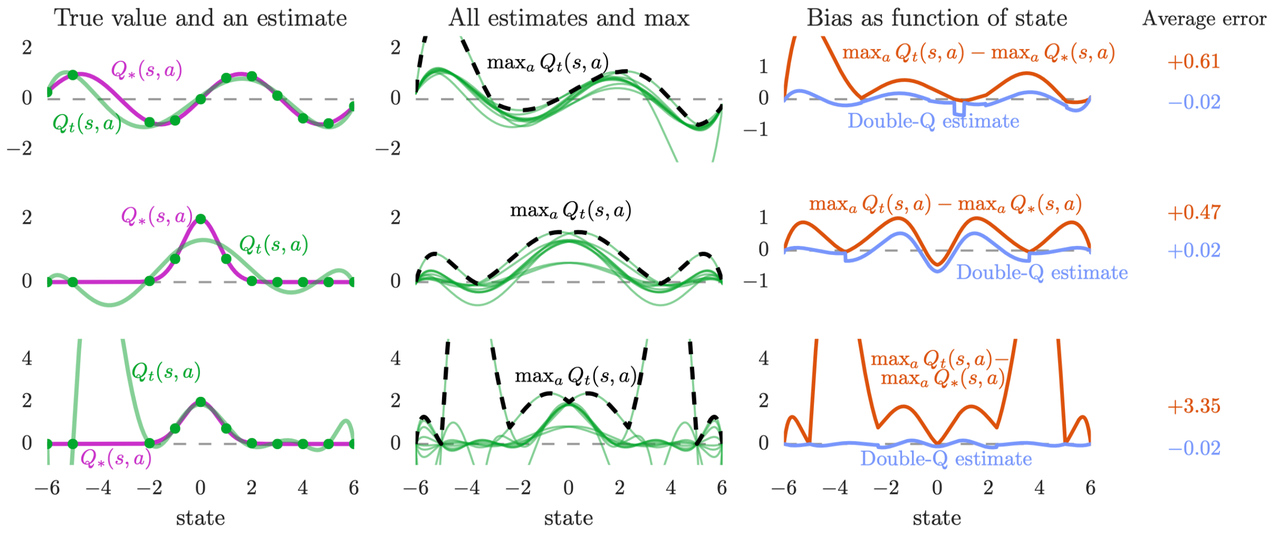
in DQN setting: Double DQN

- same problem in Deep Learning setting

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t)\left(r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right)$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha(Y_t^{\mathrm{Q}} - Q(S_t, A_t; \boldsymbol{\theta}_t))\nabla_{\boldsymbol{\theta}_t} Q(S_t, A_t; \boldsymbol{\theta}_t)$$

$$Y_t^{\mathrm{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-)$$



Double DQN (paper, NIPS ’10)
https://arxiv.org/abs/1509.06461

Solution
Double Estimator

Algorithm 1 Double Q-learning

Initialize Q^A,Q^B,s
repeat
  Choose a, based on $Q^A(s,\cdot)$ and $Q^B(s,\cdot)$, observe r, s'
  Choose (e.g. random) either UPDATE(A) or UPDATE(B)
  if UPDATE(A) **then**
    Define $a^* = \arg\max_a Q^A(s',a)$
    $Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)\left(r + \gamma Q^B(s',a^*) - Q^A(s,a)\right)$
  else if UPDATE(B) **then**
    Define $b^* = \arg\max_a Q^B(s',a)$
    $Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)(r + \gamma Q^A(s',b^*) - Q^B(s,a))$
  end if
  $s \leftarrow s'$
until end

- TD Update in Q-Learning

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha(Y_t^{\mathrm{Q}} - Q(S_t, A_t; \boldsymbol{\theta}_t))\nabla_{\boldsymbol{\theta}_t} Q(S_t, A_t; \boldsymbol{\theta}_t)$$

$$Y_t^{\mathrm{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-)$$

- Selection & Evaluation

$$Y_t^{\mathrm{Q}} = R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t)$$

$$Y_t^{\mathrm{DoubleQ}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t')$$

$$Y_t^{\mathrm{DoubleDQN}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname*{argmax}_a Q(S_{t+1}, a; \boldsymbol{\theta}_t), \boldsymbol{\theta}_t^-)$$

Double DQN (paper, NIPS ’10)
https://arxiv.org/abs/1509.06461

Overestimation Problem

in Actor-Critic setting: TD3

Algorithm 1 DDPG algorithm

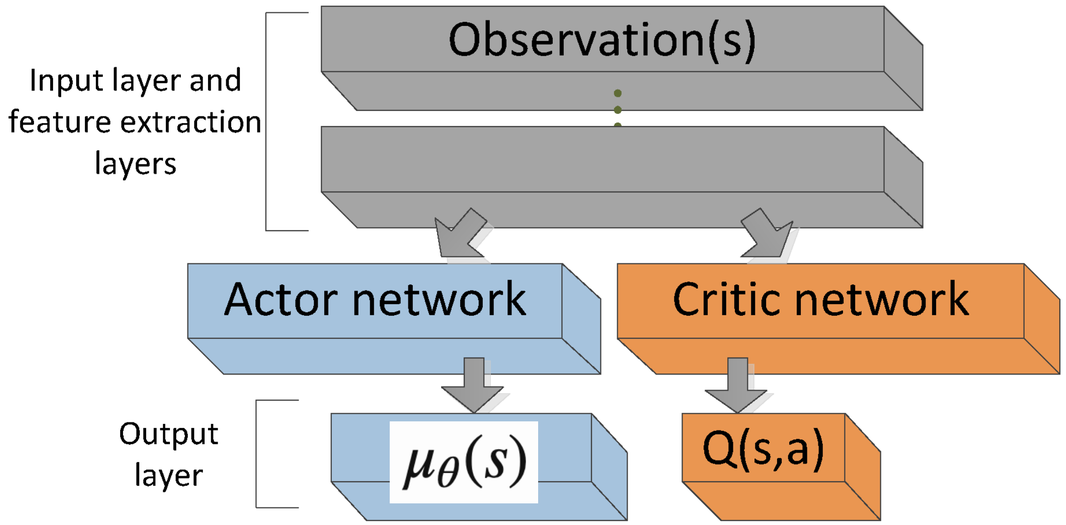
Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for



DDPG (paper, ICLR '16)
https://arxiv.org/abs/1509.02971

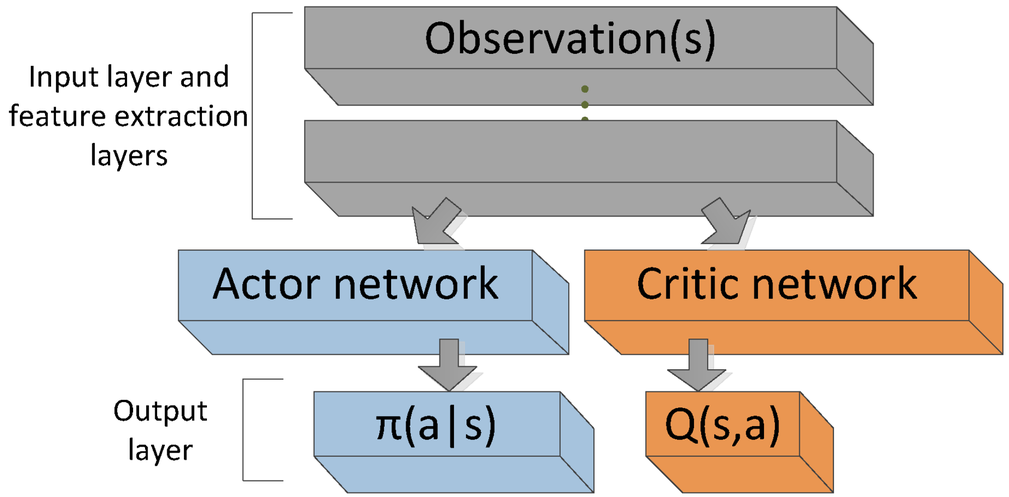
Preliminary

RL > DRL > Model-Free > Actor Critic

Actor Update

$$J(\theta) = \sum_{s \in \mathcal{S}} d^{\pi}(s) V^{\pi}(s) = \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(a|s) Q^{\pi}(s, a)$$

$$\begin{aligned} \nabla_{\theta} J(\theta) &\propto \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} Q^{\pi}(s, a) \nabla_{\theta} \pi_{\theta}(a|s) \\ &= \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(a|s) Q^{\pi}(s, a) \frac{\nabla_{\theta} \pi_{\theta}(a|s)}{\pi_{\theta}(a|s)} \\ &= \mathbb{E}_{\pi}[Q^{\pi}(s, a) \nabla_{\theta} \ln \pi_{\theta}(a|s)] \end{aligned}$$



Critic Update

$$\delta_t = r_t + \gamma Q_w(s', a') - Q_w(s, a)$$

$$w \leftarrow w + \alpha_w \delta_t \nabla_w Q_w(s, a)$$

TD3 (blog)
https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html#td3

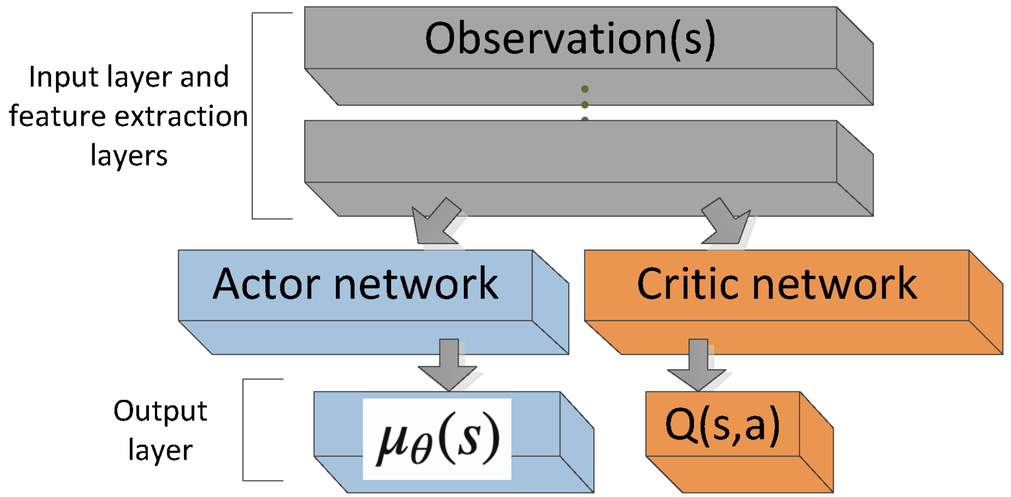
Preliminary

RL > DRL > Model-Free > Actor Critic > DDPG

Actor Update

$$J(\theta) = \sum_{s \in \mathcal{S}} d^{\pi}(s) V^{\pi}(s) = \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(a|s) Q^{\pi}(s, a)$$

$$\begin{aligned} \nabla_{\theta} J(\theta) &\propto \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} Q^{\pi}(s, a) \nabla_{\theta} \pi_{\theta}(a|s) \\ &= \sum_{s \in \mathcal{S}} d^{\pi}(s) \sum_{a \in \mathcal{A}} \pi_{\theta}(a|s) Q^{\pi}(s, a) \frac{\nabla_{\theta} \pi_{\theta}(a|s)}{\pi_{\theta}(a|s)} \\ &= \mathbb{E}_{\pi}[Q^{\pi}(s, a) \nabla_{\theta} \ln \pi_{\theta}(a|s)] \end{aligned}$$



Critic Update

$$\delta_t = r_t + \gamma Q_w(s', a') - Q_w(s, a)$$

$$w \leftarrow w + \alpha_w \delta_t \nabla_w Q_w(s, a)$$

TD3 (blog)
https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html#td3

Preliminary

RL > DRL > Model-Free > Actor Critic > DDPG

Algorithm 1 DDPG algorithm

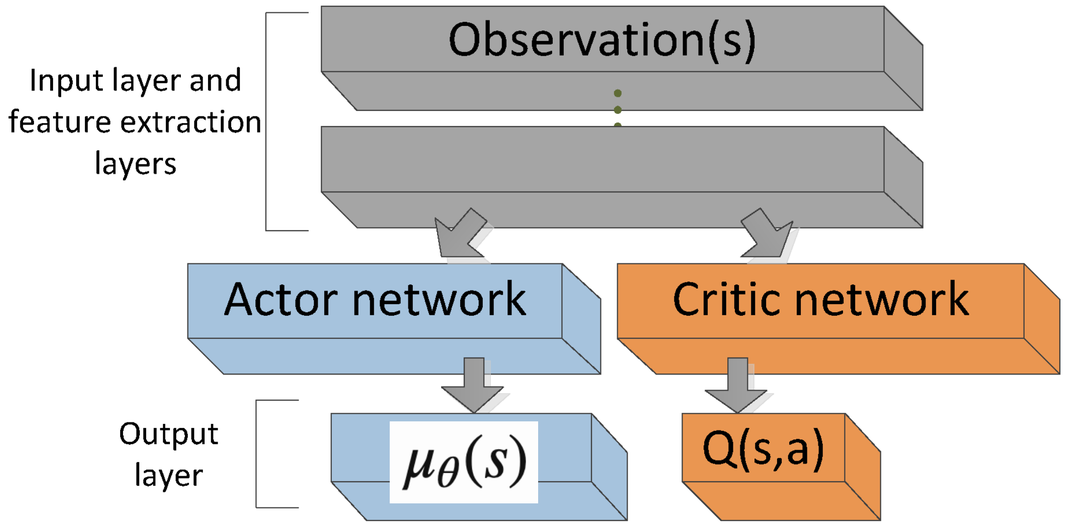
Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

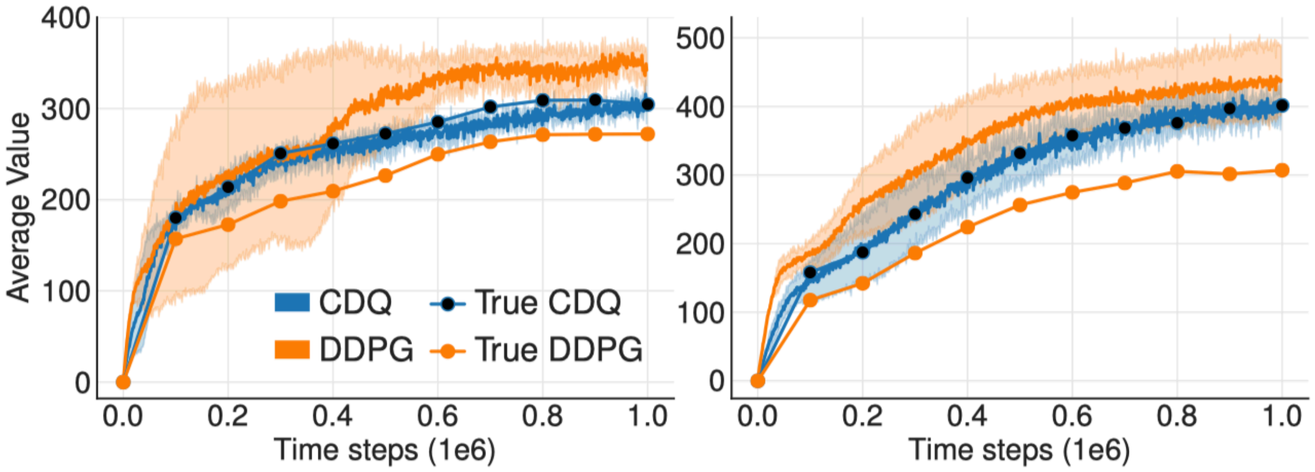


DDPG (paper, ICLR '16)
https://arxiv.org/abs/1509.02971

Overestimation Problem

in Actor-Critic setting: TD3

- still, problem!



(a) Hopper-v1 (b) Walker2d-v1

DQN Update

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha(Y_t^{\mathrm{Q}} - Q(S_t, A_t; \boldsymbol{\theta}_t))\nabla_{\boldsymbol{\theta}_t} Q(S_t, A_t; \boldsymbol{\theta}_t)$$

$$Y_t^{\mathrm{DQN}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t^-)$$

Critic Update

$$\delta_t = r_t + \gamma Q_w(s', a') - Q_w(s, a)$$

$$w \leftarrow w + \alpha_w \delta_t \nabla_w Q_w(s, a)$$

TD3 (paper, NIPS '18)
https://arxiv.org/pdf/1802.09477.pdf

Solution

Twin Critics + Clipped Estimator

- Original

$$y = r + \gamma Q_{\theta'}(s', \pi_{\phi}(s'))$$

- Twin Critics

$$y_1 = r + \gamma Q_{\theta'_2}(s', \pi_{\phi_1}(s'))$$
$$y_2 = r + \gamma Q_{\theta'_1}(s', \pi_{\phi_2}(s'))$$

- Clipped TD target

$$y_1 = r + \gamma \min_{i=1,2} Q_{\theta'_i}(s', \pi_{\phi_1}(s'))$$

TD3 (paper, NIPS '18)
https://arxiv.org/pdf/1802.09477.pdf

Full Method

Twin Critics + Clipping + Regularization + Delayed Update

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 $a \sim \pi(s) + \epsilon$,
  $\epsilon \sim \mathcal{N}(0, \sigma)$ 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 $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s) + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$ Target policy smoothing
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$ Clipped Double Q-learning
  Update critics $\theta_i \leftarrow \min_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if $t \bmod d$ **then** Delayed update of target and policy networks
    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

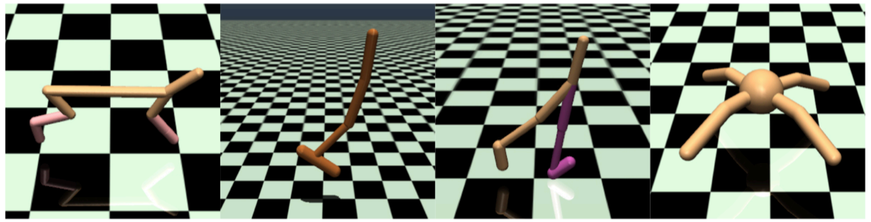
- Twin Networks
- Exploration Noise
- Noisy Action Regularization
  - = Target policy smoothing
- Clipped TD target
  - = Clipped Double Q-learning
- Delayed update: Target Critics & Actor

TD3 (paper, NIPS '18)
https://arxiv.org/pdf/1802.09477.pdf

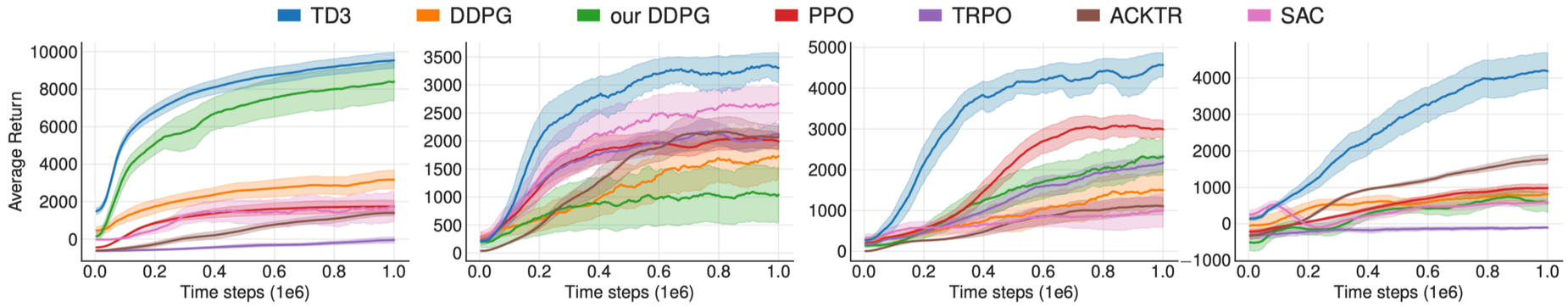
TD3 (blog)
https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html#td3

Experiment

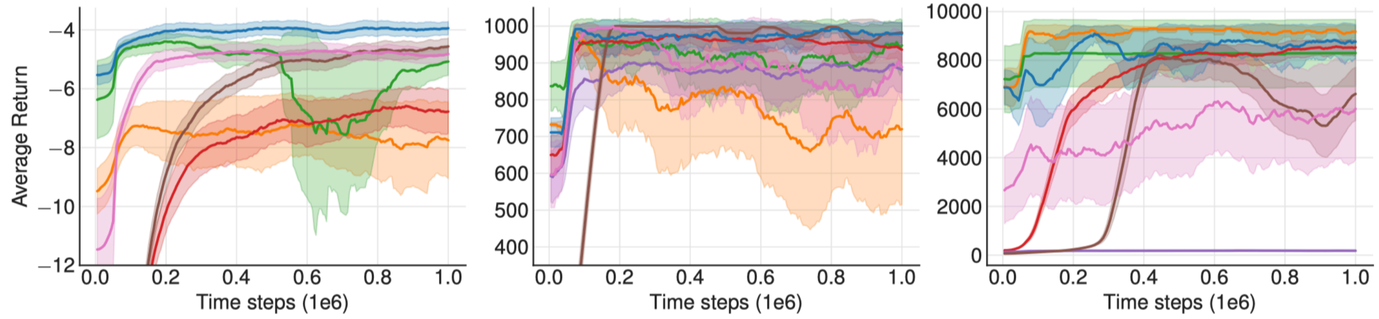
Mujoco Environment



(a) (b) (c) (d)



(a) HalfCheetah-v1 (b) Hopper-v1 (c) Walker2d-v1 (d) Ant-v1



(e) Reacher-v1 (f) InvertedPendulum-v1 (g) InvertedDoublePendulum-v1

Summary

Address overestimation problem in Actor-Critic methods by TD3

- **Overestimation** of value function according to $\max_{a'} Q(s', a')$ in TD target.

- DPG -> DDPG -> **Twin Delayed** DDPG (TD3) + Regularization
  - **Twin** critic network: Q1, Q2 -> Clipped Double Q-Learning: $\min_{i=1,2} Q_i(s', \tilde{a})$
  - **Delayed**: update target critics & actor network every T updates of Q1, Q2
  - Noisy Action Regularization: $\tilde{a} = a + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma)$

TD3 (paper, NIPS '18)
https://arxiv.org/pdf/1802.09477.pdf

TD3 (blog)
https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html#td3

RL (blog)
https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html

Additionals

Ablation study

Table 2. Average return over the last 10 evaluations over 10 trials of 1 million time steps, comparing ablation over delayed policy updates (DP), target policy smoothing (TPS), Clipped Double Q-learning (CDQ) and our architecture, hyper-parameters and exploration (AHE). Maximum value for each task is bolded.

Method	HCheetah	Hopper	Walker2d	Ant
TD3	9532.99	**3304.75**	**4565.24**	**4185.06**
DDPG	3162.50	1731.94	1520.90	816.35
AHE	8401.02	1061.77	2362.13	564.07
AHE + DP	7588.64	1465.11	2459.53	896.13
AHE + TPS	9023.40	907.56	2961.36	872.17
AHE + CDQ	6470.20	1134.14	3979.21	3818.71
TD3 - DP	9590.65	2407.42	**4695.50**	3754.26
TD3 - TPS	8987.69	2392.59	4033.67	**4155.24**
TD3 - CDQ	9792.80	1837.32	2579.39	849.75
DQ-AC	9433.87	1773.71	3100.45	2445.97
DDQN-AC	**10306.90**	2155.75	3116.81	1092.18

6.2. Ablation Studies

We perform ablation studies to understand the contribution of each individual component: Clipped Double Q-learning (Section 4.2), delayed policy updates (Section 5.2) and target policy smoothing (Section 5.3). We present our results in Table 2 in which we compare the performance of removing each component from TD3 along with our modifications to the architecture and hyper-parameters. Additional learning curves can be found in the supplementary material.

The significance of each component varies task to task. While the addition of only a single component causes insignificant improvement in most cases, the addition of combinations performs at a much higher level. The full algorithm outperforms every other combination in most tasks. Although the actor is trained for only half the number of iterations, the inclusion of delayed policy update generally improves performance, while reducing training time.

We additionally compare the effectiveness of the actor-critic variants of Double Q-learning (Van Hasselt, 2010) and Double DQN (Van Hasselt et al., 2016), denoted DQ-AC and DDQN-AC respectively, in Table 2. For fairness in comparison, these methods also benefited from delayed policy updates, target policy smoothing and use our architecture and hyper-parameters. Both methods were shown to reduce overestimation bias less than Clipped Double Q-learning in Section 4. This is reflected empirically, as both methods result in insignificant improvements over TD3 - CDQ, with an exception in the Ant-v1 environment, which appears to benefit greatly from any overestimation reduction. As the inclusion of Clipped Double Q-learning into our full method outperforms both prior methods, this suggests that subduing the overestimations from the unbiased estimator is an effective measure to improve performance.

Additionals

When do actor-critic methods fail to learn? These results suggest that the divergence that occurs without target networks is the result of policy updates with a high variance value estimate. Figure 3, as well as Section 4, suggest failure can occur due to the interplay between the actor and critic updates. Value estimates diverge through overestimation when the policy is poor, and the policy will become poor if the value estimate itself is inaccurate.

Table 1. Max Average Return over 10 trials of 1 million time steps. Maximum value for each task is bolded. ± corresponds to a single standard deviation over trials.

Environment	TD3	DDPG	Our DDPG	PPO	TRPO	ACKTR	SAC
HalfCheetah	**9636.95 ± 859.065**	3305.60	8577.29	1795.43	-15.57	1450.46	2347.19
Hopper	**3564.07 ± 114.74**	2020.46	1860.02	2164.70	2471.30	2428.39	2996.66
Walker2d	**4682.82 ± 539.64**	1843.85	3098.11	3317.69	2321.47	1216.70	1283.67
Ant	**4372.44 ± 1000.33**	1005.30	888.77	1083.20	-75.85	1821.94	655.35
Reacher	**-3.60 ± 0.56**	-6.51	**-4.01**	-6.18	-111.43	-4.26	-4.44
InvPendulum	**1000.00 ± 0.00**	**1000.00**	**1000.00**	**1000.00**	985.40	**1000.00**	**1000.00**
InvDoublePendulum	**9337.47 ± 14.96**	**9355.52**	8369.95	8977.94	205.85	9081.92	8487.15