

Implicit Distributional Reinforcement Learning (IDAC)

백승언

30 May, 2022

- **Introduction**

- Estimation of probability distribution
- Limitation of distribution estimation in conventional Reinforcement Learning

- **Implicit Distributional Actor-Critic(IDAC)**

- Motivation
- Backgrounds
- IDAC

- **Experiment results**

- **Implementation of IDAC**

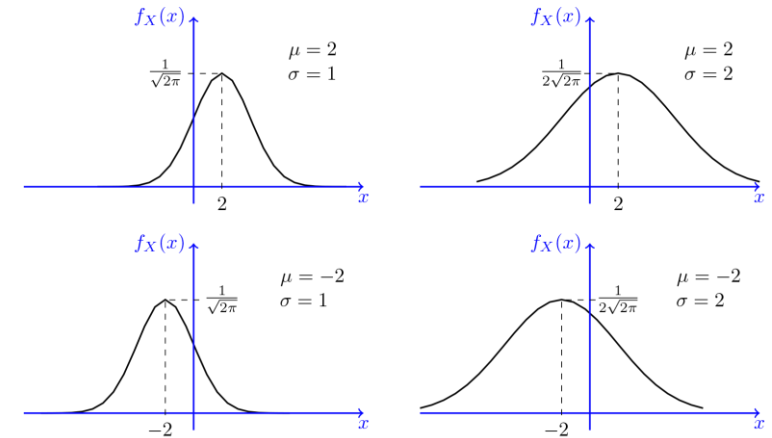
Introduction

Estimation of probability distribution

● Moments of a function are quantitative measures related to shape of the function's graph

- Probability distribution functions(PDF) are generally represented with 4 measures, mean, variance, skewness, kurtosis

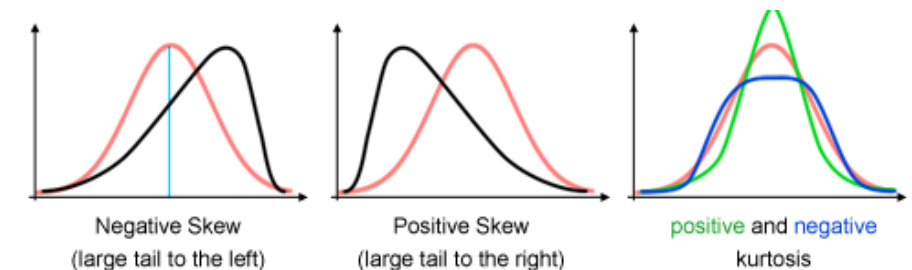
- Mean : first moment of PDF, ($\mu = E[X]$)
- Variance : second moment of PDF, ($\sigma^2 = E[(X - \mu)^2]$)
- Skewness : third moment of PDF, ($\gamma = E\left[\left(\frac{X - \mu}{\sigma}\right)^3\right]$)
- Kurtosis : fourth moment of PDF, ($Kurt[x] = \frac{E[(X - \mu)^4]}{(E[(X - \mu)^2])^2}$)



Various shape of normal dist. based on μ, σ

● Estimation of PDF $f(x)$ requires following two-steps

- 1. Determination that variable x follows known distributions(Bernoulli, Binomial, Normal, Beta, Poisson, ..)
- 2. Estimate the **parameter(moment)** of pdf using stacked data

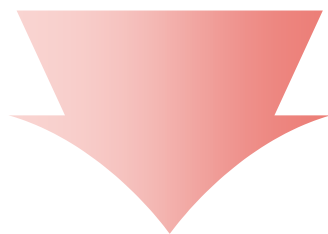


Various shape of normal dist. based on $\gamma, Kurt[x]$

Limitation of distribution estimation in conventional Reinforcement Learning

● Naïve approaches in estimation of return

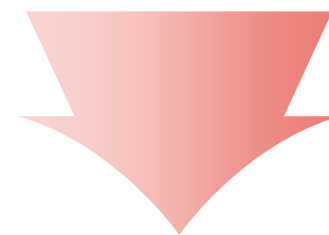
- Due to stochastic nature of return G , in general RL, G is approximated with $Q(s, a)$ and $V(s)$
 - $G = \sum \gamma^t r_{t+1}$
 - $Q(s, a) = E[\sum \gamma^t r_{t+1} | s, a]$
 - $V(s) = E[\sum \gamma^t r_{t+1} | s]$
- $Q(s, a), V(s)$ are generally estimated with normal bellman operator \mathcal{T}_B using only mean value
 - $(\mathcal{T}_B V)(s) = E[r(s, \pi(s))] + \gamma E[V(s')]$
 - $(\mathcal{T}_B Q)(s, a) = E[r(s, a)] + \gamma E[Q(s', a')]$



Variance, skewness, kurtosis of return G are easily neglected

● Naïve assumption of policy distribution with diagonal normal dist.

- In deterministic policy
 - $a = \mu(s) + \epsilon, \epsilon \sim N(0, I)$
- In stochastic policy
 - $a \sim N(\mu(s), \text{diag}\{\sigma^2(s)\})$



Diagonal Gaussian model could not capture the multi-modality, relation between action dimensions

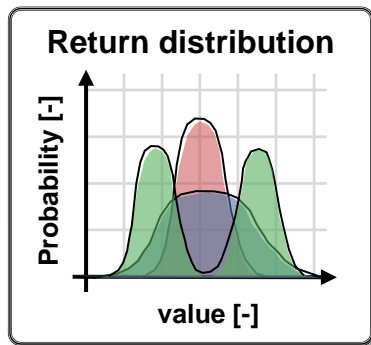
Implicit Distributional Actor-Critic (IDAC)

<https://arxiv.org/pdf/2007.06159.pdf>

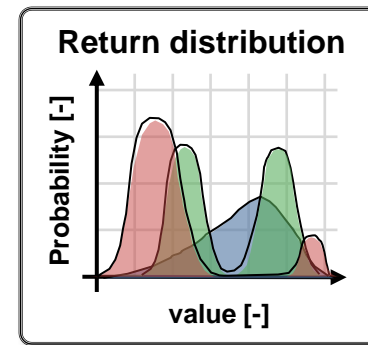
Motivations

- **Consideration of the intrinsic randomness of return¹ and policy², and further integrate³ these two frameworks for strengthen each other**

- **1** : Using distributional critic and distributional bellman operator \mathcal{T}_Z for accurate estimation
 - $(\mathcal{T}_Z Z)(s, a) \stackrel{D}{=} r(s, a) + \gamma Z(s', a')$, Z is distribution of random return, $\stackrel{D}{=}$ denotes "equal in distribution"
 - Thanks to this setup, return could directly be modeled rather than expectation



These distributions have same mean, but they are not same!



Complex dist. could be modeled in this framework!

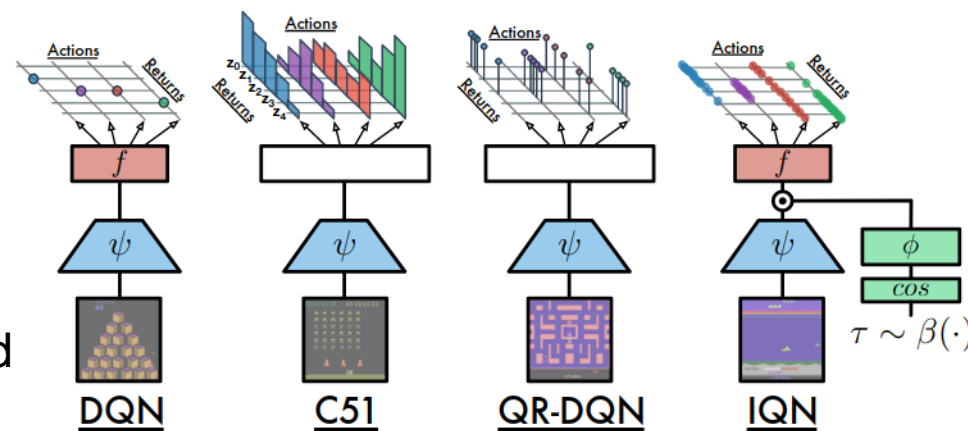
- **2** : Using semi-implicit actor for better modeling of stochastic policy
 - $\pi_\theta(a|s) = \int_{\xi_0} \pi_\theta(a|s, \xi) p(\xi) d\xi$, where $\pi_\theta(a|s, \xi) = \mathcal{N}(a; \mu_\theta(s, \xi), \text{diag}\{\sigma_\theta^2(s, \xi)\})$, $\xi \sim p(\xi)$ denotes a random noise
 - Thanks to this setup, states could affect the mean and covariance of diagonal Gaussian policy.
- **3** : Harmonious integration of these two frameworks into the actor-critic framework

● Distributional RL

- In distributional RL, the cumulative return of a chosen action at state is modeled with the full distribution rather than expectation of it, $Z_{\theta}(s, a) := \frac{1}{N} \sum_{i=1}^N \delta_{\theta_i}(s, a)$
 - So that the model can capture its intrinsic randomness instead of just first-order moment(skewness, multi-modality in state-action value function)
- Distributional RL algorithms
 - **C51** : Pre-determination of the supports of return distribution and then training the categorical distribution.
 - **QR-DQN** : To avoid pre-determined supports, and decrease theory-practice gap, introducing the quantile regression
 - **IQN** : Using sampling the quantile, training the full quantile function and consider risk of policy

● RL Algorithms that influenced IDAC

- **TD3** : To mitigate the overestimation of Q value, using twin network and adjusting update frequency between actor, critic
- **SAC** : To learn a relevant policy entropy in entropy regularized RL, proposing the auto tuning of entropy coefficient α
 - $J(\theta) = E[G - \alpha \log \pi_{\theta}(a|s)]$



Network architectures for distributional RL algorithms

● Overview of IDAC

- Using deep generative network(DGN) for satisfying distributional matching between true return G and G_ω
 - $G_\omega(s, a, \epsilon) \stackrel{D}{=} R(s, a) + \gamma G_\omega(s', a', \epsilon')$, where $\epsilon, \epsilon' \sim iid p(\epsilon)$
- Applying twin-delayed DGNs for mitigating the overestimation of $G_\omega(s, a, \epsilon)$
 - Authors designed two DGNs $G_{\omega_1}(s, a, \epsilon)$ and $G_{\omega_2}(s, a, \epsilon)$ which independent initialization of ω_1, ω_2 and independent input noises
 - Target quantile is determined by element-wise minimum operation
 - Original target quantiles are following: $(\overrightarrow{y_{1,1}}, \dots, \overrightarrow{y_{1,K}}) = \text{sort}(y_{1,1}, \dots, y_{1,K})$, $(\overrightarrow{y_{2,1}}, \dots, \overrightarrow{y_{2,K}}) = \text{sort}(y_{2,1}, \dots, y_{2,K})$
 - New target quantiles for twin DGNs become: $(\overrightarrow{y_1}, \dots, \overrightarrow{y_K}) = (\min(\overrightarrow{y_{1,1}}, \overrightarrow{y_{2,1}}), \dots, \min(\overrightarrow{y_{1,K}}, \overrightarrow{y_{2,K}}))$
- Using semi-implicit policy that enriches the diagonal Gaussian distribution by randomizing its parameters with another distribution
 - $\pi_\theta(a|s) = \int_{\xi_0} \pi_\theta(a|s, \xi) p(\xi) d\xi$, where $\pi_\theta(a|s, \xi) = \mathcal{N}(a; \mu_\theta(s, \xi), \text{diag}\{\sigma_\theta^2(s, \xi)\})$, $\xi \sim p(\xi)$ denotes a random noise
 - Based on previous proofs, semi-implicit distribution can extract the lower bound of entropy
 - $\mathcal{H}_L := -E_{\xi^0, \dots, \xi^L \sim iid p(\xi)} E_{a \sim \pi_\theta(a|s, \xi^0)} \left[\log \frac{1}{\sum_{l=0}^L \pi_\theta(a|s, \xi^l)} \right]$, $\mathcal{H}_l \leq \mathcal{H}_{l+1} \leq \mathcal{H} := E_{a \sim \pi_\theta(a|s)} [\log \pi_\theta(a|s)]$, $\forall l \geq 0$
 - Thus, entropy regularization term could be used in actor loss

● Critic loss of IDAC

- Unlike the conventional RL algorithms, distributional algorithms needed the novel metric
 - Conventional RL algorithm: $Q(s, a)$ and TD-target is scalar, so, loss function is constructed with L2 distance
 - $L(\theta) = r + \gamma Q_\theta(s', a') - Q_\theta(s, a)$
 - Distributional RL algorithm: $Z(s, a)$ and TD-target is distribution, so loss function should measure the distance between two distributions.
 - $L(\theta) = D(r + \gamma Z_\theta(s', a') || Z(s, a))$
 - In C-51, KL divergence is used. (fixed support), In QR-DQN and IQN, p - Wasserstein distance is used (through quantile regression)
- In QR-DQN and IQN, p - Wasserstein distance is approximated by that between their empirical distributions supported on K random samples.
 - $\hat{X} = \frac{1}{K} \sum_{k=1}^K \delta_{x_k}$, $\hat{Y} = \frac{1}{K} \sum_{k=1}^K \delta_{y_k}$, $W_p(X, Y)^p \approx W_p(\hat{X}, \hat{Y})^p = \frac{1}{K} \sum_{k=1}^K |\vec{x}_k - \vec{y}_k|^p$
 - At this time, $\vec{x}_{1:K}$ is obtained by sorting the K elements in $x_{1:K}$ in increasing order.
 - However, QR-DQN, IQN could not satisfy this constraint, thus they use complex loss.
 - $L_{QR}(X, Y) \approx L_{QR}(\hat{X}, \hat{Y}) = \frac{1}{K^2} \sum_{k=1}^K \sum_{k'=1}^K \rho_{\tau_k}^\kappa(y_{k'} - \vec{x}_k)$
 - $\rho_\tau^\kappa(u) = |\tau - \mathbb{1}_{\{u < 0\}}| L_\kappa(u)$, $u = r + \delta_j(s', a') - \delta_j(s, a)$, $L_\kappa(u) = \begin{cases} -\frac{1}{2}u^2, & |u| < \kappa \\ \kappa(|u| - \frac{1}{2}\kappa), & \text{otherwise} \end{cases}$, $\tau_k = (k - 0.5)/K$
- In IDAC, critics could generate $\{G_\omega(s, a, \epsilon_1), \dots, G_\omega(s, a, \epsilon_K)\} := x_{1:K}$ with *iid* sampled noise $\epsilon_{1:K}$, thus, $\vec{x}_{1:K}$ is easily obtained with sort
 - The objective function of DGN parameter ω becomes
 - $J(\omega) = L_{QR}(\hat{X}, \text{StopGradient}\{\hat{Y}\}) = \frac{1}{K^2} \sum_k \sum_{k'} \rho_{\tau_k}^\kappa(\text{StopGradient}\{y_{k'}\} - \vec{x}_k)$

● Pseudo code

- Hyper-params inputting $(\lambda, \tau, \theta, \omega_1, \omega_2, \eta)$, and target network initialization corresponds with main network
 - Replay buffer initialization (\mathcal{D})
- Every single step, noise parameter ξ_t is sampled, and then action a_t is sampled using s_t, ξ_t
 - Env proceeds one step, transition is stored in replay buffer
- Transitions sampled from replay buffer
- Minimum target value is determined between $G_{\omega_1}(s', a')$ and $G_{\omega_2}(s', a')$
 - QR-loss is computed and ω are updated
- Policy entropy is computed
 - Actor loss is computed and θ are updated
- Delayed update is proceeded

Algorithm 1 IDAC: Implicit Distributional Actor-Critic (see Appendix B for more implementation details)

Require: Learning rate λ , smoothing factor τ . Initial policy network parameter θ , distributional generator network parameters ω_1, ω_2 , entropy coefficient η ;

$\tilde{\omega}_1 \leftarrow \omega_1, \tilde{\omega}_2 \leftarrow \omega_2, \mathcal{D} \leftarrow \emptyset$

for Each iteration **do**

for Each environment step **do**

$\xi_t \sim p(\xi), a_t \sim \pi_\theta(\cdot | s_t, \xi_t)$ {Sample noise and then action}

$s_{t+1} \sim p(\cdot | s_t, a_t)$ {Observe next state}

$\mathcal{D} \leftarrow \mathcal{D} \cup (s_t, a_t, r_t, s_{t+1})$ {Store transition tuples}

end for

 Sample transitions from the replay buffer

$\omega_i \leftarrow \omega_i - \lambda \nabla_{\omega_i} J(\omega_i)$ for $i = 1, 2$ {Update DGNs, Eq. (10)}

$\theta \leftarrow \theta - \lambda \nabla_\theta \bar{J}(\theta)$ {Update SIA, Eq. (16)}

$\eta \leftarrow \eta - \lambda \nabla_\eta J(\eta)$, let $\alpha = \exp(\eta)$ {Update entropy coefficient, Eq. (17)}

$\tilde{\omega}_i \leftarrow \tau \omega_i + (1 - \tau) \tilde{\omega}_i$ for $i = 1, 2$ {Soft update delayed networks}





end for

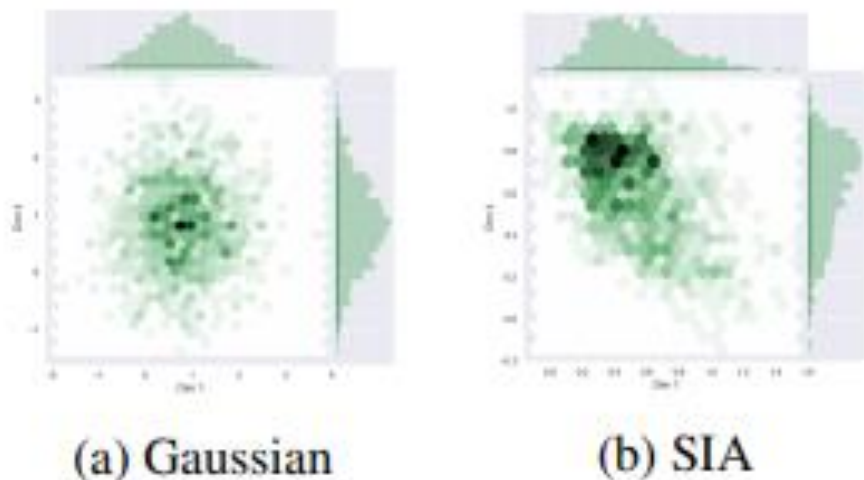
Pseudo code of gSDE

Experiment Results

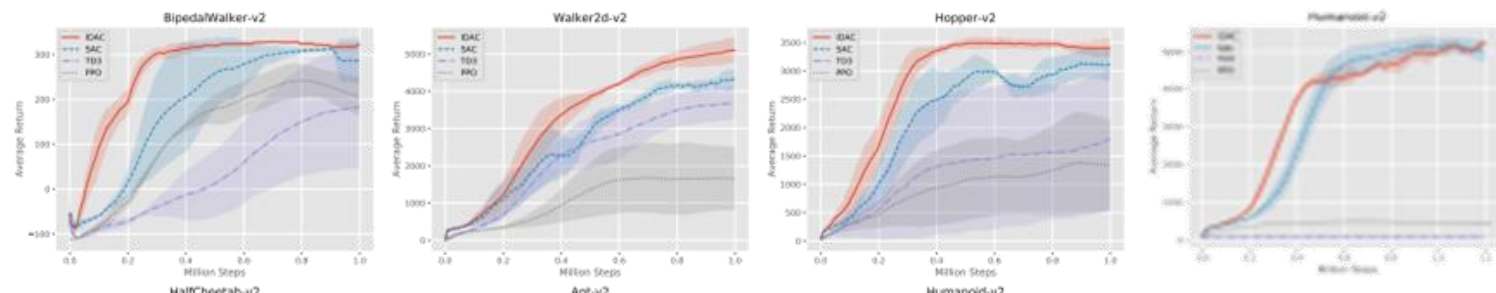
Experiment results (I)

● Comparison with previous methods

- Experiments show the superior performance of IDAC compared to previous methods in BipedalWalker(OpenAI Gym env) and various problems of MuJoCo env(each methods trained with 1M step)
 - IDAC(Implicit Distributional Actor-Critic) 
 - SAC(Soft Actor-Critic) 
 - PPO(Proximal Policy Optimization) 
 - TD3(Twin-Delayed Deep Deterministic Policy Gradient) 
- Policy visualization shows that SIA could capture the correlation in each action dimensions, while Gaussian policy could not
 - Left figure shows relation with dim 1 and 4 of action in Walker-2d env at early training stage



Histogram of randomly sampled actions(1000)



	BipedalWalker	Walker2d	Hopper	HalfCheetah	Ant	Humanoid
PPO	241.79 ± 36.7	1679.39 ± 942.49	1380.68 ± 899.70	1350.37 ± 128.79	141.79 ± 451.10	498.88 ± 20.10
TD3	182.80 ± 135.76	3689.48 ± 434.03	1799.78 ± 1242.63	10209.65 ± 548.14	4905.74 ± 203.09	105.76 ± 53.65
SAC	312.48 ± 2.81	4328.95 ± 249.27	3138.93 ± 299.62	10626.34 ± 73.78	3732.23 ± 602.83	5055.64 ± 62.96
IDAC	328.44 ± 1.23	5107.07 ± 351.37	3497.86 ± 93.30	12222.80 ± 157.15	4930.73 ± 242.78	5233.43 ± 85.87

Return for IDAC and previous methods(1M step)

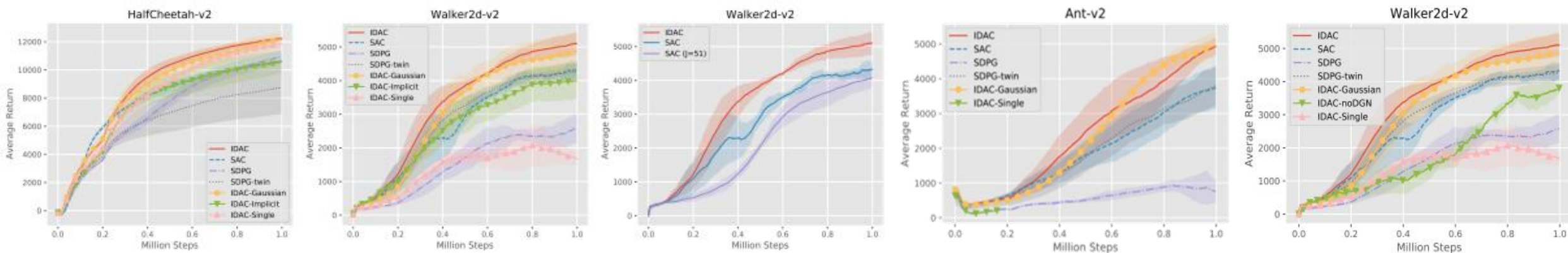
Experiment results (II)

● Ablation study

- Comprehensive set of ablation study was performed to answer the following questions
 - (a) : How important is the type of policy distribution, such as semi-implicit, diagonal Gaussian, deterministic policies?
 - (b) : How much improvement does using distributional critic brings?
 - (c) : How critical is the twin-delayed network?
 - (d) : Will other baselines(such as SAC) benefit from using multiple actions ($J > 1$) for policy gradient estimation?

Table 2: Variants for ablation study.

Ablations	IDAC	SAC	SAC-J ²	SDPG ³	SDPG-Twin	IDAC-Gaussian	IDAC-Implicit	IDAC-Single
Policy dist.	semi-implicit	Gaussian	Gaussian	deterministic	deterministic	Gaussian	implicit	semi-implicit
Distributional	yes	no	no	yes	yes	yes	yes	yes
Twin or single	twin	twin	twin	single	twin	twin	twin	single



Training curves of ablation study

Implementation for IDAC

Environment setting

● Environment

■ LunarLanderContinuous-v2

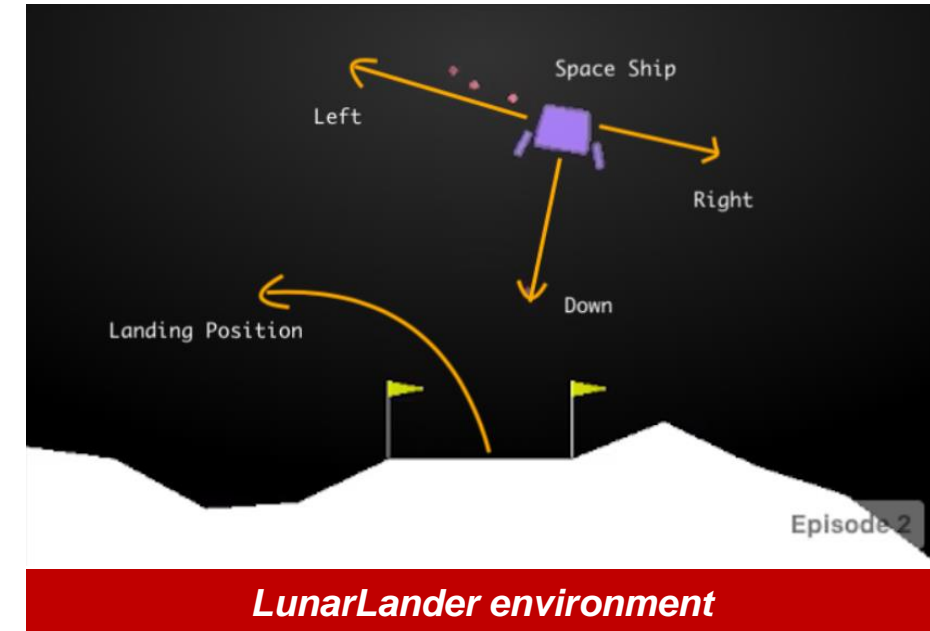
- Well-known environment for continuous control in OpenAI's Gym library(Box2D problem)

■ MDP parameters

- State
 - (8,)
- Action
 - (2,) Main engine(-1~1), direction engine(-1~1)
- Reward
 - Arrival with near-zero speed, +100~140
 - Landing with rocket leg, respectively + 10
 - Landing +100, Crash landing +100
 - Fuel consumption penalty, -0.3 per frame, (infinite fuel)

● Algorithm(with Tensorflow 2)

- IDAC(~ing), IDAC-Gaussian(~ing), TD3(with PER), TD3-gSDE
 - IDAC, IDAC-Gaussian, IDAC-Single
 - TD3 + gaussian noise, TD3 + gSDE



<https://gym.openai.com/envs/LunarLanderContinuous-v2/>
<https://towardsdatascience.com/solving-lunar-lander-openaigym-reinforcement-learning-785675066197>

TD3: <https://arxiv.org/pdf/1802.09477.pdf>
PER: <https://arxiv.org/abs/1511.05952>
gSDE: <https://arxiv.org/abs/2005.05719>

Thank you!

Q&A