RL 스터디

4주차

Soft Actor Critic?

- Off policy를 쓰고 싶다
- Maximum entropy 환경에서 학습시키고 싶다.
- Robust 한 모델을 만들고 싶다.
- ……

- 이러한 모든 점을 만족시키는 것이 SAC

Maximum entropy Framework(MaxEnt)

- 최대 엔트로피 모델은 종분포를 회귀학습 시키는 경우 유명한 방식 중 하나
- 보상과 행동의 엔트로피가 동시에 최대가 되도록 학습한다.

=>

- Otimal 한 정책 주변도 학습을 할 수 있으며 탐색능력을 매우 높일 수 있다.
- 그렇기에 robust한 모델이 되도록 유도한다.

$$\pi^{*}_{\text{MaxEnt}} = \arg\max_{\pi} \sum_{t} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_{\pi}} \left[r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot | \mathbf{s}_t)) \right]$$

기존에는 엔트로피를 어떻게 적용?

- 일반적으로 그라디언트에 엔트로피항을 더해주는 형식 단순히 과적합을 막기 위한 형태로 추가되는 regularization 항과 같은 존재

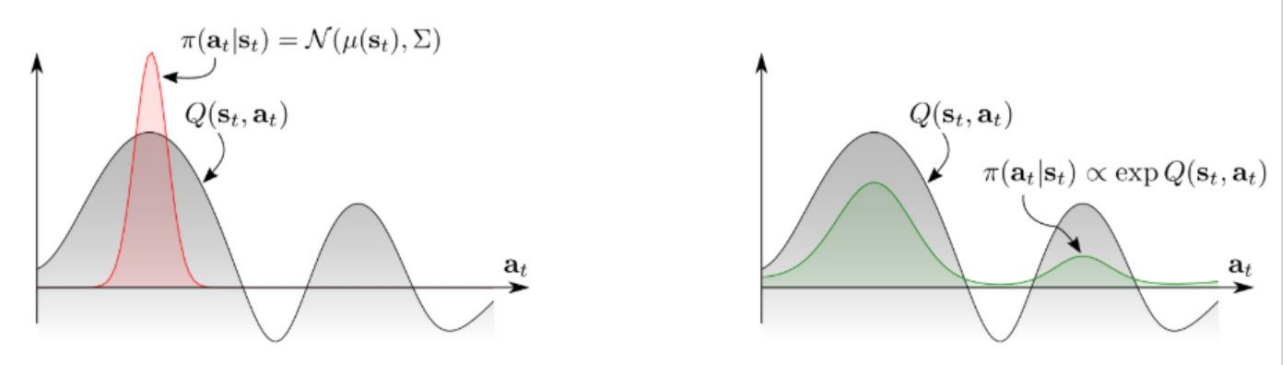
- MaxEnt 를 강화학습에 적용하기 위해 많은 노력들이 있었고 그중 가장 성공적인 방식
- => Soft Q learning

Soft Q learning - energy base model

- MaxEnt 환경을 도입하는데 energy base model을 사용하면 매우 쉽게 그리고 높은 성능으로 적용이 됨을 확인
- energy base model 이란 데이터 간의 의존성을 마치 에너지 형태처럼 나타내는 모델
- 볼쯔만 분포를 통해 확률 분포로 변경할 수 있다.

$$\pi(\mathbf{a}_t|\mathbf{s}_t) \propto \exp\left(-\mathcal{E}(\mathbf{s}_t, \mathbf{a}_t)\right)$$

- 최대 엔트로피도 에너지 기반 모델과 에너지 기반 모델 자체가 매우 유사



Soft Q learning

- Q 함수 <= 에너지 함수라고 보고
- 다음과 같은 식을 만족한다.
- 증명은 논문 생략

Theorem 1. *Let the soft Q-function be defined by*

$$Q^*_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) = r_t + \tag{4}$$

$$\mathbb{E}_{(\mathbf{s}_{t+1},\dots)\sim\rho_\pi}\left[\sum_{l=1}^{\infty}\gamma^l\left(r_{t+l}+\alpha\mathcal{H}\left(\pi^*_{\text{MaxEnt}}(\cdot|\mathbf{s}_{t+l})\right)\right)\right],$$

and soft value function by

$$V^*_{\text{soft}}(\mathbf{s}_t) = \alpha\log\int_{\mathcal{A}}\exp\left(\frac{1}{\alpha}Q^*_{\text{soft}}(\mathbf{s}_t,\mathbf{a}')\right)d\mathbf{a}'. \tag{5}$$

Then the optimal policy for (2) is given by

$$\pi^*_{\text{MaxEnt}}(\mathbf{a}_t|\mathbf{s}_t) = \exp\left(\tfrac{1}{\alpha}\left(Q^*_{\text{soft}}(\mathbf{s}_t,\mathbf{a}_t) - V^*_{\text{soft}}(\mathbf{s}_t)\right)\right). \tag{6}$$

Soft Q learning

◦ 벨만 방정식도 만족

$$Q^*_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) = r_t + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}} \left[V^*_{\text{soft}}(\mathbf{s}_{t+1}) \right]$$

Soft Q learning

◦ 벨만 방정식도 만족

$$Q^*_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) = r_t + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}} \left[V^*_{\text{soft}}(\mathbf{s}_{t+1}) \right]$$

Soft Q iteration

$$Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) \leftarrow r_t + \gamma\, \mathbb{E}_{\mathbf{s}_{t+1} \sim p_{\mathbf{s}}} \left[V_{\text{soft}}(\mathbf{s}_{t+1}) \right], \ \forall \mathbf{s}_t, \mathbf{a}_t$$

$$V_{\text{soft}}(\mathbf{s}_t) \leftarrow \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q_{\text{soft}}(\mathbf{s}_t, \mathbf{a}') \right) d\mathbf{a}', \forall \mathbf{s}_t$$

위 방식은 샘플링이 어려워서 아래 방식대로 TD error를 이용하여 학습

$$J_Q(\theta) = \mathbb{E}_{\mathbf{s}_t \sim q_{\mathbf{s}_t}, \mathbf{a}_t \sim q_{\mathbf{a}_t}} \left[\frac{1}{2} \left(\hat{Q}^{\bar{\theta}}_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) - Q^{\theta}_{\text{soft}}(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \right]$$

$$V^{\theta}_{\text{soft}}(\mathbf{s}_t) = \alpha \log \mathbb{E}_{q_{\mathbf{a}'}} \left[\frac{\exp\left(\frac{1}{\alpha} Q^{\theta}_{\text{soft}}(\mathbf{s}_t, \mathbf{a}') \right)}{q_{\mathbf{a}'}(\mathbf{a}')} \right]$$

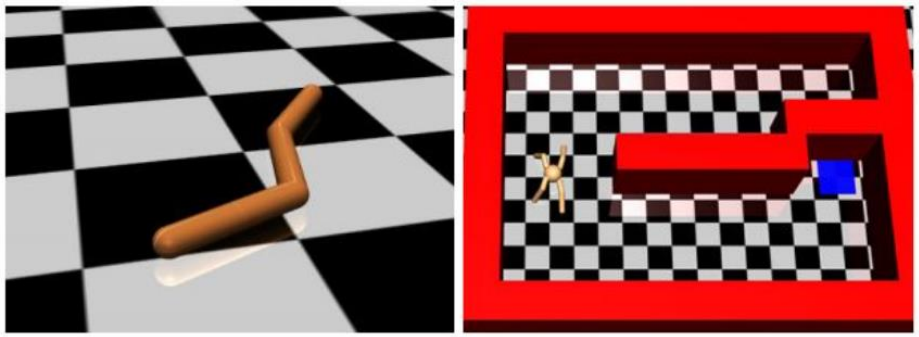
결론

◦ Soft Q를 깊게 수학적으로 설명할 것은 아니다

$$\mathbf{a}_t = f^{\phi}(\xi; \mathbf{s}_t), \text{ parametrized by } \phi.$$

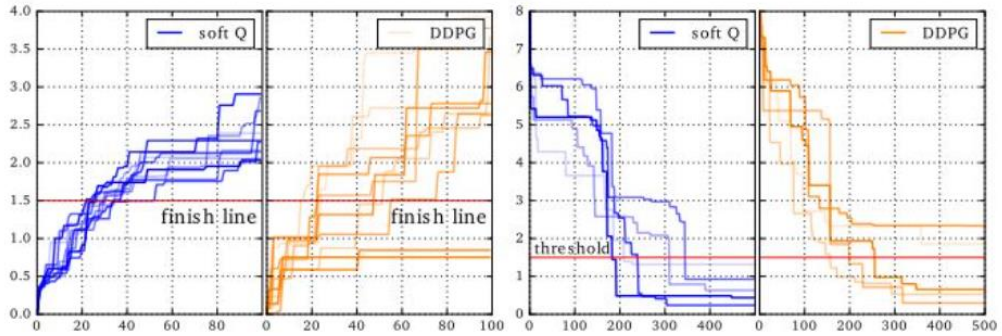
직접 에너지에서 정책을 계산 하는게 아니라 행동함수를 가우시안 분포 위에 맵핑이 되도록 파라미터화 해서 학습시킨다.

성능



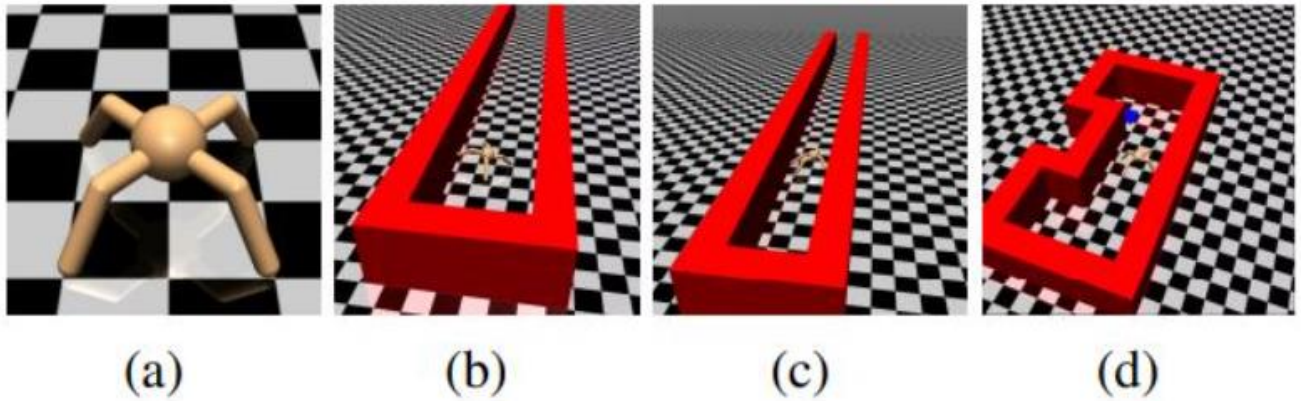
(a) Swimming snake (b) Quadrupedal robot

Figure 2. Simulated robots used in our experiments.

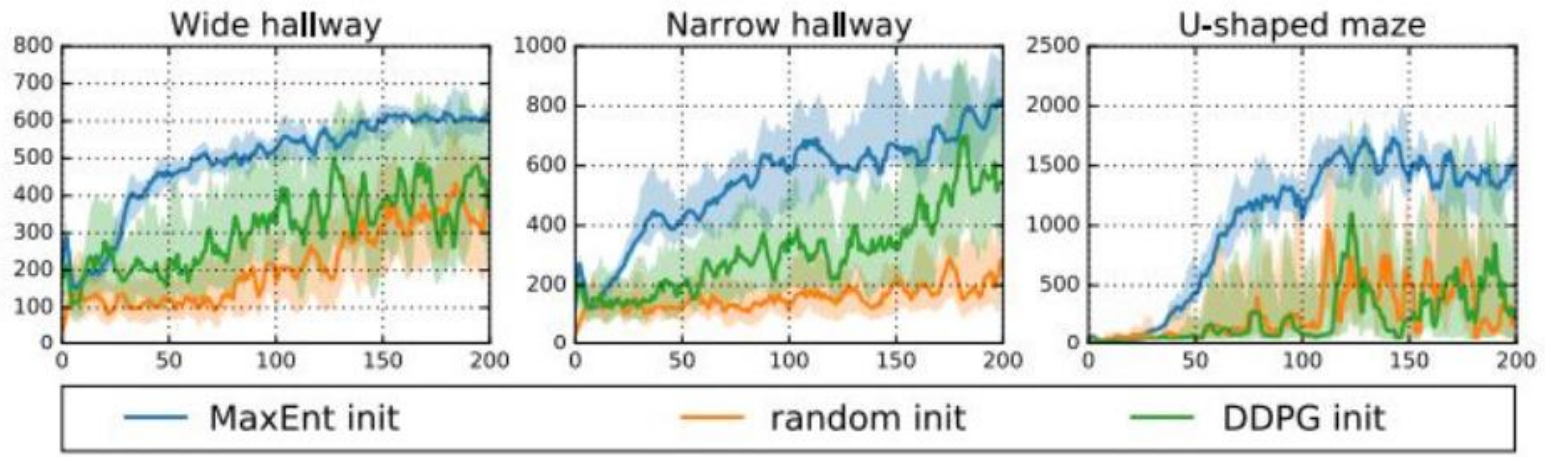


(a) Swimmer (higher is better) (b) Quadruped (lower is better)

성능



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


Wide hallway
Narrow hallway
U-shaped maze
MaxEnt init
random init
DDPG init

SAC – Q 함수 학습

(soft policy evaluation)

$$\mathcal{T}^{\pi} Q(\mathbf{s}_t, \mathbf{a}_t) \triangleq r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \, \mathbb{E}_{\mathbf{s}_{t+1} \sim p} \left[V(\mathbf{s}_{t+1}) \right],$$

where

$$V(\mathbf{s}_t) = \mathbb{E}_{\mathbf{a}_t \sim \pi} \left[Q(\mathbf{s}_t, \mathbf{a}_t) - \log \pi(\mathbf{a}_t | \mathbf{s}_t) \right]$$

$$\pi_{\text{new}} = \arg \min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}} \left(\pi'(\cdot | \mathbf{s}_t) \,\middle\|\, \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(\mathbf{s}_t, \cdot) \right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)} \right)$$

SAC – Q 함수 학습

- 초기 SAC 에서는 V를 따로 근사했는데 이후는 Q로 쉽게 V를 구할 수 있어서 Q 만 근사.
- 아래와 같이 Q 함수를 계산한다.

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

$$\hat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathbf{a}_t, \mathbf{s}_t) \left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \left(r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \left(Q_{\bar{\theta}}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) - \alpha \log \left(\pi_\phi(\mathbf{a}_{t+1} | \mathbf{s}_{t+1}) \right) \right) \right) \right)$$

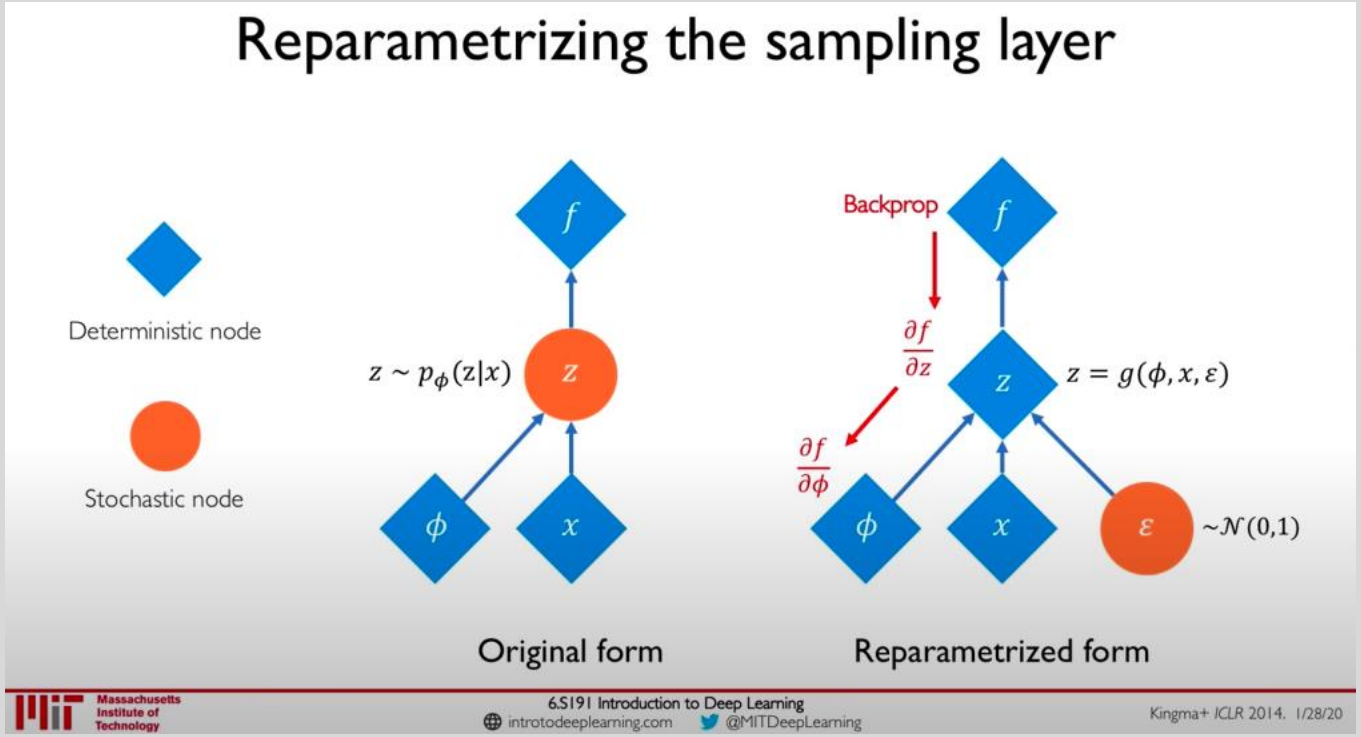
SAC - 목적 함수

$$\pi_{\text{new}} = \arg\min_{\pi' \in \Pi} \mathrm{D}_{\mathrm{KL}} \left(\pi'(\cdot|\mathbf{s}_t) \,\middle\|\, \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(\mathbf{s}_t, \cdot)\right)}{Z^{\pi_{\text{old}}}(\mathbf{s}_t)} \right)$$

$$\begin{aligned} D_{KL}(P||Q) &= \mathbb{E}_{x\sim P}[-\log Q(x)] - \mathbb{E}_{x\sim P}[-\log P(x)] \\ &= \mathbb{E}_{x\sim P}[-\log Q(x) - (-\log P(x))] \\ &= \mathbb{E}_{x\sim P}[-\log Q(x) + \log P(x)] \\ &= \mathbb{E}_{x\sim P}[\log P(x) - \log Q(x)] \\ &= \mathbb{E}_{x\sim P}[\log \frac{P(x)}{Q(x)}] \end{aligned}$$

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} \left[\mathbb{E}_{\mathbf{a}_t \sim \pi_\phi} \left[\alpha \log\left(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)\right) - Q_\theta(\mathbf{s}_t, \mathbf{s}_t) \right] \right]$$

Reparameterization Trick



SAC

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}} \left[\log \pi_\phi(f_\phi(\epsilon_t; \mathbf{s}_t) | \mathbf{s}_t) - Q_\theta(\mathbf{s}_t, f_\phi(\epsilon_t; \mathbf{s}_t)) \right]$$

$$\begin{aligned} \hat{\nabla}_\phi J_\pi(\phi) = & \nabla_\phi \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t) \\ & + (\nabla_{\mathbf{a}_t} \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t) - \nabla_{\mathbf{a}_t} Q(\mathbf{s}_t, \mathbf{a}_t)) \nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) \end{aligned}$$

SAC – alpha 조정

$$\max_{\pi_{0:T}} \mathbb{E}_{\rho_\pi} \left[\sum_{t=0}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right] \text{ s.t. } \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[-\log(\pi_t(\mathbf{a}_t | \mathbf{s}_t)) \right] \geq \mathcal{H} \ \ \forall t$$

$$\mathbb{E}_{(\mathbf{s}_T, \mathbf{a}_T) \sim \rho_\pi} \left[-\log(\pi_T(\mathbf{s}_T | \mathbf{s}_T)) \right] \geq \mathcal{H}$$

$$\max_{\pi_T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathbf{s}_T, \mathbf{a}_T) \right] = \min_{\alpha_T \geq 0} \max_{\pi_T} \mathbb{E} \left[r(\mathbf{s}_T, \mathbf{a}_T) - \alpha_T \log \pi(\mathbf{a}_T | \mathbf{s}_T) \right] - \alpha_T \mathcal{H},$$

SAC – alpha 조정

$$\alpha_t^* = \arg\min_{\alpha_t} \mathbb{E}_{\mathbf{a}_t \sim \pi_t^*} \left[-\alpha_t \log \pi_t^*(\mathbf{a}_t | \mathbf{s}_t; \alpha_t) - \alpha_t \bar{\mathcal{H}} \right]$$

$$J(\alpha) = \mathbb{E}_{\mathbf{a}_t \sim \pi_t} \left[-\alpha \log \pi_t(\mathbf{a}_t | \mathbf{s}_t) - \alpha \bar{\mathcal{H}} \right]$$

최종 알고리즘

Algorithm 1 Soft Actor-Critic

Input: θ_1, θ_2, ϕ ▷ Initial parameters
$\bar{\theta}_1 \leftarrow \theta_1, \bar{\theta}_2 \leftarrow \theta_2$ ▷ Initialize target network weights
$\mathcal{D} \leftarrow \emptyset$ ▷ Initialize an empty replay pool
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$ ▷ Sample action from the policy
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ ▷ Sample transition from the environment
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$ ▷ Store the transition in the replay pool
  end for
  for each gradient step **do**
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$ ▷ Update the Q-function parameters
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$ ▷ Update policy weights
    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J(\alpha)$ ▷ Adjust temperature
    $\bar{\theta}_i \leftarrow \tau\theta_i + (1-\tau)\bar{\theta}_i$ for $i \in \{1, 2\}$ ▷ Update target network weights
  end for
end for
Output: θ_1, θ_2, ϕ ▷ Optimized parameters

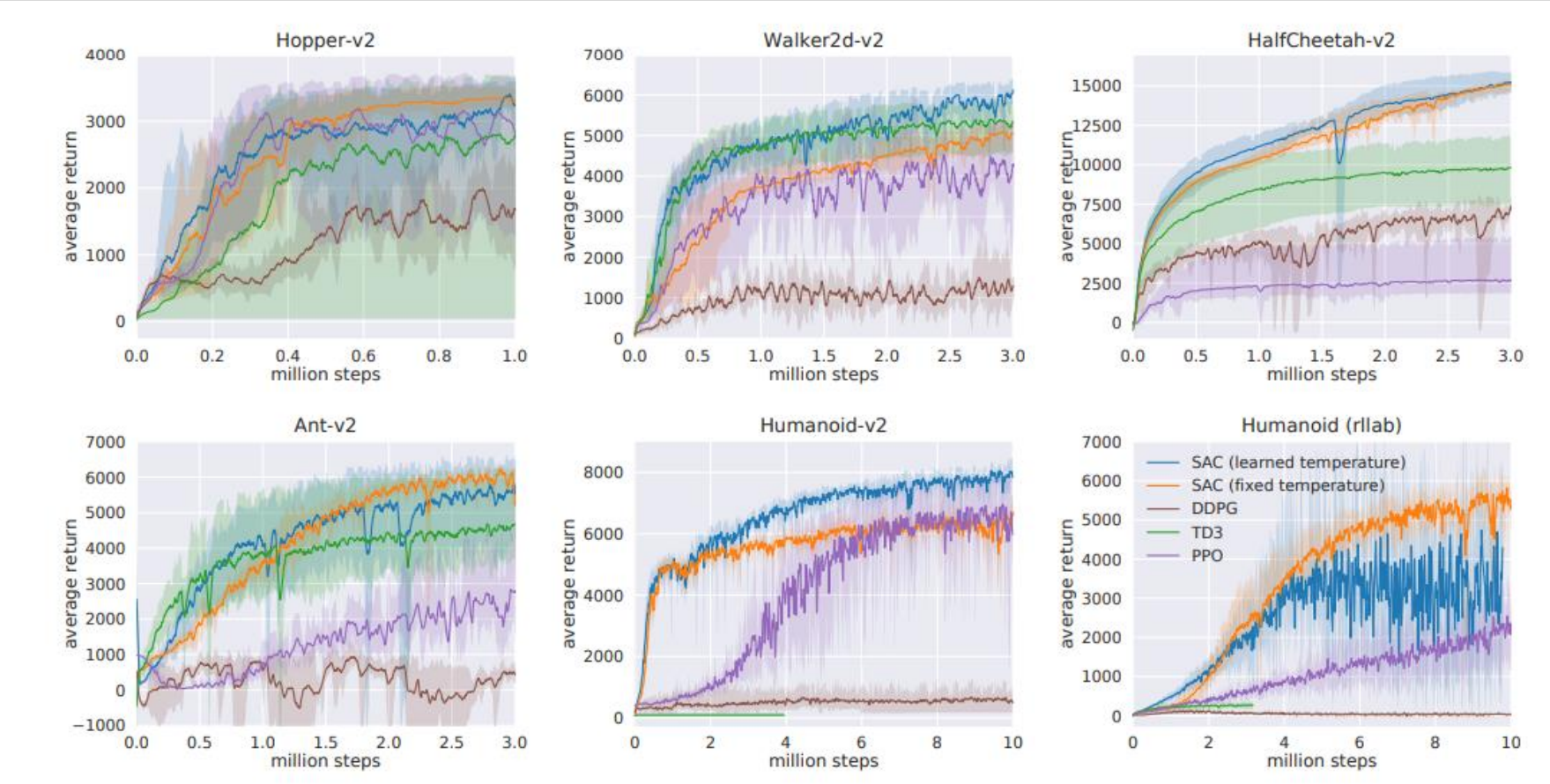
구현 이슈

- TD3에서 영향을 받아 두개의 Q를 학습, 작은 Q를 선택하는 방식을 도입
- Reward scale이 너무 작은 경우 또는 큰 경우 학습이 잘 안됨
- 정책 학습에는 확률적으로 선택을 하지만 최종적으로는 결정론적으로 바꾸어 평가함
- Target 네트워크로 전이를 너무 빠르게 가져가면 안됨

Table 1: SAC Hyperparameters

Parameter	Value
optimizer	Adam (Kingma & Ba, 2015)
learning rate	$3 \cdot 10^{-4}$
discount (γ)	0.99
replay buffer size	10^6
number of hidden layers (all networks)	2
number of hidden units per layer	256
number of samples per minibatch	256
entropy target	$-\dim(\mathcal{A})$ (e.g. , -6 for HalfCheetah-v1)
nonlinearity	ReLU
target smoothing coefficient (τ)	0.005
target update interval	1
gradient steps	1

성능


Hopper-v2
Walker2d-v2
HalfCheetah-v2
Ant-v2
Humanoid-v2
Humanoid (rllab)
average return
million steps
SAC (learned temperature)
SAC (fixed temperature)
DDPG
TD3
PPO

-10-04

추가 – 오목 구현

- 알파제로 계열을 사용하지 않고 오목을 구현하는 방식

입력 데이터 – Conv 데이터

- 자기 말만 1로 표시한 데이터 + 상대 말 만 1로 표시한 데이터
- 시간 순 별로 이전 몇 개의 데이터를 넣어주는 것도 좋음
- 지금 두는 말의 위치를 표현하는 데이터
- 내가 선인지 말인지 선택하는 데이터 <= ex) 전부 0이거나 1이거나

보상처리

- 승리시 1 패배시 -1 => 최종에만 보상을 줄 수도 몬테카를로 방식처럼 한 게임을 끝내고 전체 행동에 대해 보상을 처리 할 수도 있으

데이터 학습

- 오목 데이터를 넣어서 학습 시킬 수도
- 자가 플레이를 하면서 데이터를 생성해서 학습 할 수도

휴리스틱 알고리즘

- 완전히 ai 에 의존하는 것보다
- 예를 들어 한 줄이 3개가 되는 것을 발견 했다 => 무조건 둘 수 있는 공간을 3줄 양쪽으로 한정을 지어버린다.

- 이런 식의 휴리스틱한 알고리즘이 들어가면 좋다.