HiPPO

Sub-Policy Adaptation for Hierarchical Reinforcement Learning

홍성학

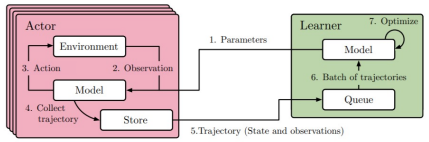
Index

- Introduction
- Paper Review
  - Abstract
  - Key Concepts
  - Experiments

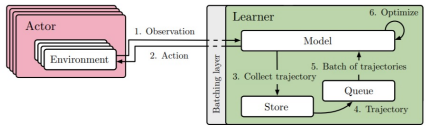
Introduction

- 인공지능연구실에서 지능형 로봇 프레임워크 연구 진행
- 국방 M&S 회사 전문연구요원으로 입사 후 국방 분야 강화학습 연구 진행
  - 분산강화학습 프레임워크 연구
  - 다수 드론 군집 임무 강화학습 환경 & 모델 연구
  - 전투기 교전 강화학습 모델 연구
- 게임회사 이직 이후 게임도메인 강화학습 연구 진행
  - Multi-agent PvP 환경에서의 강화학습 모델 연구

Introduction

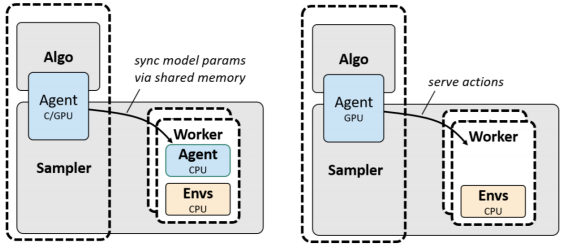


(a) IMPALA architecture (distributed version)

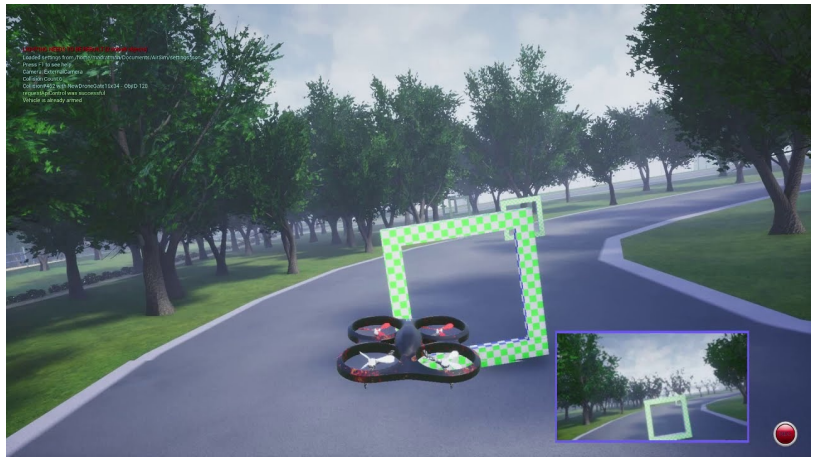


(b) SEED architecture, see detailed replay architecture in Figure 3.

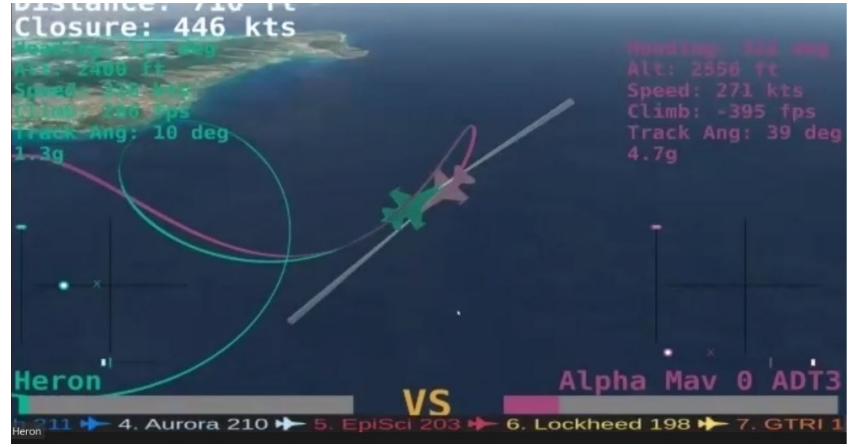
IMPALA / SEEDRL



rlpyt



AirSIM



Alpha Dogfight

분산강화학습 프레임워크 연구

다수 드론 군집 임무 강화학습 환경
& 모델 연구

전투기 교전 강화학습 모델 연구

Abstract

- Sub-Policy Adaptation for Hierarchical Reinforcement Learning
  (ICLR 2020.03, UC Berkeley(Pieter abbeel)
- Reinforcement Learning 에서 모듈 간 추상화와 재사용 등에 대한 한계를 극복하기 위한 방법으로 HRL 연구
- Temporal hierarchical 을 위한 framework 및 policy gradient 근사 식 유도
- High-level policy 와 low-level policy 간 decoupling 과 sub-policy baseline 활용이 가능한 HiPPO(Hierarchical Proximal Optimization) 알고리즘 제시

Key Concepts

- Discrete-time finite-horizon discounted MDP 제안

$M = (S, A, P, r, P_0, \gamma, H)$
$S : state, A : action, P : S * A * S \rightarrow R_+, \gamma : discount\ factor, H : time\ horizon$

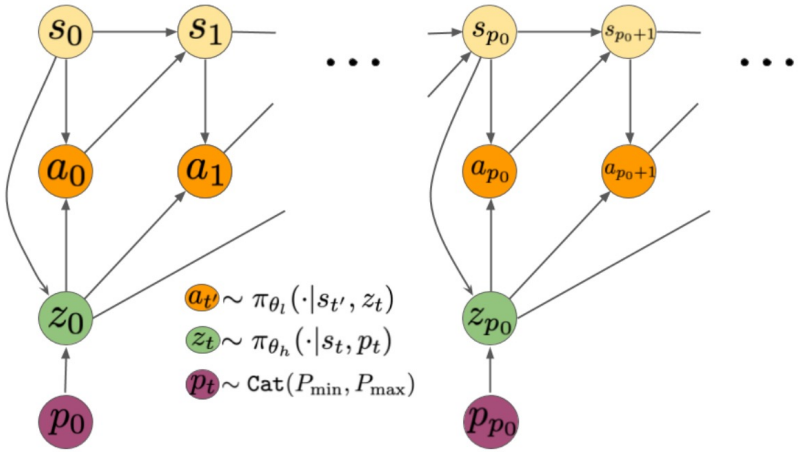


Figure 1: Temporal hierarchy studied in this paper. A latent code z_t is sampled from the manager policy $\pi_{\theta_h}(z_t|s_t)$ every p time-steps, using the current observation s_{kp}. The actions a_t are sampled from the sub-policy $\pi_{\theta_l}(a_t|s_t, z_{kp})$ conditioned on the same latent code from $t = kp$ to $(k+1)p - 1$

- High-level policy = $\pi_{\theta_h}(z_t|s_t)$ n-sub policies 혹은 skill 실행을 위한 latent code Z 결정
- Low-level policy = $\pi_{\theta_l}(a_t|s_t, z_{kp})$ state, Z 에 대한 low level action 결정

Key Concepts

- Proximal Policy Optimization (PPO, 2017, OpenAI)
  - Policy parameter theta 의 변화가 action의 큰 변화를 가져올 수 있기 때문에, theta 를 안정적으로 update 할 수 있도록 하는 알고리즘

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

Old policy　　New policy

  - update 할 policy parameter 가 기존 policy parameter 기준으로 크게 변화하지 않도록 new policy 를 clipping 한 뒤, old policy-new policy 중 작은 값으로 update하여 lower bound 형성
  - PPO는 전신인 TRPO의 식을 구현관점에서 개선하면서도 성능 유지
  - 구현이 간단하며 성능이 좋아 새로운 환경, 문제에서 Baseline으로 활용

Key Concepts

- HiPPO Loss Function

$$L^{CLIP}_{HiPPO}(\theta) = \mathbb{E}_\tau \Big[\sum_{k=0}^{H/p} \min \left\{ w_{h,kp}(\theta) A(s_{kp}, z_{kp}), w^{\texttt{clip}}_{h,kp}(\theta) A(s_{kp}, z_{kp}) \right\} + \sum_{t=0}^{H} \min \left\{ w_{l,t}(\theta) A(s_t, a_t, z_{kp}), w^{\texttt{clip}}_{l,t}(\theta) A(s_t, a_t, z_{kp}) \right\} \Big]$$

$$w_{h,kp}(\theta) = \frac{\pi_{\theta_h}(z_{kp}|s_{kp})}{\pi_{\theta_{h,old}}(z_{kp}|s_{kp})} \qquad w_{l,t}(\theta) = \frac{\pi_{\theta_l}(a_t|s_t, z_{kp})}{\pi_{\theta_{l,old}}(a_t|s_t, z_{kp})}$$

- From PPO Loss Function

$$L^{CLIP}(\theta) = \mathbb{E}_t \min \left\{ w_t(\theta) A_t,\ \texttt{clip}(w_t(\theta), 1-\epsilon, 1+\epsilon) A_t \right\} \quad w_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

Key Concepts

- Rollout and Policy update sudo code
  - 대부분의 HRL 에서 low-level skill에 대해 fixed time-commitment 를 부여하거나 option framework 활용
  - HiPPO 에서는 fixed distribution time 구간 중 무작위 시간 sampling

Algorithm 1 `HiPPO Rollout`

Input: skills $\pi_{\theta_l}(a|s,z)$, manager $\pi_{\theta_h}(z|s)$, time-commitment bounds $P_{\min}$ and $P_{\max}$, horizon H
Reset environment: $s_0 \sim \rho_0, t = 0$.
while $t < H$ **do**
  Sample time-commitment $p \sim \texttt{Cat}([P_{\min}, P_{\max}])$
  Sample skill $z_t \sim \pi_{\theta_h}(\cdot|s_t)$
  for $t' = t \dots (t+p)$ **do**
    Sample action $a_{t'} \sim \pi_{\theta_l}(\cdot|s_{t'}, z_t)$
    Observe new state $s_{t'+1}$ and reward $r_{t'}$
  end for
  $t \leftarrow t + p$
end while
Output: $(s_0, z_0, a_0, s_1, a_1, \dots, s_H, z_H, a_H, s_{H+1})$

Algorithm 2 HiPPO

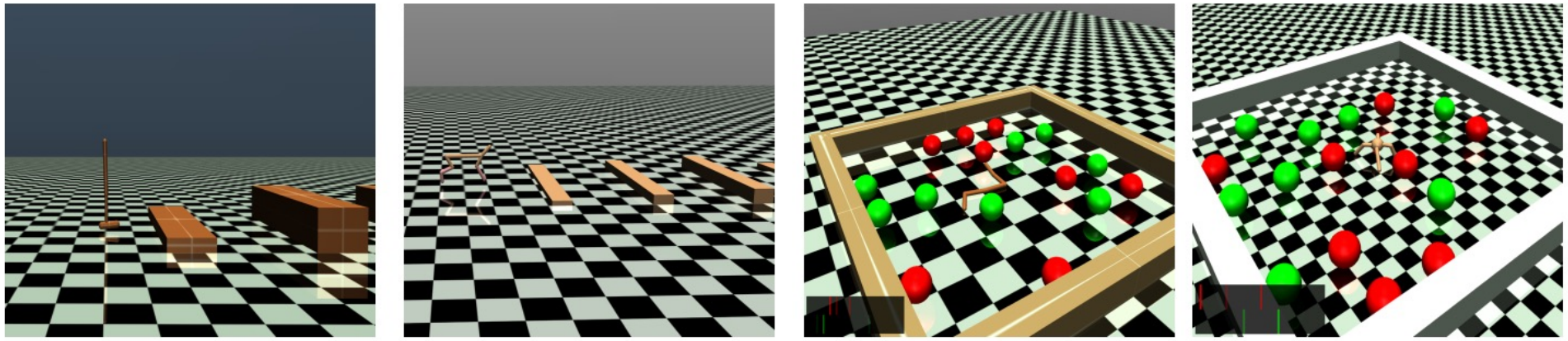
Input: skills $\pi_{\theta_l}(a|s,z)$, manager $\pi_{\theta_h}(z|s)$, horizon H, learning rate α
while not done **do**
  for actor = 1, 2, ..., N **do**
    Obtain trajectory with `HiPPO Rollout`
    Estimate advantages $\hat{A}(a_{t'}, s_{t'}, z_t)$ and $\hat{A}(z_t, s_t)$
  end for
  $\theta \leftarrow \theta + \alpha \nabla_\theta L^{CLIP}_{HiPPO}(\theta)$
end while

Experiments

- Scratch 학습 시 flat PPO 와 비교하여 HiPPO 성능 비교
- 학습 Environment 의 물리적 변경에 대한 Policy Robustness 검증
- 미리 학습 된 Skill 에 대한 적응 검증

Experiments

- Robotic locomotion 과 navigation task (continuous) 환경에서 실험 진행



(a) Block Hopper (b) Block Half Cheetah (c) Snake Gather (d) Ant Gather

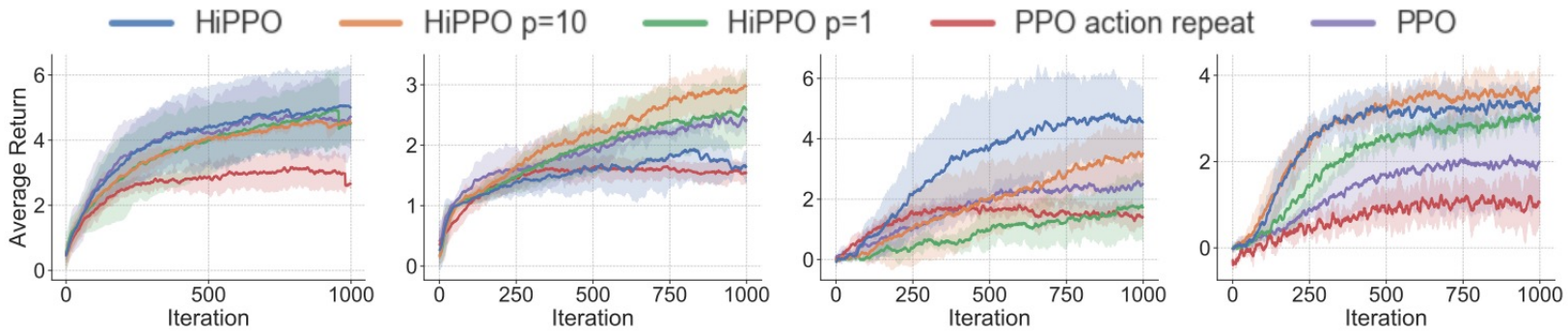
Figure 2: Environments used to evaluate the performance of our method. Every episode has a different configuration: wall heights for (a)-(b), ball positions for (c)-(d)

Hopper / Half-Cheetah robot 으로
랜덤 높이의 벽을 점프하면서 보행 학습

Lidar-type sensor 를 가지고
bomb(red ball)을 피해
apple(green ball)을 모으는 학습

Experiments

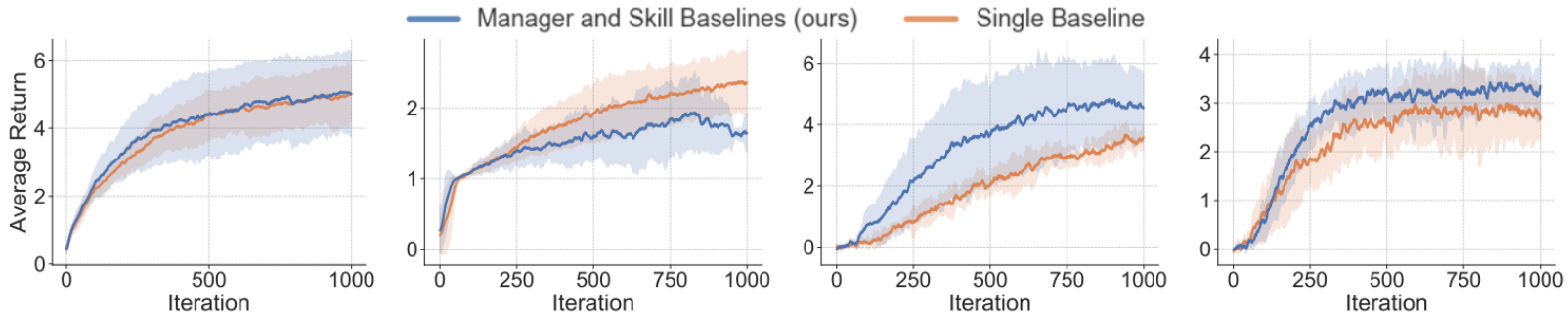
- Scratch 학습 시 HiPPO가 flat PPO 보다 좋은 성능을 보임



(a) Block Hopper (b) Block Half Cheetah (c) Snake Gather (d) Ant Gather

Figure 3: Analysis of different time-commitment strategies on learning from scratch.

- Skill-conditioned baseline 을 사용했을 때 단일 baseline 을 사용한 것 보다 좋은 성능을 보임

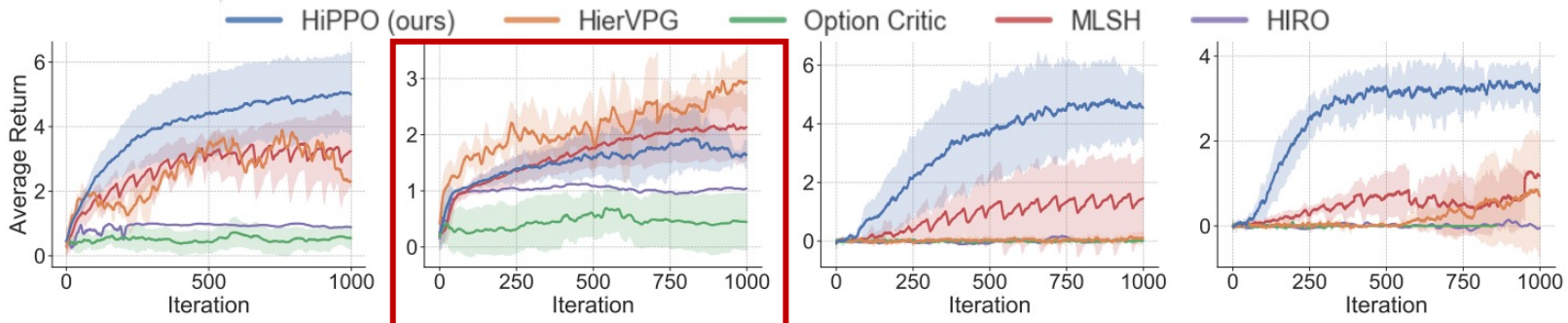


(a) Block Hopper (b) Block Half Cheetah (c) Snake Gather (d) Ant Gather

Figure 4: Using a skill-conditioned baseline, as defined in Section 4.2, generally improves performance of HiPPO when learning from scratch.

Experiments

- 대표적인 HRL 구조인 HIRO, Option-Ciritic, MLSH, HierVPG 보다 좋은 성능을 보임



(a) Block Hopper (b) Block Half Cheetah (c) Snake Gather (d) Ant Gather

Figure 5: Comparison of HiPPO and HierVPG to prior hierarchical methods on learning from scratch.

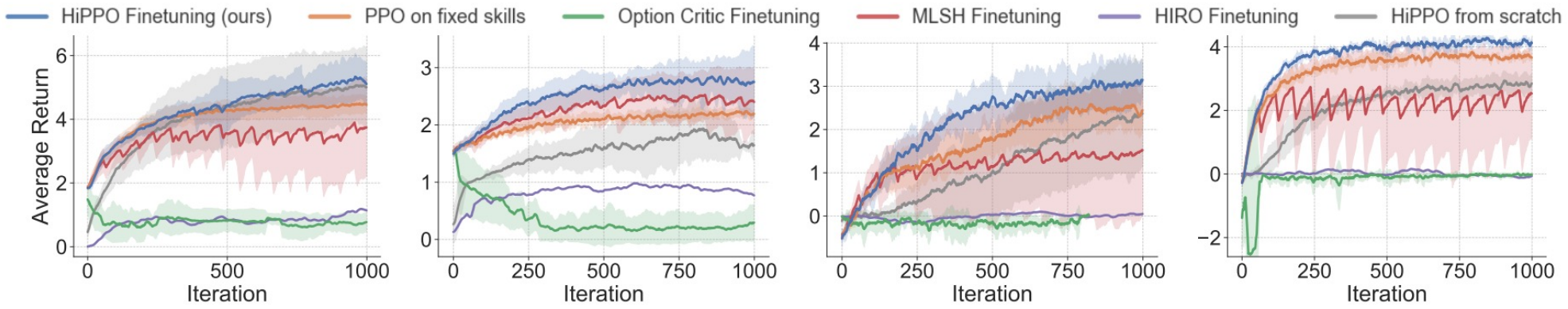
- Snake/Ant Gather 환경의 물리적 특성 변경을 통한 robustness 검증 시 6/8 task에서 좋은 성능

Gather	Algorithm	Initial	Mass	Dampening	Inertia	Friction
Snake	Flat PPO	2.72	3.16 (+16%)	2.75 (+1%)	2.11 (-22%)	2.75 (+1%)
	HiPPO, $p = 10$	4.38	3.28 (-25%)	3.27 (-25%)	3.03 (-31%)	3.27 (-25%)
	HiPPO random p	5.11	**4.09** (-20%)	**4.03** (-21%)	**3.21** (-37%)	**4.03** (-21%)
Ant	Flat PPO	2.25	2.53 (+12%)	2.13 (-5%)	2.36 (+5%)	1.96 (-13%)
	HiPPO, $p = 10$	3.84	3.31 (-14%)	**3.37** (-12%)	2.88 (-25%)	**3.07** (-20%)
	HiPPO random p	3.22	**3.37** (+5%)	2.57 (-20%)	**3.36** (+4%)	2.84 (-12%)

Table 1: Zero-shot transfer performance. The final return in the initial environment is shown, as well as the average return over 25 rollouts in each new modified environment.

Experiments

- Pre-trained sub-policies 를 가진 HiPPO가 skill를 고정한 PPO나 HiPPO from scratch 보다 좋은 성능을 보임



(a) Block Hopper (b) Block Half Cheetah (c) Snake Gather (d) Ant Gather

Figure 6: Benefit of adapting some given skills when the preferences of the environment are different from those of the environment where the skills were originally trained. Adapting skills with HiPPO has better learning performance than leaving the skills fixed or learning from scratch.

감사합니다

Q&A