Accelerating Reinforcement Learning with Learned Skill Priors

Karl Pertsch, Youngwoon Lee, Joseph J. Lim
University of Southern California

4th Conference on Robot Learning (CoRL 2020)

Project Page | Paper

Learning Agents 홍준형 2025.03.13

Top conferences in AI and robotics

	Organizing Institution	Since [Year]	Review Process	Open Review	Avg. Acceptance Rate (last 10yr)	Track	Month of Annual Conf.
ICRA (2025)	IEEE RAS (Robotics and Automation Society)	1984	Single-blind	No	around 42%	Multi-track	May
IROS (2025)	IEEE RAS & RSJ (Robotics Society of Japan)	1988	Single-blind	No	around 46%	Multi-track	October
RSS (2025)	RSS Foundation	2005	Double-blind	No	around 31%	Single-track	June
CoRL (2025)	RLF (Robot Learning Foundation)	2017	Double-blind	Yes	around 36% (8 years avg. no data for 17/18)	Single-track	September

https://www.youtube.com/watch?v=XItdLkDWzH0

Motivation

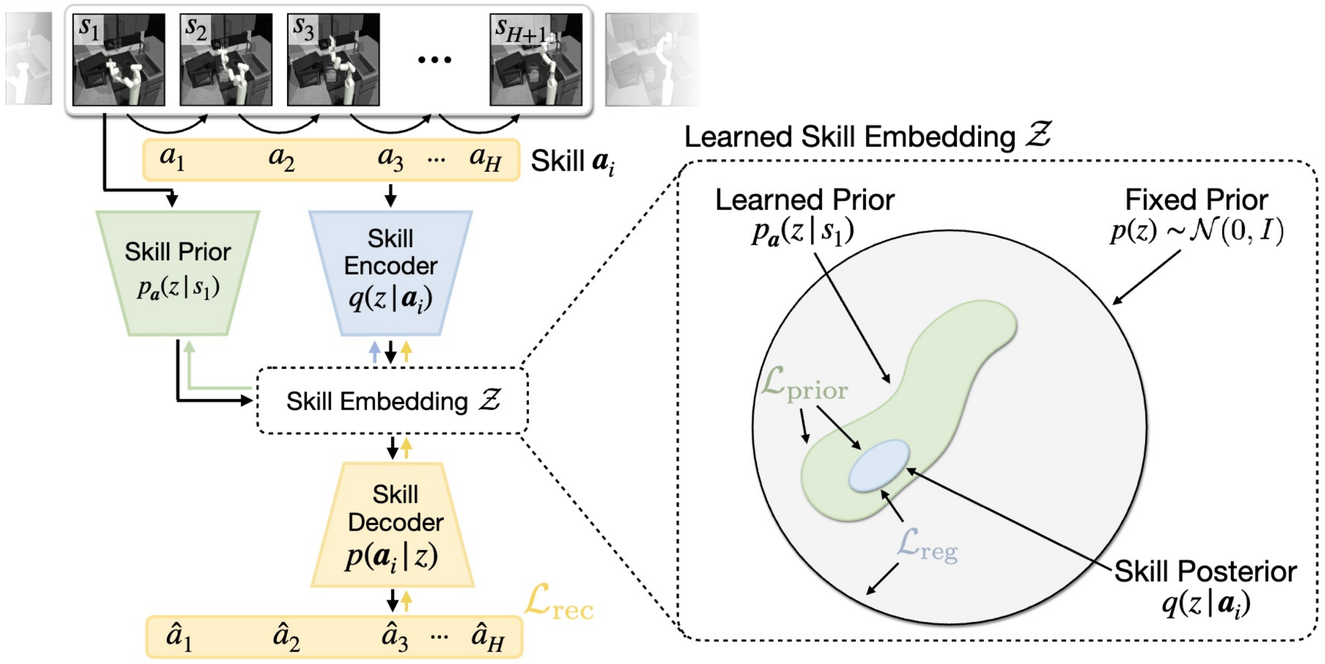
- 사람은 새로운 task를 배울 때 Prior experience를 이용하는데 RL을 이용해 Downstream task를 학습 할 때도 Prior experience를 활용해 효율적으로 학습하고싶다.
- Q) 어떻게 활용 할 것인가? A) Prior experience에서 Skill을 추출해내자
- Prior experience에서 얻어진 Skill들을 활용할 때 Skill 너무 많아진다면 Skill space에서도 많은 exploration을 해야하기 때문에 학습에 어려움이 있다. **하지만** 사람은 새로운 task를 배울 때 모든 Skill을 고려하지 않는다. 왜? Downstream Task에 유망한 Skill이 존재하므로(Skill prior) 거기서 Skill을 골라보도록 하자!
- 또한 dataset이 커질 경우 Skill이 너무 많아져 학습이 어려워지는데 이럴 경우에도 잘하고 싶다.

New Task

what is the most promising skill when if we learn new task?

Overview

- 1. 주어진 Dataset을 이용해 Skill Embedding과 Skill Prior를 학습
- 2. SAC를 이용해 Skill Prior에 Regularized 된 Policy를 학습



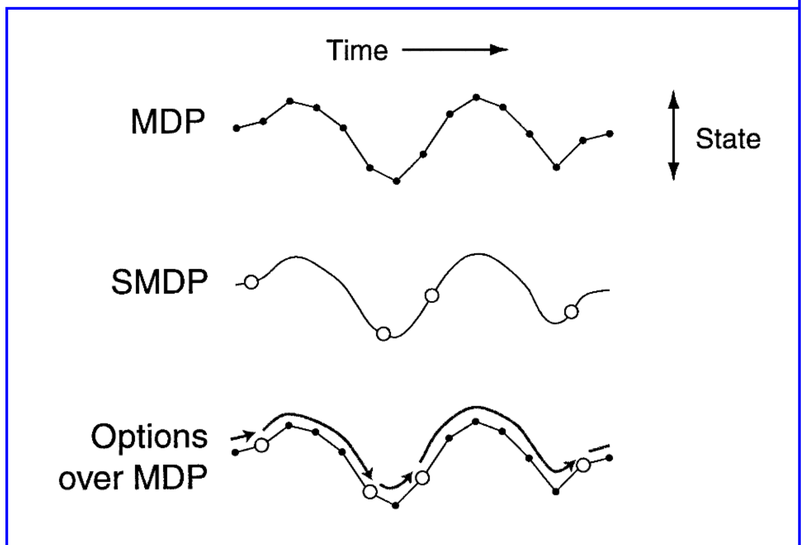
Hierarchical Reinforcement Learning

option : sequence of actions. $< I, \beta, \pi >$로 구성됨.

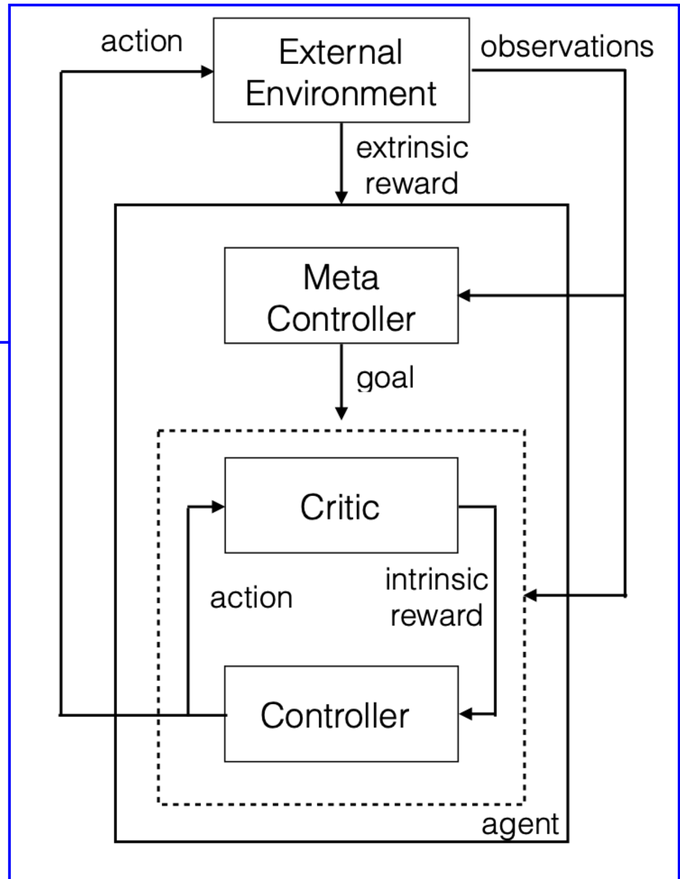
- Meta Controller : option(skill, goal)을 결정하는 agent 흔히 High-Level Policy라고 부름
- Controller : 환경에 적용 할 action을 결정하는 agent 흔히 Low-Level Policy라고 부름

Advantage

- **Temporal abstraction(시간적 추상화)**
- **Reusable**
- **Meaningful state abstract**



SMDP , H-DQN



Skill Prior learning with VAE

- Latent Variable Model

- Learning Objective

$$\log p(\boldsymbol{a}_i) \geq \mathbb{E}_q \Big[\underbrace{\log p(\boldsymbol{a}_i|z)}_{\text{reconstruction}} - \beta \big(\underbrace{\log q(z|\boldsymbol{a}_i) - \log p(z)}_{\text{regularization}} \big) \Big].$$

- VAE를 통해서 prior($p(z)$)에 regularize 된 skill의 embedding space를 학습 할 수 있다.
- 본 논문에서는 항상 일정한 길이의 action sequences를 Skill로 정의함 $\mathbf{a_i} = \{a_t^i, a_{t+1}^i, \ldots, a_{t+H-1}^i\}$

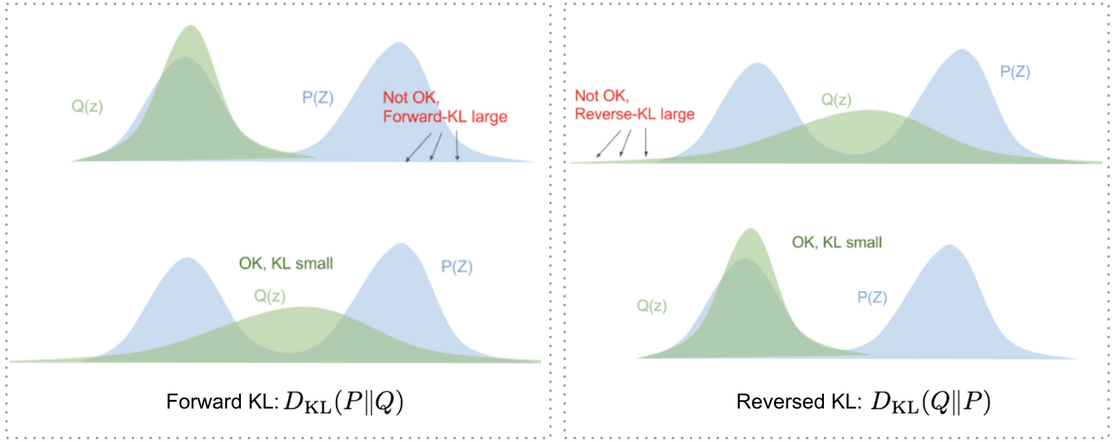
Skill Prior learning with VAE

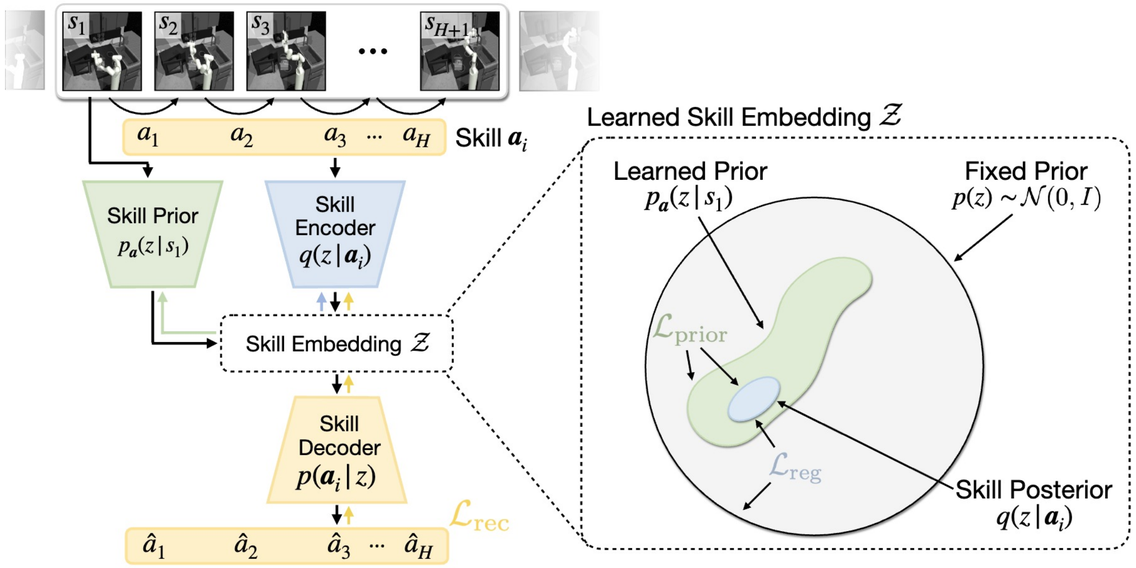
- Z는 Skill embedding space로 skill 그 자체로 생각해도 되며 vector space이기 때문에 많은 skill을 학습 가능하다.

- 그렇다면 skill prior는 어떻게 정의 할 것인가?
  논문에서는 $p_{\mathbf{a}}(z|s_t)$으로 정의하지만 s_t 자리에 task를 푸는데 있어 적절한 무언가로 상황에 맞게 적절하게 바꿀 수 있다고 한다.

- Skill prior learning objective : $\mathbb{E}_{(s,\boldsymbol{a}_i)\sim\mathcal{D}} D_{\mathrm{KL}}\big(q(z|\boldsymbol{a}_i), p_{\boldsymbol{a}}(z|s_t)\big)$.

- Skill Posterior의 전역적인 부분을 prior가 커버 할 수 있도록 forward KL을 이용함





Learning from Offline Data

- SPiRL은 offline data로 학습 할 때 조금 더 널널한 data 형태를 필요로 한다.

	reward labeling	Expert	Sorted Data
SPiRL	X	X	O
Offline RL	O	X	X
BC	X	O	X

Skill Prior Regularized RL

- SPiRL 목적은 Offline Data를 이용해서 RL을 효율 적으로 하는 것

그럼 어떻게 Skill Prior를 이용하면서 RL을 해야할까?

- Max Ent RL : $max\mathbb{E}_{\pi}[\sum_{t=0}\gamma^{t}r_{t}+\alpha H(\pi(a_{t}|s_{t}))]$
  단순한 $maxH(\pi)$은 $minKL(\pi|U)$ 과 같으므로
  $max\mathbb{E}_{\pi}[\sum_{t=0}\gamma^{t}r_{t}-\alpha KL(\pi(a_{t}|s_{t})|U(a_{t}))]$로 나타날 수 있다.

- 우리가 원하는 것은 Policy가 Skill Prior를 적당히 고려하는 것이므로 SPiRL의 목적식은 아래와 같다.

$$max\mathbb{E}_{\pi}[\sum_{t=0}\gamma^{t}\tilde{r}_{t}-\alpha KL(\pi(z_{t}|s_{t})\parallel p_{\mathbf{a}}(z_{t}|s_{t}))]$$

Skill Prior Regularized RL

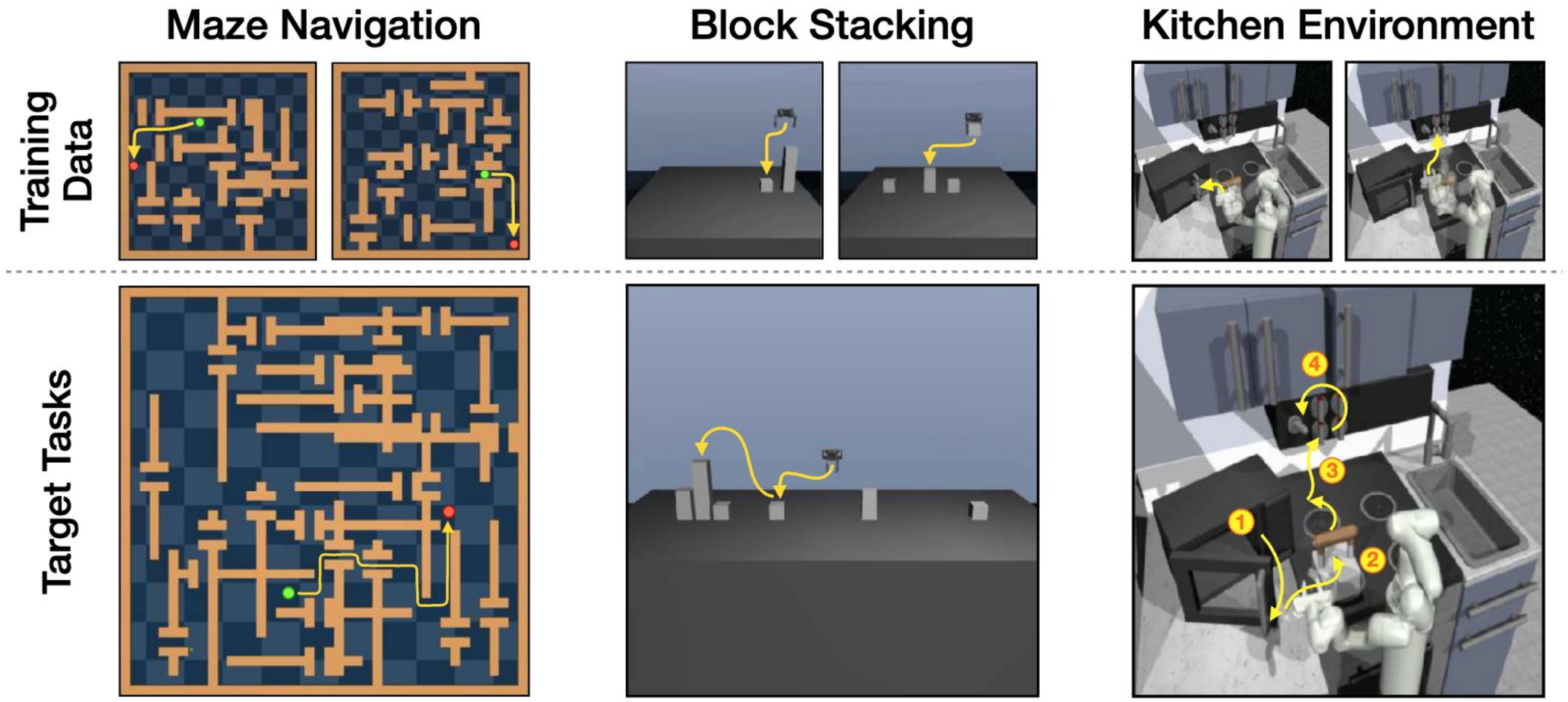
$$\tilde{r}_t = \sum_{k=0}^{H-1} r_{t+k}, s_{t+H} \sim p(s_{t+H}|s_t, z_t)$$

이와 같이 고정된 H step 만큼의 transition과 reward를 고려한다면 high level policy에 대해 model free RL 적용 가

Algorithm 1 SPiRL: Skill-Prior RL

Inputs: H-step reward function $\tilde{r}(s_t, z_t)$, discount γ, target divergence δ, learning rates $\lambda_\pi, \lambda_Q, \lambda_\alpha$, target update rate τ.
Initialize replay buffer $\mathcal{D}$, high-level policy $\pi_\theta(z_t|s_t)$, critic $Q_\phi(s_t, z_t)$, target network $Q_{\bar{\phi}}(s_t, z_t)$
for each iteration **do**
  for every H environment steps **do**
    $z_t \sim \pi(z_t|s_t)$ ▷ sample skill from policy
    $s_{t'} \sim p(s_{t+H}|s_t, z_t)$ ▷ execute skill in environment
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, z_t, \tilde{r}(s_t, z_t), s_{t'}\}$ ▷ store transition in replay buffer
  for each gradient step **do**
    $\bar{Q} = \tilde{r}(s_t, z_t) + \gamma\left[Q_{\bar{\phi}}(s_{t'}, \pi_\theta(z_{t'}|s_{t'})) - \alpha D_{\mathrm{KL}}\left(\pi_\theta(z_{t'}|s_{t'}), p_{\boldsymbol{a}}(z_{t'}|s_{t'})\right)\right]$ ▷ compute Q-target
    $\theta \leftarrow \theta - \lambda_\pi \nabla_\theta\left[Q_\phi(s_t, \pi_\theta(z_t|s_t)) - \alpha D_{\mathrm{KL}}(\pi_\theta(z_t|s_t), p_{\boldsymbol{a}}(z_t|s_t))\right]$ ▷ update policy weights
    $\phi \leftarrow \phi - \lambda_Q \nabla_\phi\left[\frac{1}{2}\left(Q_\phi(s_t, z_t) - \bar{Q}\right)^2\right]$ ▷ update critic weights
    $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha\left[\alpha \cdot \left(D_{\mathrm{KL}}(\pi_\theta(z_t|s_t), p_{\boldsymbol{a}}(z_t|s_t)) - \delta\right)\right]$ ▷ update alpha
    $\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}$ ▷ update target network weights
return trained policy $\pi_\theta(z_t|s_t)$

Experiments

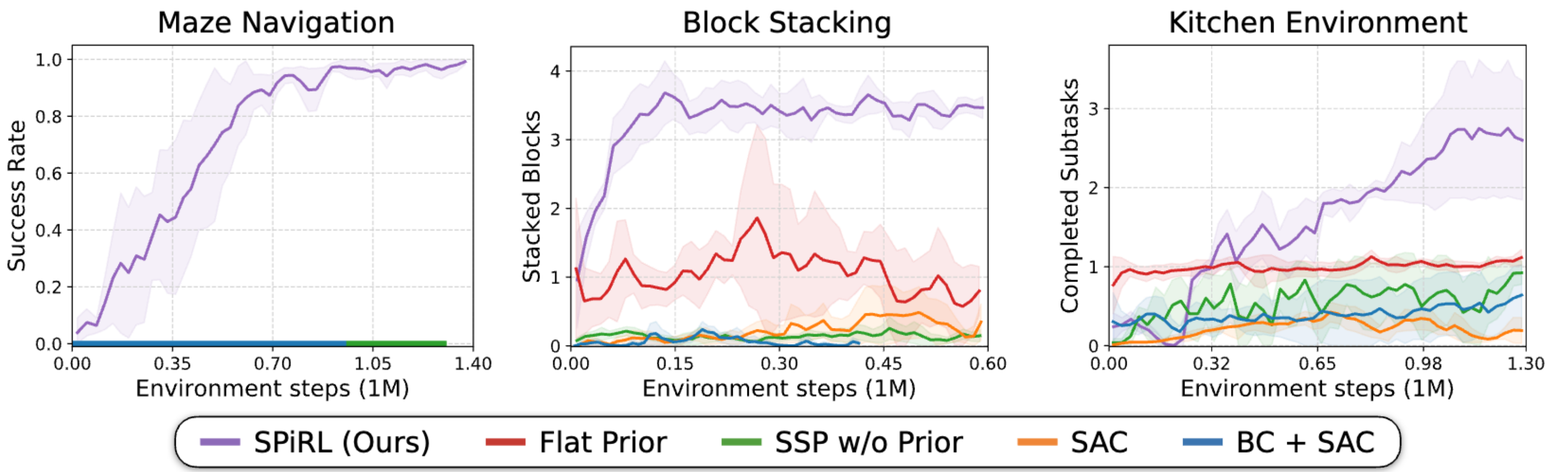


단순한 미로 -> 더 크고 복잡한 미로
state : image

5개 쌓기 -> 11개 쌓기
state : image

4task -> 7task
state : robot configuration

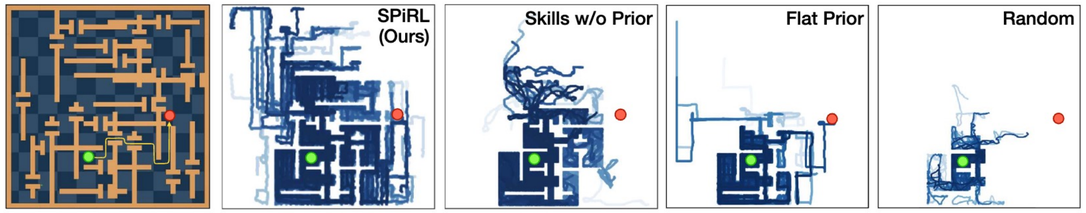
Experiments



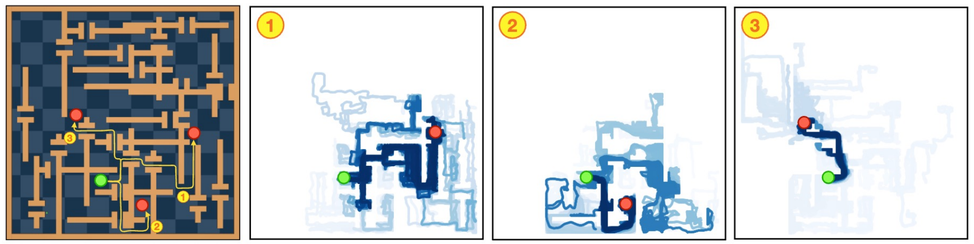
- **SAC** : from scratch SAC
- **SAC with single-step action prior (Flat prior)**: regularized by BC ,instead of skill prior
- **BC finetune SAC** : policy network initialized by BC
- **Skill Space Policy without prior** : policy learn in the skill space without skill prior

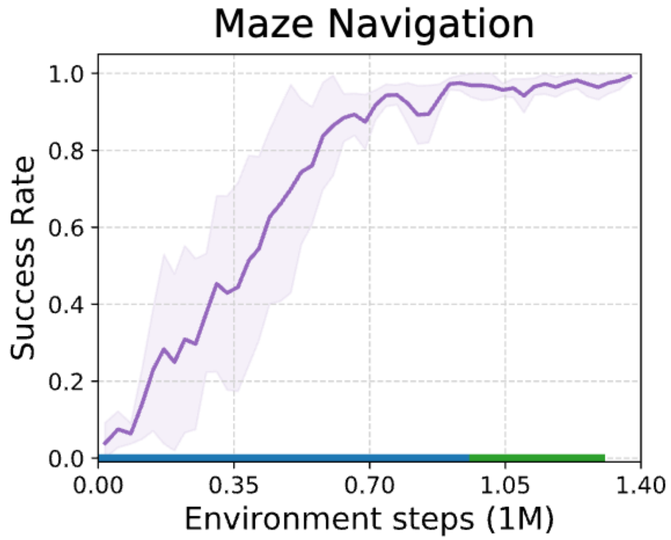
Experiments

학습 과정 동안 **MAZE**를 탐색한 영역



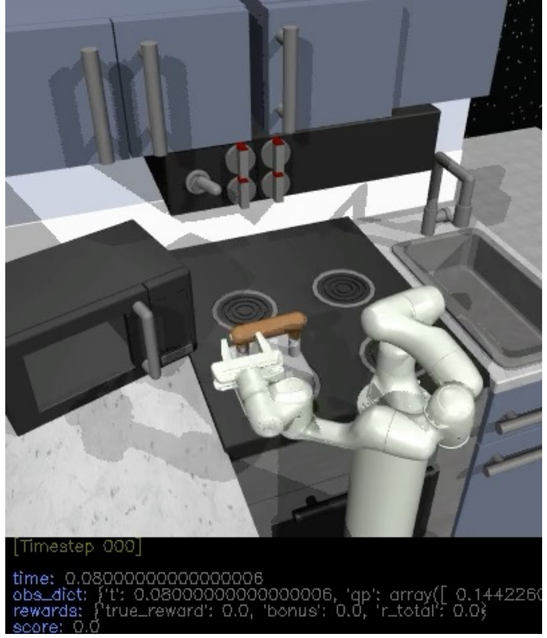
하나의 **Skill Prior**로 다양한 **goal**에 **reaching** 가능

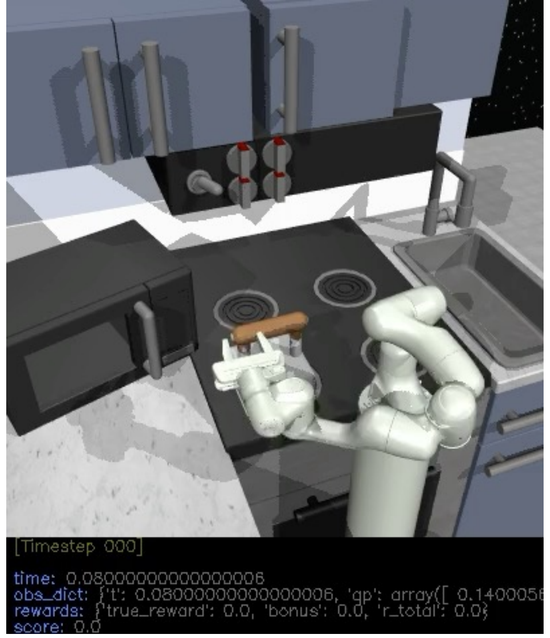




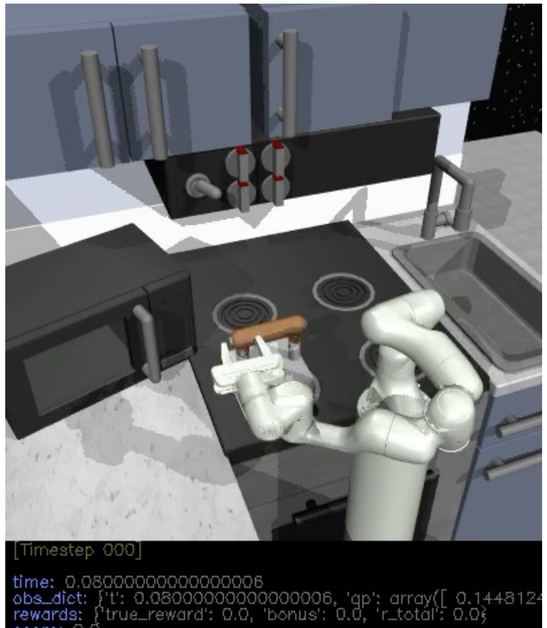
skill을 기반한 temporal abstraction과 skill prior를 이용하기에 벽에 Random처럼 벽에 부딪히지 않고 더 넓은 탐색 가능하다. flat prior는 temporal abstraction이 부족하고, skills w/o prior는 skill embedding space에서 무작위하게 explore 하기에 효과적으로 탐색하지 못함

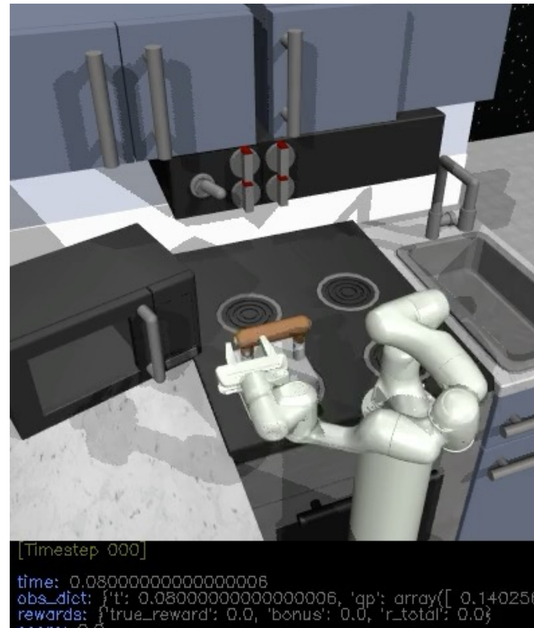
1M step





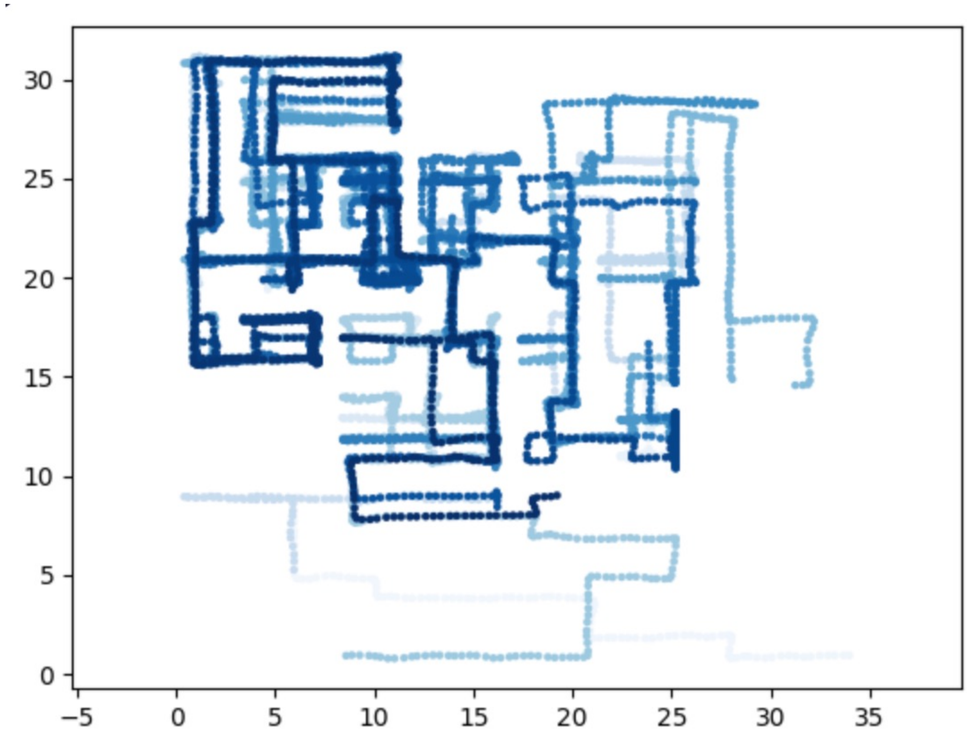
0.3M step

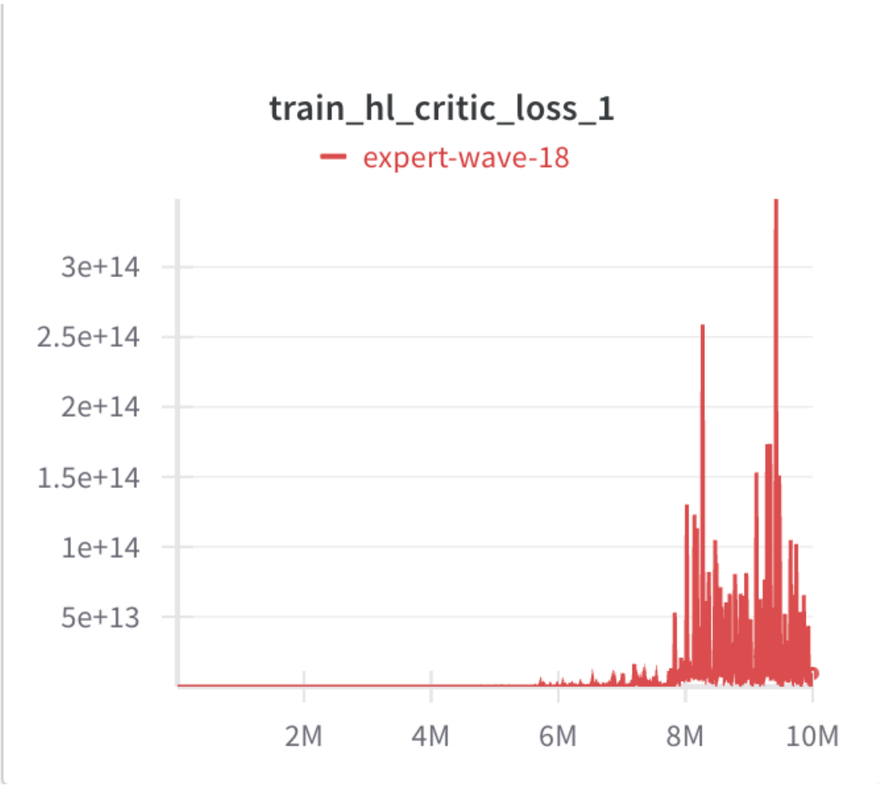




Experiments

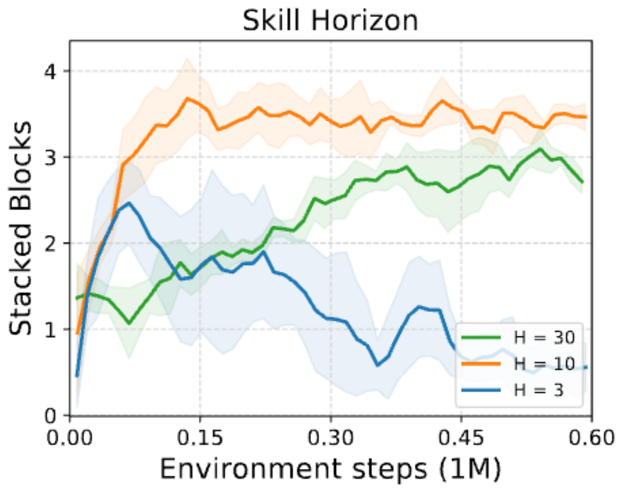
굉장히 뚜렷하게 길을 지나 다님

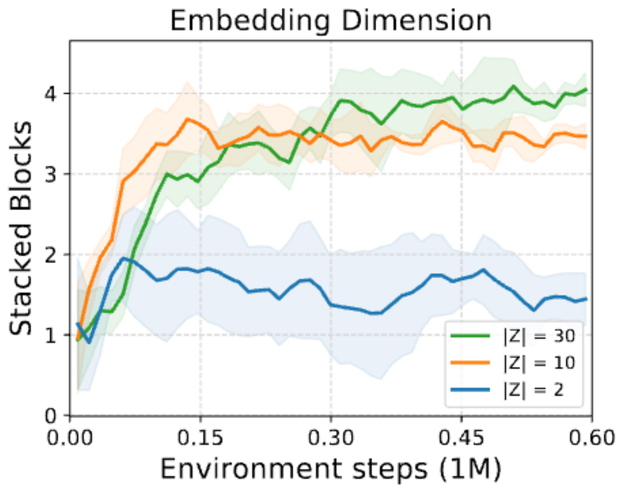




Experiments

Skill의 Horizon이 미치는 영향과 Embedding Dimension의 크기에 따른 성능 변화





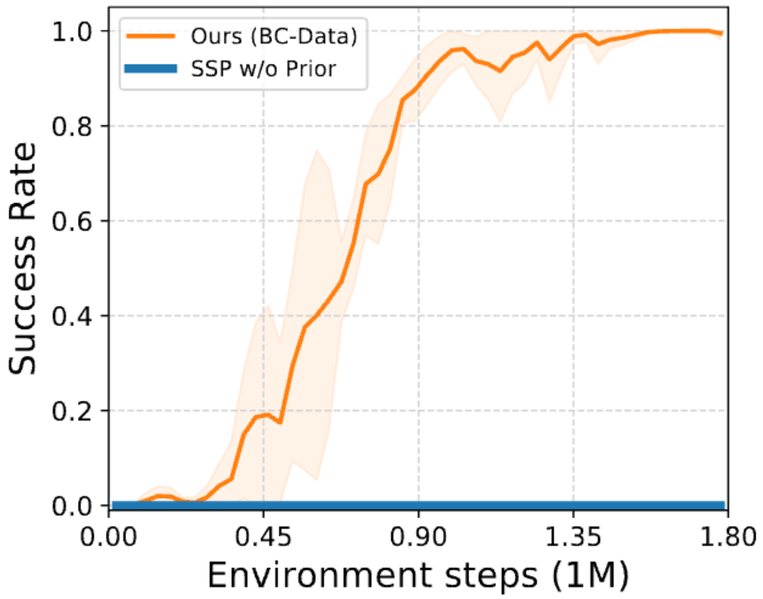
- embedding space이 어느정도 이상이라면 dimension을 늘려도 성능의 상승이 없다.
  왜? 어차피 사용하는 exploration하는 embedding space region는 제한적이기 때문에 차원이 높아져도 쓸 부분
  사용한다.

- 적당한 Horizon이라면 충분히 잘 동작하지만 너무 짧으면 temporal abstraction의 이득이 없고 너무 길면 skill
  경우의 수가 더 많아지게 되어 skill exploration이 더 어려워져 수렴이 느려진다.

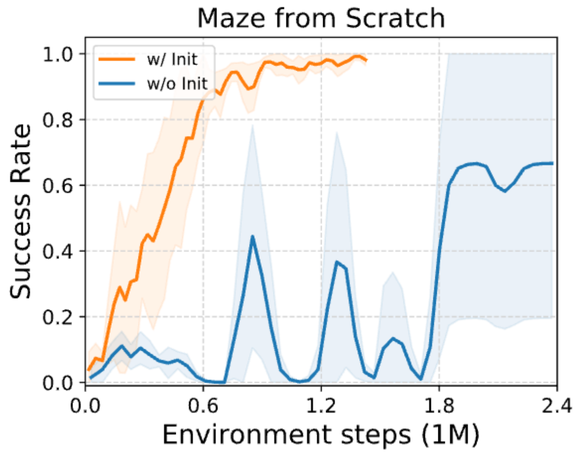
Experiments

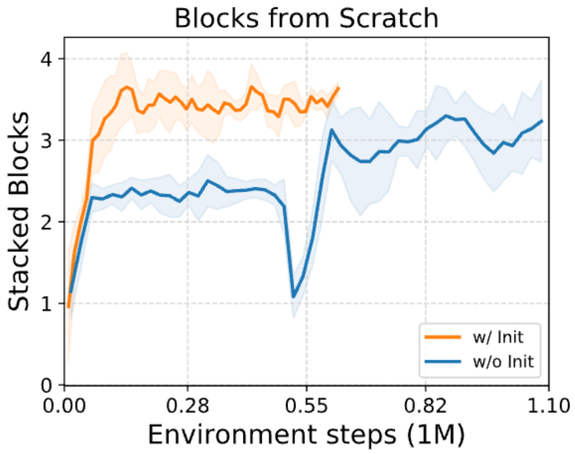
data가 non-expert일 때도 잘할 수 있는지 알기위해
BC로 학습해 얻은 data를 통해 SPiRL을 해봄

Maze from scratch



Policy를 skill prior로 초기화 하는 것이 큰 영향을 미침





Contrbutions

(1) we design a model for jointly learning an embedding space of skills and a prior over skills from unstructured data

(2) we extend maximum-entropy RL to incorporate learned skill priors for downstream task learning

(3) we show that learned skill priors accelerate learning of new tasks across three simulated navigation and robot ma