Composing Complex Skills by Learning Transition Policies

Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S. Hu, Joseph J. Lim

Composing Complex Skills by Learning Transition Policies

Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S. Hu, Joseph J. Lim

Motivation

- Can machines similary learn new and complex tasks by reusing acquired skills and learning transitions between them?

Approach

- A transition policy, which learns to smoothly navigate from an ending state of a skill to suitable initial states of the following skill
- A proximity predictor, which outputs the proximity to the initiation set of the next skill and acts as a dense reward function for the transition policy

Results & New finding

- Transition policies enable us to effectively compose complex skills with existing primitive skills
- The proposed induced rewards computed using the proximity predictor further improve training efficiency by providing more dense information than the sparse rewards from the environments

Discussion & Comments

- Learning to assess the successful termination of primitive policies together with learning transition policies is a promising future direction

Background

- Can machines similary learn new and complex tasks by reusing acquired skills and learning transitions between them?

Riedmiller (2018)

- Learning to perform composite and long-term tasks from scratch
- (cons) requires extensive exploration and sophisticated reward design , which can introduce undesired behaviors

Pastor (2009), Mulling (2013), Andreas (2017)

- Modular methods sequentially execute acquired skills with a rule-basd meta-policy, enabling machines to solve complicated tasks
- This approach assumes that a task can be clearly decomposed into several subtasks which are smoothly connected to each other
- (cons) This assumption does not hold in many continuous control problems where a given skill may be executed in stating states not considered during training or designing and thus, fail to achieve its goal

Suggestion

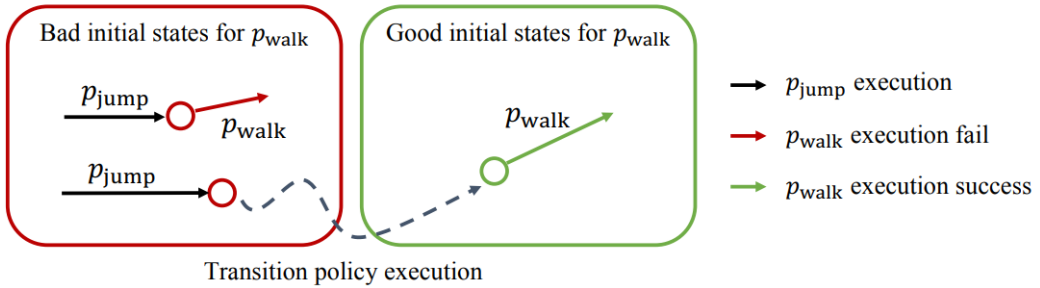

Figure 1: Concept of a transition policy. Composing complex skills using primitive skills requires smooth transition between primitive skills since a following primitive skill might not be robust to ending states of the previous one. In this example, the ending states (red circles) of the primitive policy p_{jump} are not good initial states to execute the following policy p_{walk}. Therefore, executing p_{walk} from these states will fail (red arrow). To smoothly connect the two primitive policies, we propose a transition policy which navigates an agent to suitable initial states for p_{walk} (dashed arrow), leading to a successful execution of p_{walk} (green arrow).

A transition policy

- To bridge the gap between skills
- learns to smoothly navigate from an ending state of a skill to suitable initial states of the following skill

Suggestion

Problem

- Due to
  - the temporal credit assignment problem
  - The lack of information from failing trajectories
- Sparse success/failure reward is challenging to learn from

A proximity predictor

- outputs the proximity to the initiation set of the next skill
- acts as a dense reward function for the transition policy

1) Real-world tasks often require diverse behaviors and longer temporal dependencies

Sutton (1999)

- In hierarchical reinforcement learning, the option framework learns meta actions (options), a series of primitive actions over a period of time

Schmidhuber (1990), Daniel (2016), Bacon (2017), Vezhnevets (2017), Dilokthanakul (2017), Levy (2017), Frans (2018), Co-Reyes (2018), Mao (2018)

- Unsupervised approaches to discover meta actions have been proposed

Andreas (2016)

- To exploit pre-trained modules as low-level controllers, neural module networks have been proposed, which constructs a new network dedicated to a given query using a collection of reusable modules

1) Real-world tasks often require diverse behaviors and longer temporal dependencies

In the RL domain

- a meta-controller is trained
  - to follow instructions (Ohet, 2017)
  - to follow demonstrations (Xu, 2017)
  - to support multi-level hierarchies (Gudimella, 2017)

In the robotics domain

- A modular approach that
  - learns table tennis by selecting appropriate low-level controllers (Pastor, 2009) (Kober, 2010) (Mulling, 2013)
  - learn abstract skills while experiencing a distributions of tasks and then solve a new task with the learned primitive skills (Andreas, 2017) (Frans, 2018)

These modular approaches result in undefined behavior when two skills are not smoothly connected

2) Deep RL techniques for continuous control demand dense dense reward signals

Ho & Ermon (2016), Merel (2017), Wang (2017), Bahdanau (2019)

- Adversarial reinforcement learning employs a discriminator which learns to judge the state or the policy
- The policy takes as rewards the output of the discriminator

These methods assume ground truth trajectories or goal states are given

Summary

These modular approaches result in
undefined behavior
when two skills are not smoothly connected

Our proposed framework aims to bridge this gap
by training transition policies
in a model-free manner
to navigate the agent
from unseen states for following skills
to suitable initial states

These methods assume
ground truth trajectories or goal states are given

Our method collects
both success and failure trajectories online
to train proximity predictors
which provide rewards for transition policies

Modular Framework with Transition Policies

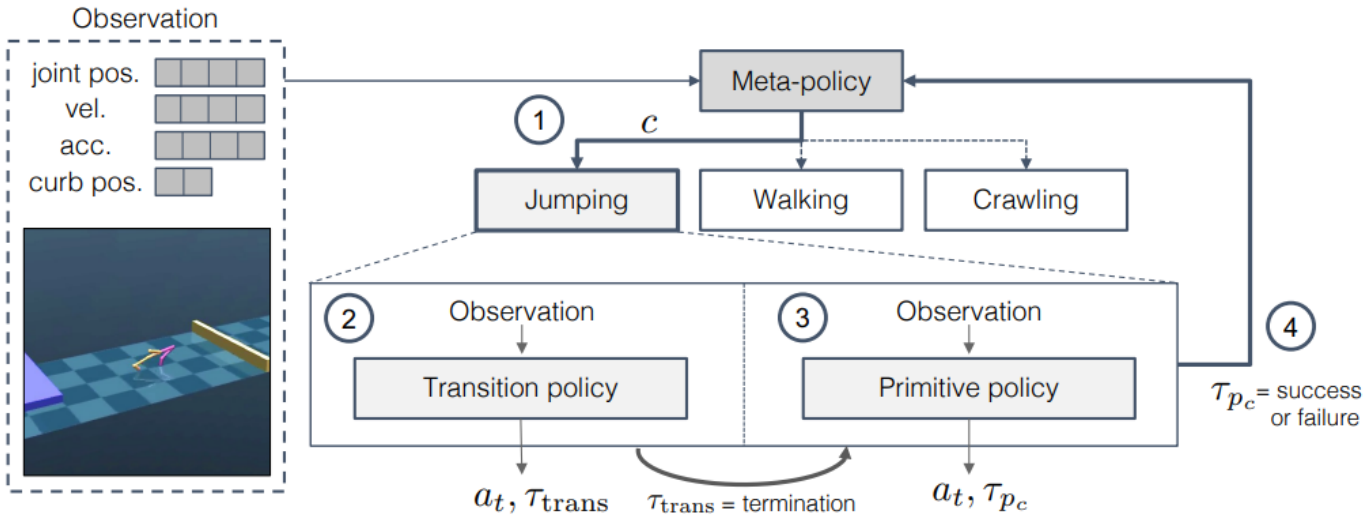

Figure 2: Our modular network augmented with transition policies. To perform a complex task, our model repeats the following steps: (1) The meta-policy chooses a primitive policy of index c; (2) The corresponding transition policy helps initiate the chosen primitive policy; (3) The primitive policy executes the skill; and (4) A success or failure signal for the primitive skill is produced.

- A modular framework = a meta-policy + primitive policies + transition policies

Modular Framework with Transition Policies

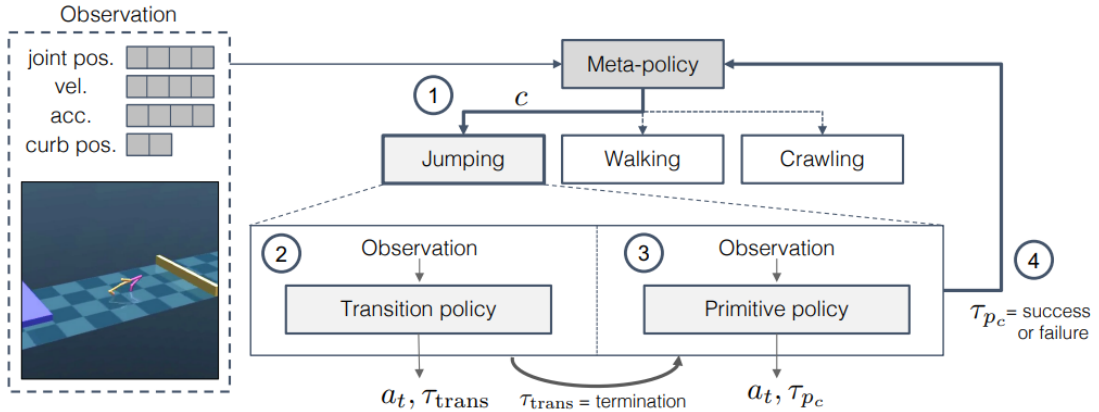

Figure 2: Our modular network augmented with transition policies. To perform a complex task, our model repeats the following steps: (1) The meta-policy chooses a primitive policy of index c; (2) The corresponding transition policy helps initiate the chosen primitive policy; (3) The primitive policy executes the skill; and (4) A success or failure signal for the primitive skill is produced.

- (1) The meta-policy chooses a primitive skill p_c to execute
  (at the beginning and whenever the primitive skill is terminated)
  - the agent generates an action $a_t \sim \pi p_c(a|s_t)$ based on the current state s_t

Modular Framework with Transition Policies

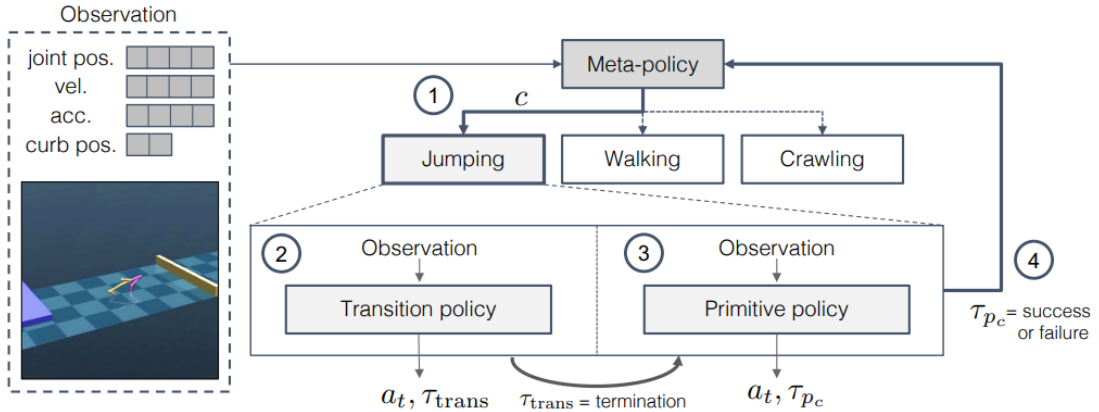

Figure 2: Our modular network augmented with transition policies. To perform a complex task, our model repeats the following steps: (1) The meta-policy chooses a primitive policy of index c; (2) The corresponding transition policy helps initiate the chosen primitive policy; (3) The primitive policy executes the skill; and (4) A success or failure signal for the primitive skill is produced.

- (2) For smooth transition, prior to running p_c, the transition policy for p_c is executed to bring the current state to a plausible initial state for p_c
  - The transition policy also learns a termination signal τ_{trans} which indicates transition termination to successfully initiate p_c

Modular Framework with Transition Policies

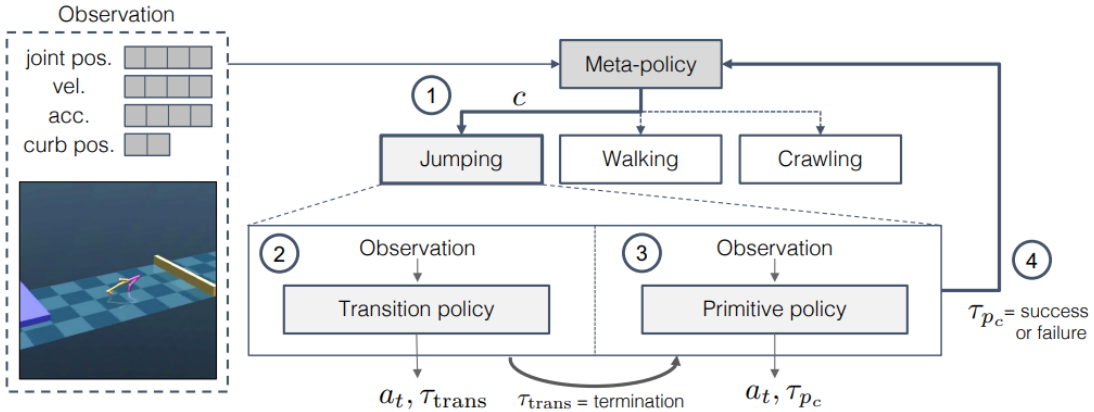

Figure 2: Our modular network augmented with transition policies. To perform a complex task, our model repeats the following steps: (1) The meta-policy chooses a primitive policy of index c; (2) The corresponding transition policy helps initiate the chosen primitive policy; (3) The primitive policy executes the skill; and (4) A success or failure signal for the primitive skill is produced.

- (3) p_c can be successfully performed

Modular Framework with Transition Policies

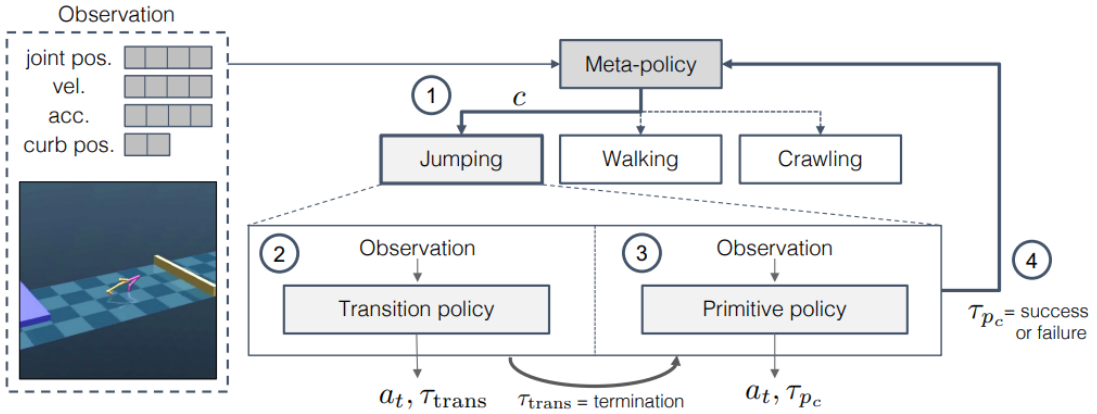

Figure 2: Our modular network augmented with transition policies. To perform a complex task, our model repeats the following steps: (1) The meta-policy chooses a primitive policy of index c; (2) The corresponding transition policy helps initiate the chosen primitive policy; (3) The primitive policy executes the skill; and (4) A success or failure signal for the primitive skill is produced.

- (4) Every primitive policy is required to generate termination signals $\tau p_c \in$ {continue,success, fail} to indicate policy completion and whether it believes the execution is successful or not

Training Transition Policies

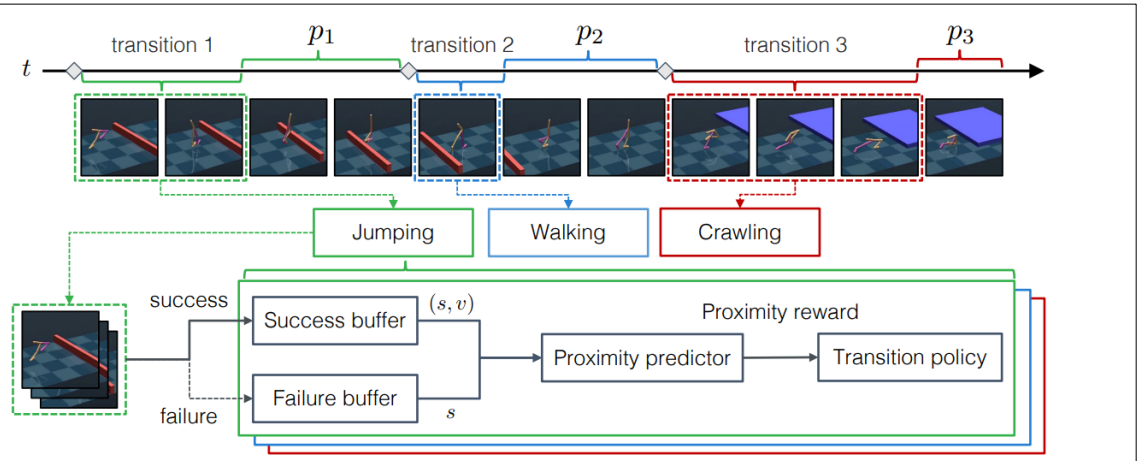

Figure 3: Training of transition policies and proximity predictors. After executing a primitive policy, a previously performed transition trajectory is labeled and added to a replay buffer based on the execution success. A proximity predictor is trained on states sampled from the two buffers to output the proximity to the initiation set. The predicted proximity serves as a reward to encourage the transition policy to move toward good initial states for the corresponding primitive policy.

- During rollouts, transition trajectories are collected and each trajectory can be naively labeled by the success execution of its corresponding primitive policy
- To alleviate the sparsity of rewards and maximize the objective of moving to viable initial states for the next primitive, we propose a proximity predictor that learns and provides a dense reward, dubbed proximity reward, of how close transition states are to the initiation set of the corresponding primitive p_c

Training Transition Policies

(1) The proximity predictor

proximity prediction

$$L_P(\omega, \mathcal{B}^S, \mathcal{B}^F) = \frac{1}{2}\mathbb{E}_{(s,v)\sim\mathcal{B}^S}[(P_\omega(s) - v)^2] + \frac{1}{2}\mathbb{E}_{s\sim\mathcal{B}^F}[P_\omega(s)^2], \quad (1)$$

collections of states from success and failure

proximity of state

trained to minimize a mean squared error of proximity prediction

Training Transition Policies

(2) The transition policy

proximity prediction at the ending state of the transition trajectory

proximity reward, the increase of predicted proximity to the initiation set

$$R_{\text{trans}}(\phi) = \mathbb{E}_{(s_0, s_1, \ldots, s_T) \sim \pi_\phi} \left[\gamma^T P_\omega(s_T) + \sum_{t=0}^{T-1} \gamma^t \left(P_\omega(s_{t+1}) - P_\omega(s_t) \right) \right]. \quad (2)$$

trained to maximize the expected discounted return

The goal of transition policy is to get close to an initiation set which can be formulated as seeking a state s predicted to be in the initiation set by the proximity predictor

Conclusions

- A modular frawework with transition policies to empower reinforcement learning agents to learn complex tasks with sparse reward by utilizing prior knowledge
- A proximity predictor which generates dense reward signals and jointly train transition policies and proximity predictors

Results

- The proposed framework solves complex tasks without reward shaping and outperforms baseline RL algorithms and other ablated baselines

Limitation and future work

- Jointly learning of a meta-policy and transition policies on a new task would make our framework more flexible
- To alleviate the exploration problem with sparse rewards, our transition policy training can incorporate exploration methods such as count-based exploration bonuses and curiosity-driven intrinsic reward
- Learning to assess the successful termination of primitive policies together with learning transition policies is a promising future direction