

1710.09767 - Meta Learning Shared Hierarchies

- **Yunqiu Xu**
 - Other reference:
 - [OpenAI Blog: Learning a Hierarchy](#)
 - [Meta Learning Shared Hierarchies | Two Minute Papers](#)
 - [Videos for "Meta Learning Shared Hierarchies"](#)
 - [Paper Notes : Meta Learning Shared Hierarchies](#)
-

1. Introduction

- Challenge: different tasks have different optimal policies → hard to generalize (or "transfer learning")
- Our work: MLSH
 - Hierarchical model similar to "options framework"
 - Contain a set of shared sub-policies(primitives)
 - A task-specific master policy is used to switch these sub-tasks
 - Allow for quick learning on new tasks
- What is a good hierarchy: find a set of low-level motor primitives that enable the high-level master policy to be learned quickly.
- How to solve this optimization problem:
 - Repeatedly reset the master policy to adapt the sub-policies for fast learning
 - More details can be seen in "Algorithm" part, that master policy keeps getting reset for each new task

2. Related Work

- Hierarchical RL:
 - Sutton et al. [Between mdps and semi-mdps: A framework](#)

for temporal abstraction in reinforcement learning : option framework, assumes that the options are given, some recent work seeks to learn them automatically

- Florensa et al. [Stochastic Neural Networks for Hierarchical Reinforcement Learning](#) : Use stochastic NN to learn the span of skills, the sub-policies are defined according to information-maximizing statistics
- Bacon et al. [The option-critic architecture](#) : end-to-end learning of hierarchy through the options framework
- Feudal Network [Dayan & Hinton 1993](#) , [DeepMind 2017](#) : learn a decomposition of complicated tasks into sub-goals (Manager & Workers)
- Their limitation: focus on single-task, while our work uses **multi-task setting** to learn temporally extended primitives
- Meta learning: learning to learn
 - [Dual et al. 2016](#) , [Wang et al. 2016](#) : RNN as the entire learning process
 - Mishra et al. [Meta-Learning with Temporal Convolutions](#) : Utilize temporal convolutions rather than recurrency
 - MAML [Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks](#) : Treat the test error on sampled tasks as the training error of meta-learning process, fine-tune a shared policy by optimizing through a second gradient step
 - Our difference: they try to **learn as much as possible** in a few gradient updates, we try to **learn as fast as possible** over a large number of gradient updates

3. Problem Statement

- The goal of meta-learning: optimize the expected return over the sampled tasks
→ try to find shared parameter **ψ** which can be generalize to a new MDP

$$\text{maximize}_{\phi} E_{M \sim P_M, t=0, \dots, T-1} [R]$$

- Shared vector ϕ :
 - Consist of a set of subvectors : $\phi_1, \phi_2, \dots, \phi_K$
 - Each subvector ϕ_k defines a sub-policy $\pi_{\phi_k}(a|s)$

- Parameter θ : shape stochastic policy (master policy), whose action is to choose sub-policy's index ($1, 2, \dots, k$) every N timesteps

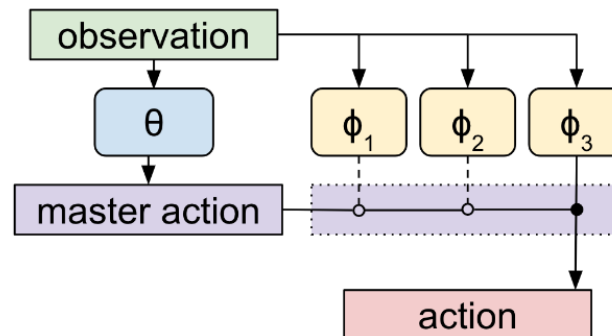


Figure 1: Structure of a hierarchical sub-policy agent. θ represents the master policy, which selects a sub-policy to be active. In the diagram, ϕ_3 is the active sub-policy, and actions are taken according to its output.

- By discovering a strong set of subpolicies ϕ , we can handle new task learning by updating master policy θ only
- Why it's faster: treat a learning problem with a horizon which is only $1/n$ times as long

4. Algorithm

Algorithm 1 Meta Learning Shared Hierarchies

```

Initialize  $\phi$ 
repeat
  Initialize  $\theta$ 
  Sample task  $M \sim P_M$ 
  for  $w = 0, 1, \dots, W$  (warmup period) do
    Collect  $D$  timesteps of experience using  $\pi_{\phi, \theta}$ 
    Update  $\theta$  to maximize expected return from  $1/N$  timescale viewpoint
  end for
  for  $u = 0, 1, \dots, U$  (joint update period) do
    Collect  $D$  timesteps of experience using  $\pi_{\phi, \theta}$ 
    Update  $\theta$  to maximize expected return from  $1/N$  timescale viewpoint
    Update  $\phi$  to maximize expected return from full timescale viewpoint
  end for
until convergence

```

- Policy update:

- Goal: learn sub-policy parameters ϕ
- Sample a task \mathcal{M} and initialize master policy θ
- Warm-up period: optimize θ
 - Consider the selection of a sub-policy as a single action
 - The next N timesteps, along with corresponding state changes and rewards, are viewed as a single environment transition
- Joint update period: both θ and ϕ are updated
 - Treat the master policy as an extension of the environment
 - For each N -timestep slice of experience, we only update the parameters of the sub-policy that had been activated by master policy
- For a new task: reset θ and repeat

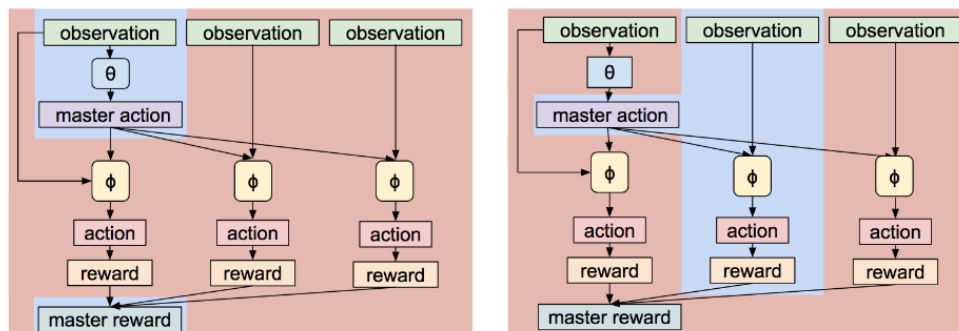


Figure 2: Unrolled structure for a master policy action lasting $N = 3$ timesteps. Left: When training the master policy, the update only depends on the master policy’s action and total reward (blue region), treating the individual actions and rewards as part of the environment transition (red region). Right: When training sub-policies, the update considers the master policy’s action as part of the observation (blue region), ignoring actions in other timesteps (red region)

- Why our method is faster:
 - Traditional meta-learning : optimize reward (master policy θ) over an entire inner loop
 - MLSH : optimize ϕ towards maximizing reward within a single episode
 - An assumption : given fixed sub-policies ϕ , we can learn optimal θ during warmup period
 - Only when θ is nearly optimal will ϕ be updated

5. Experiment

- More results can be seen in <https://sites.google.com/site/mlshsupplementals/>

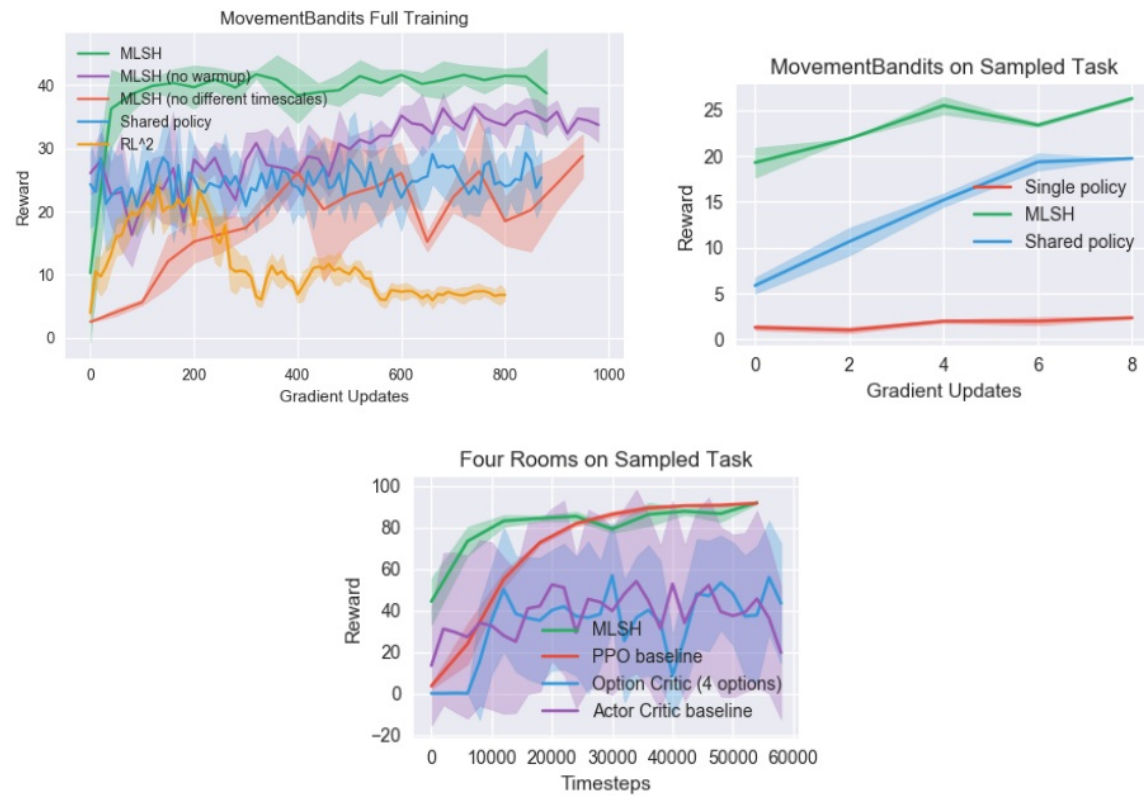


Figure 4: Learning curves for 2D Moving Bandits and Four Rooms



Figure 7: Learning curves for Twowalk and Walk/Crawl tasks

6. Conclusion

- This work combines meta-learning with hierarchical RL, try to learn faster over a large number of gradient updates
- For each task, the master policy θ is reset and learned to be optimal with fixed sub-policies ϕ , then nearly-optimal master policy will be treated as an extension of environment and ϕ will be updated