1710.03641 - Continuous Adaptation via Meta-Learning in Nonstationary and Competitive Environments

- **Yunqiu Xu**
- Other reference:
  - ICLR 2018 openreview
  - OpenAI blog : Meta-Learning for Wrestling

---

1. Introduction

- Challenge: real world is non-stationary
  - Dynamics and objectives change over life-long time
  - Multiple learning actors
- Traditional method: context detection and tracking
  - learn policy based on those already happened, then continuously fine-tuning for new environment
  - Limitation:
    - Inefficient
    - Limited interaction before changing $\rightarrow$ maybe insifficient to learn policy $\rightarrow$ **we need few-shot**
- Insight: non-stationary task can be seen as a sequence of stationary tasks $\rightarrow$ change to multi-task problem
- This paper:
  - Consider the problem of continuous adaptation to a learninging opponent in a competitive multi-agent setting
  - Design a new environment: RoboSumo
  - Dvelop gradient-based meta-learning algorithm for adaptation in dynamically

changing and adversarial scenarios

2. Related Work

- Life-long learning: solving multiple tasks sequentially by using already learned knowledge
- Never-ending learning: keep a set of tasks, the set keeps growing, and the performance on all tasks keeps growing as well
- **Continuous adaptation**:
    - Solve a single but nonstationary task or environment
    - Learn an agent to adapt to changes of environment
    - With limited data or experience (few-shot)

3. Method

- Goal: handle continuous adaptation with few-shot
- Method: rederive MAML for multi-task, then extend to dynamically changing tasks

3.1 Probabilistic view of MAML

- Here Eq.1-3 are similar to original MAML
- A task is defined as

$$T = \{L_T, P_T(x), P_T(x_{t+1}|x_t, a_t), H\} \quad (1)$$

- Inner loop:
    - In original MAML, ϕ is called θ'
    - In original MAML, α is call β in inner loop

$$\phi := \theta - \alpha \nabla_\theta \mathcal{L}_T\left(\boldsymbol{\tau}_\theta^{1:K}\right), \text{ where } \mathcal{L}_T\left(\boldsymbol{\tau}_\theta^{1:K}\right) := \frac{1}{K}\sum_{k=1}^{K} \mathcal{L}_T(\boldsymbol{\tau}_\theta^k), \text{ and } \boldsymbol{\tau}_\theta^k \sim P_T(\boldsymbol{\tau} \mid \theta) \quad (2)$$

- Meta objective:

$$\min_{\theta} \mathbb{E}_{T\sim\mathcal{D}(T)}\left[\mathcal{R}_T(\theta)\right], \text{ where } \mathcal{R}_T(\theta) := \mathbb{E}_{\boldsymbol{\tau}_\theta^{1:K}\sim P_T(\boldsymbol{\tau}|\theta)}\left[\mathbb{E}_{\boldsymbol{\tau}_\phi\sim P_T(\boldsymbol{\tau}|\phi)}\left[\mathcal{L}_T(\boldsymbol{\tau}_\phi)\mid \boldsymbol{\tau}_\theta^{1:K},\theta\right]\right] \quad (3)$$

where $\boldsymbol{\tau}_\theta$ and $\boldsymbol{\tau}_\phi$ are trajectories obtained under π_θ and π_ϕ, respectively.

- Probabilistic view:
  - Task T , trajectories τ and policies π_θ are random variables, phi is generated from conditional distribution $P_T(\phi|\theta, \tau_{1:k})$
  - Inner loop update: equivalent to assuming the delta distribution

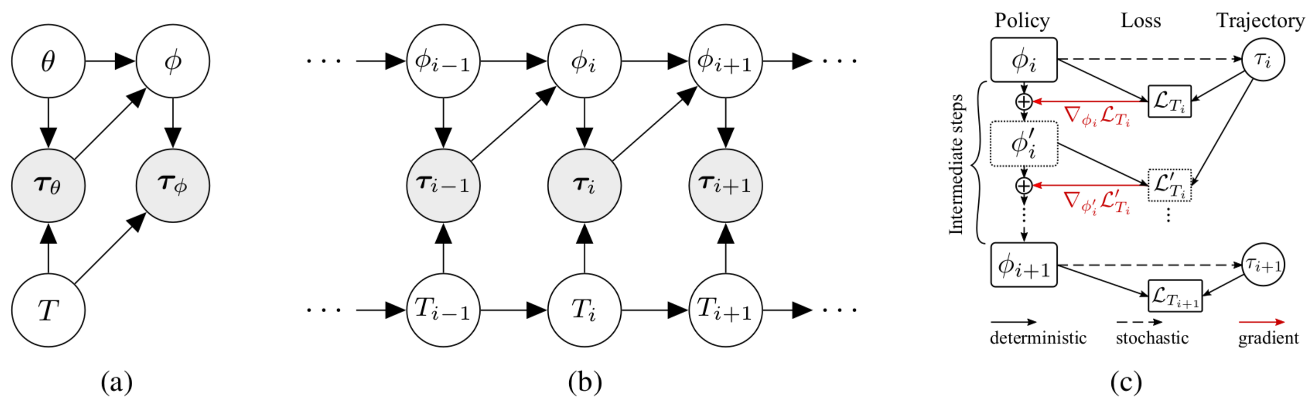
$$P_T(\phi|\theta, \tau_{1:k}) := \delta(\theta - \alpha\nabla_\theta \frac{1}{K}\sum_{k=1}^{K} L(\tau_k))$$

  - Optimize meta-objective: PG where the gradient of $R_T(\theta)$

$$\nabla_\theta \mathcal{R}_T(\theta) = \mathbb{E}_{\substack{\boldsymbol{\tau}_\theta^{1:K}\sim P_T(\boldsymbol{\tau}|\theta)\\ \boldsymbol{\tau}_\phi\sim P_T(\boldsymbol{\tau}|\phi)}}\left[\mathcal{L}_T(\boldsymbol{\tau}_\phi)\left[\nabla_\theta \log \pi_\phi(\boldsymbol{\tau}_\phi) + \nabla_\theta \sum_{k=1}^{K} \log \pi_\theta(\boldsymbol{\tau}_\theta^k)\right]\right] \quad (4)$$

3.2 Continuous adaptation via meta-learning

- $D(T)$ is defined by the environment changes, and the tasks become sequentially dependent
- Goal:
  - Find the dependence between consecutive tasks
  - Meta-learn a rule to minimize total expected loss during interacting
  - An example, if our enemy changes action, we need to do some adjustment as well



- Probabilistic graph of MAML

- (a) MAML in a multi-task RL setting
- (b) Extend to continuous adaptation
  - Policy and trajectories at a previous step are used to construct a new policy for the current step, i.e. $\{\phi_i, \tau_i\} \rightarrow \phi_{i+1}$
- (c) Computation graph for the meta-update from ϕ_i to ϕ_{i+1}
  - The model is optimized via truncated backpropagation through time starting from $L_{T_{i+1}}$

- From (b) we represent nonstationary as a sequence of stationary tasks, then the goal is try to minimize the expected loss of L tasks, where L is a given number

$$\min_{\theta} \mathbb{E}_{\mathcal{P}(T_0),\mathcal{P}(T_{i+1}|T_i)}\left[\sum_{i=1}^{L} \mathcal{R}_{T_i,T_{i+1}}(\theta)\right] \tag{5}$$

- Expected loss on a pair of consecutive tasks

$$\mathcal{R}_{T_i,T_{i+1}}(\theta) := \mathbb{E}_{\boldsymbol{\tau}_{i,\theta}^{1:K} \sim P_{T_i}(\boldsymbol{\tau}|\theta)}\left[\mathbb{E}_{\boldsymbol{\tau}_{i+1,\phi} \sim P_{T_{i+1}}(\boldsymbol{\tau}|\phi)}\left[\mathcal{L}_{T_{i+1}}(\boldsymbol{\tau}_{i+1,\phi}) \mid \boldsymbol{\tau}_{i,\theta}^{1:K}, \theta\right]\right] \tag{6}$$

The principal difference between the loss in (3) and (6) is that trajectories $\boldsymbol{\tau}_{i,\theta}^{1:K}$ come from the current task, T_i, and are used to construct a policy, π_ϕ, that is good for the upcoming task, T_{i+1}.

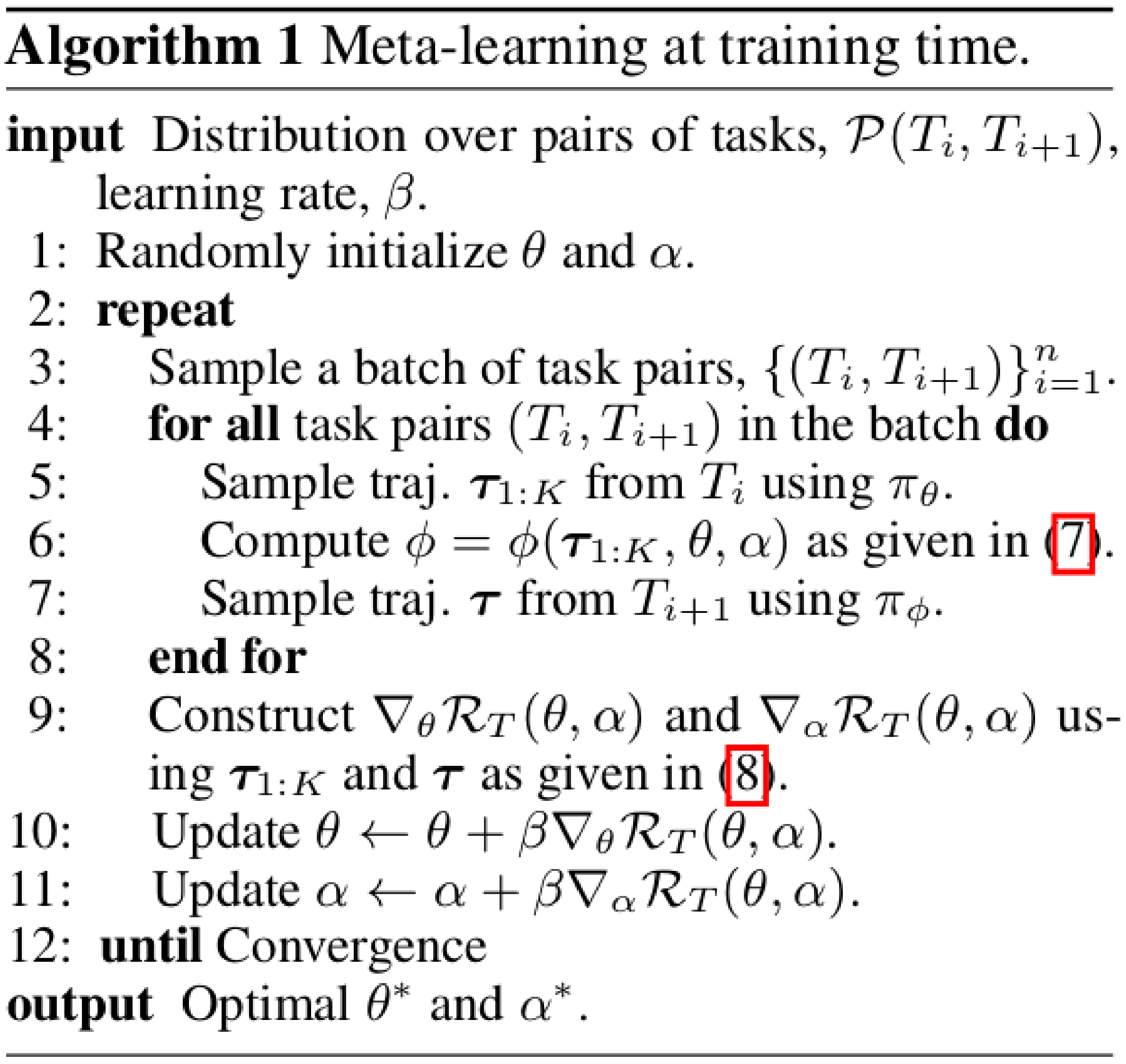
- To construct the parameters of policy for task T_{i+1}, we start from θ, note that here we can also change inner step size α

$$\begin{aligned} \phi_i^0 &:= \theta, \quad \boldsymbol{\tau}_\theta^{1:K} \sim P_{T_i}(\boldsymbol{\tau} \mid \theta), \\ \phi_i^m &:= \phi_i^{m-1} - \alpha_m \nabla_{\phi_i^{m-1}} \mathcal{L}_{T_i}\left(\boldsymbol{\tau}_{i,\phi_i^{m-1}}^{1:K}\right), \quad m = 1, \ldots, M-1, \\ \phi_{i+1} &:= \phi_i^{M-1} - \alpha_M \nabla_{\phi_i^{M-1}} \mathcal{L}_{T_i}\left(\boldsymbol{\tau}_{i,\phi_i^{M-1}}^{1:K}\right) \end{aligned} \tag{7}$$

- PG for (c), as expectation is taken to both T_i and T_{i+1}, we change (4) to

$$\nabla_{\theta,\alpha}\mathcal{R}_{T_i,T_{i+1}}(\theta,\alpha) = \mathbb{E}_{\substack{\boldsymbol{\tau}_{i,\theta}^{1:K} \sim P_{T_i}(\boldsymbol{\tau}|\theta) \\ \boldsymbol{\tau}_{i+1,\phi} \sim P_{T_{i+1}}(\boldsymbol{\tau}|\phi)}}\left[\mathcal{L}_{T_{i+1}}(\boldsymbol{\tau}_{i+1,\phi})\left[\nabla_{\theta,\alpha}\log \pi_\phi(\boldsymbol{\tau}_{i+1,\phi}) + \nabla_\theta \sum_{k=1}^{K} \log \pi_\theta(\boldsymbol{\tau}_{i,\theta}^k)\right]\right] \tag{8}$$

3.3 Meta-training for Continuous Adaptation



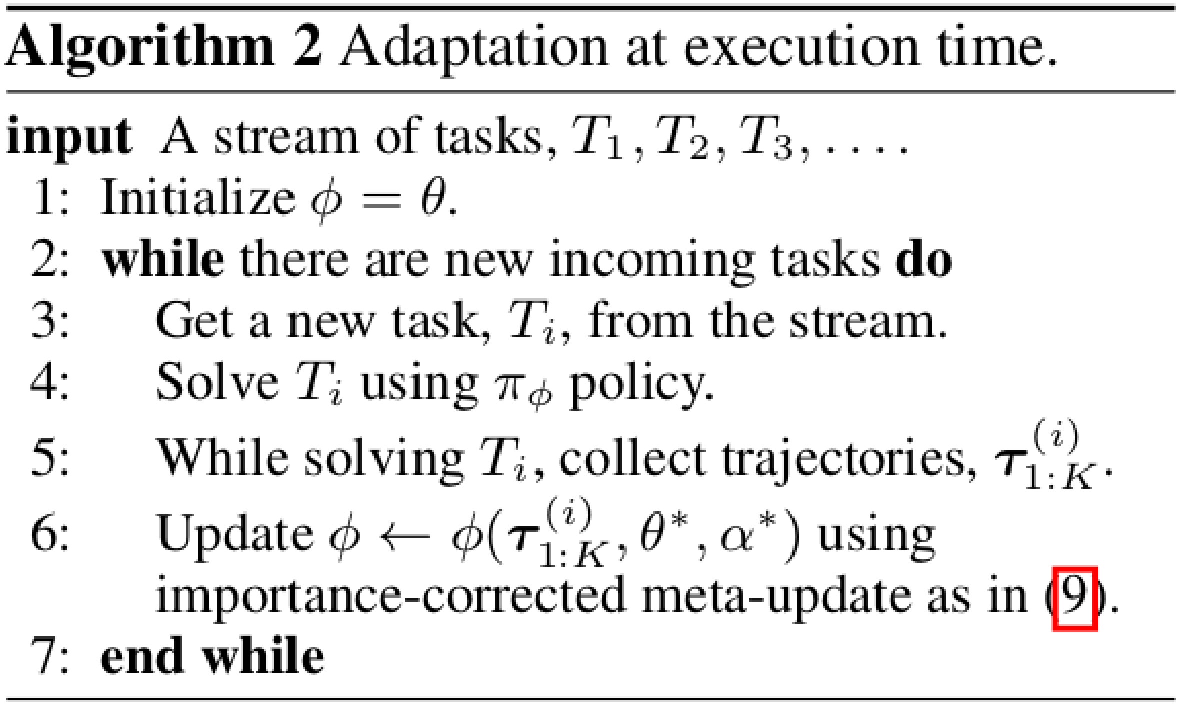
Algorithm 1 Meta-learning at training time.

```
input  Distribution over pairs of tasks, $\mathcal{P}(T_i, T_{i+1})$,
       learning rate, $\beta$.
 1: Randomly initialize $\theta$ and $\alpha$.
 2: repeat
 3:     Sample a batch of task pairs, $\{(T_i, T_{i+1})\}_{i=1}^{n}$.
 4:     for all task pairs $(T_i, T_{i+1})$ in the batch do
 5:         Sample traj. $\boldsymbol{\tau}_{1:K}$ from $T_i$ using $\pi_\theta$.
 6:         Compute $\phi = \phi(\boldsymbol{\tau}_{1:K}, \theta, \alpha)$ as given in (7).
 7:         Sample traj. $\boldsymbol{\tau}$ from $T_{i+1}$ using $\pi_\phi$.
 8:     end for
 9:     Construct $\nabla_\theta \mathcal{R}_T(\theta, \alpha)$ and $\nabla_\alpha \mathcal{R}_T(\theta, \alpha)$ us-
        ing $\boldsymbol{\tau}_{1:K}$ and $\boldsymbol{\tau}$ as given in (8).
10:     Update $\theta \leftarrow \theta + \beta \nabla_\theta \mathcal{R}_T(\theta, \alpha)$.
11:     Update $\alpha \leftarrow \alpha + \beta \nabla_\alpha \mathcal{R}_T(\theta, \alpha)$.
12: until Convergence
output  Optimal $\theta^*$ and $\alpha^*$.
```

- **Assumption: trajectories of T_i contain some infomation about T_{i+1}**
- Difference from original MAML:
  - Goal: 不仅仅优化 θ 还有一系列步长 α
  - Use a pair of consecutive tasks for training
  - During inner loop:
    - 首先基于原策略 π_θ 从 T_i 中获取一系列 trajectories $\tau_{1:K}$
    - **此处存疑, 原文中说获取trajectories的过程中还需要与T_{i+1}互动**
    - 根据这些 T_i 的 trajectories 计算 ϕ
    - π_ϕ 用于解决 T_{i+1}, 基于该策略从 T_{i+1} 中获取一个 trajectory τ
  - Meta-update:
    - 通过一系列子任务获取$\tau_{1:K}$ 以及 τ后, 计算$\nabla_\theta R_T(\theta, \alpha)$, $\nabla_\alpha R_T(\theta, \alpha)$

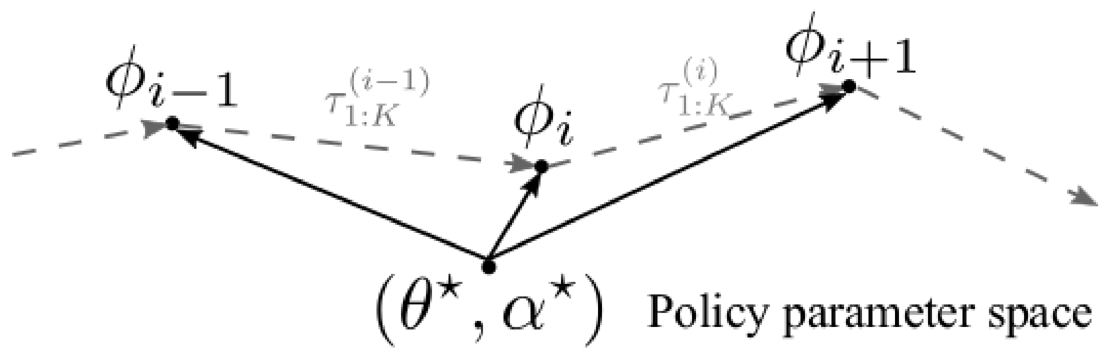
- 更新 θ 与 α

3.4 Adaptation at Execution Time (Testing)



Algorithm 2 Adaptation at execution time.

```
input A stream of tasks, $T_1, T_2, T_3, \ldots$.
1: Initialize $\phi = \theta$.
2: while there are new incoming tasks do
3:     Get a new task, $T_i$, from the stream.
4:     Solve $T_i$ using $\pi_\phi$ policy.
5:     While solving $T_i$, collect trajectories, $\boldsymbol{\tau}_{1:K}^{(i)}$.
6:     Update $\phi \leftarrow \phi(\boldsymbol{\tau}_{1:K}^{(i)}, \theta^*, \alpha^*)$ using
       importance-corrected meta-update as in (9).
7: end while
```



- Nonstationary $\rightarrow$ can not access to same task multiple times
- How to handle : keep acting according to π_ϕ and re-use past experience to for computing updates of ϕ for each new incoming task
- Importance weight correction : past experience obtained by π_ϕ is different from π_θ, we need to make some adjustment

$$\phi_i := \theta - \alpha \frac{1}{K} \sum_{k=1}^{K} \left(\frac{\pi_\theta(\boldsymbol{\tau}^k)}{\pi_{\phi_{i-1}}(\boldsymbol{\tau}^k)} \right) \nabla_\theta \mathcal{L}(\boldsymbol{\tau}^k), \quad \boldsymbol{\tau}^{1:K} \sim P_{T_{i-1}}(\boldsymbol{\tau} \mid \phi_{i-1}), \tag{9}$$

4. Experiment

- A new environment : RoboSumo
- Let robot wrestle

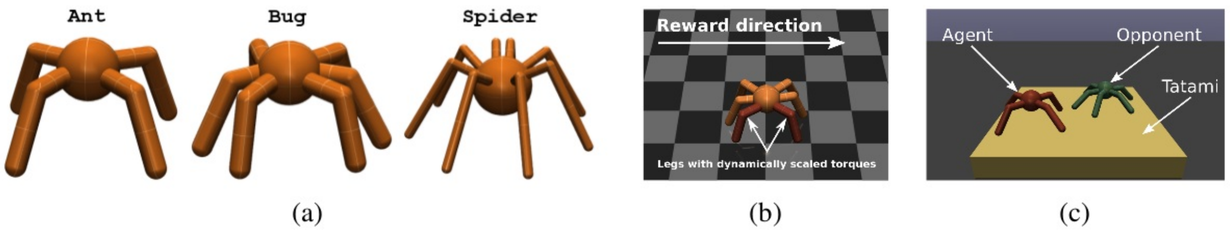


Figure 2: (a) The three types of agents used in experiments. The robots differ in the anatomy: the number of legs, their positions, and constraints on the thigh and knee joints. (b) The nonstationary locomotion environment. The torques applied to red-colored legs are scaled by a dynamically changing factor. (c) The RoboSumo environment.

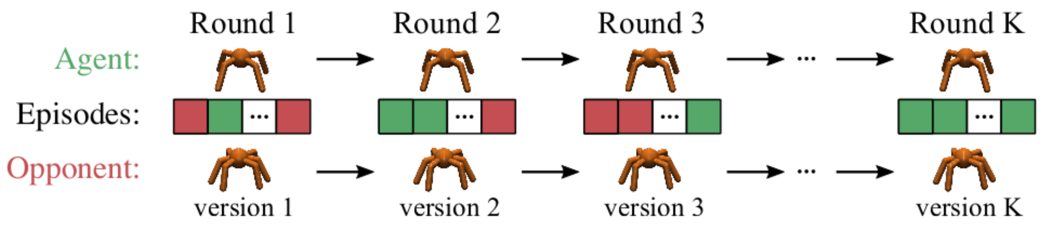


Figure 3: An agent competes with an opponent in an iterated adaptation games that consist of multi-episode rounds. The agent wins a round if it wins the majority of episodes (wins and losses illustrated with color). Both the agent and its opponent may update their policies from round to round (denoted by the version number).

- Result 1

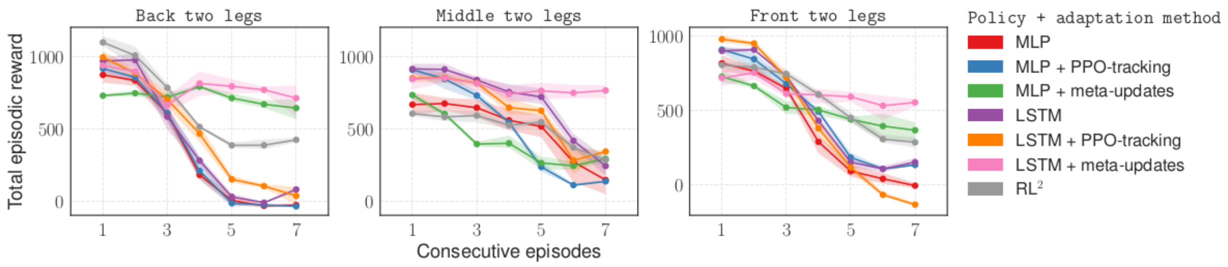


Figure 4: Episodic rewards for 7 consecutive episodes in 3 held out nonstationary locomotion environments. To evaluate adaptation strategies, we ran each of them in each environment for 7 episodes followed by a full reset of the environment, policy, and meta-updates (repeated 50 times). Shaded regions are 95% confidence intervals. Best viewed in color.

- Omit some results

5. Conclusion

- Regard nonstationarity as a sequence of stationary tasks
- Train agents to exploit the dependencies between consecutive tasks
- During testing, the model can handle similar nonstationarities + Problem : 该工作把动态环境看作一系列静态环境的序列, 但真实环境会更复杂
- Future work

Current	**Future**
One-step-ahead update of the policy	Extend to fully recurrent meta-updates $\rightarrow$ take into full history of interaction
(c) requires second order derivatives at training time	Utilize information provided by the loss but avoid explicit backpropagation through the gradients
Can't handle sparse reward $\rightarrow$ the meta-updates use policy gradients and heavily rely on the reward signal	Try to introduce auxiliary dense rewards designed to enable meta-learning