NoRML:No-Reward Meta Learning

presented by Jason TOKO

Outline

- Background & Motivation
- Review on MAML-RL
- No-Reward Meta Learning
  - Learned Advantage Function
  - Learned Offset
  - Algorithm
- Experiment
  - Point Agent
  - Continuous Control
- Conclusion

Outline

- Background & Motivation
- Review on MAML-RL
- No-Reward Meta Learning
  - Learned Advantage Function
  - Learned Offset
  - Algorithm
- Experiment
  - Point Agent
  - Continuous Control
- Conclusion

Background & Motivation

- MAML
  - MAML is successful at adapting policies to different tasks that are defined by **reward changes**.
  - However, it is less effective when adapting to other changes, such as **dynamics changes**, **sensor drifts**, or **missing reward signals**.
- An agent should recognize dynamics change from the state-action transitions alone to adapt its behavior.
- Goal:
  - To develop a **model-free meta-RL** algorithm that can learn to quickly adapt a policy to dynamics changes and sensor drifts **without external rewards**——NoRML

Outline

- Background & Motivation
- Review on MAML-RL
- No-Reward Meta Learning
  - Learned Advantage Function
  - Learned Offset
  - Algorithm
- Experiment
  - Point Agent
  - Continuous Control
- Conclusion

Review on MAML-RL

- Adaptation(Policy Gradient)：

$$\begin{aligned}\boldsymbol{\theta}_i &= \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}, D_i^{\text{train}}) \\ &= \boldsymbol{\theta} + \alpha \sum_{(\boldsymbol{s}_t, \boldsymbol{a}_t, r_t) \in D_i^{\text{train}}} A^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t) \nabla_{\theta} \log \pi_{\theta}(\boldsymbol{a}_t | \boldsymbol{s}_t).\end{aligned}$$

- Meta-trainning：

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \sum_{\mathcal{T}_i \sim \mathcal{T}} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}_i, D_i^{\text{test}}),$$

where

$$\mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}_i, D_i^{\text{test}}) = \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}, D_i^{\text{train}}), D_i^{\text{test}}).$$

Review on MAML-RL

- Algorithm

Algorithm 1 MAML Training [10]

Require: $p(\mathcal{T})$: task distribution
Require: α: adaptation learning rate
Require: β: meta learning rate
  Randomly initialize $\boldsymbol{\theta}$
  while not done **do**
    Sample a batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
    for all $\mathcal{T}_i$ in batch **do**
      Sample K trajectories D_i^{train} using $\pi_{\boldsymbol{\theta}}$ on task $\mathcal{T}_i$.
      Compute adapted parameters using D_i^{train}:
      $\boldsymbol{\theta}_i = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}, D_i^{\text{train}})$.
      Sample K trajectories D_i^{test} using $\pi_{\boldsymbol{\theta}_i}$ on task $\mathcal{T}_i$.
    end for
    Update $\boldsymbol{\theta}$ using all D_i^{train}, and D_i^{test}:
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \beta \sum_{\mathcal{T}_i \sim \mathcal{T}} \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}_i, D_i^{\text{test}})$
    with $\mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}_i, D_i^{\text{test}}) = \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}, D_i^{\text{train}}), D_i^{\text{test}})$.
  end while

Algorithm 2 MAML Fine-tuning [10]

Require: $\mathcal{T}_i$: a new test task
Require: $\boldsymbol{\theta}, \alpha$: from MAML training
  Sample K trajectories D using $\pi_{\boldsymbol{\theta}}$ on task $\mathcal{T}_i$.
  Compute fine-tuned policy:
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}, D)$.

Outline

- Background & Motivation
- Review on MAML-RL
- No-Reward Meta Learning
  - Learned Advantage Function
  - Learned Offset
  - Algorithm
- Experiment
  - Point Agent
  - Continuous Control
- Conclusion

No-Reward Meta Learning

- NoRML aims to address the challenge of MAML, consisting of two additional components:
  - A **learned advantage function** that internalizes the reward in a way that allows for reward-free adaptation.
  - A **learned parameter offset** that enables better exploration.

Learned Advantage Function

- Learned Avantage Function(LAF)：$\boxed{A_{\psi}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})}$
- Difference from $\boxed{A^{\pi}(\boldsymbol{s}_t, \boldsymbol{a}_t)} = \sum_{t'=t}^{H} \gamma^{t'-t} r_{t'} - V^{\pi}(\boldsymbol{s}_t)$：
  - A_{ψ} is a feed-forward neural network that takes in consecutive states and action (st,at,st+1) and it's trained end-to-end.
  - A_{ψ} is only used during fine-tuning, while the A^{π} is still used to compute the outer gradient during meta training.
  - A_{ψ} is not a true advantage function, it is optimized to *transform or reshape* the policy gradient $\nabla_{\theta} \log \pi_{\theta}(\boldsymbol{a}_t|\boldsymbol{s}_t)$ that achieves more effective adaptation in a single fine-tune step.

Learned Advantage Function

- The benefit of LAF:
  - Since A_ψ takes in (s_t, a_t, s_{t+1}) as input, this allows A_ψ to detect changes in the dynamics, and provide a more informed evaluation of the actions, compared to using only (s_t, a_t).
  - We eliminate the need to estimate the value function to calculate the observed advantage(during meta-test fine-tuning)
  - A_ψ directly transforms the policy gradient and can provide accurate information to the fine-tune step.
  - A_ψ keeps fixed during meta-test fine-tuning.

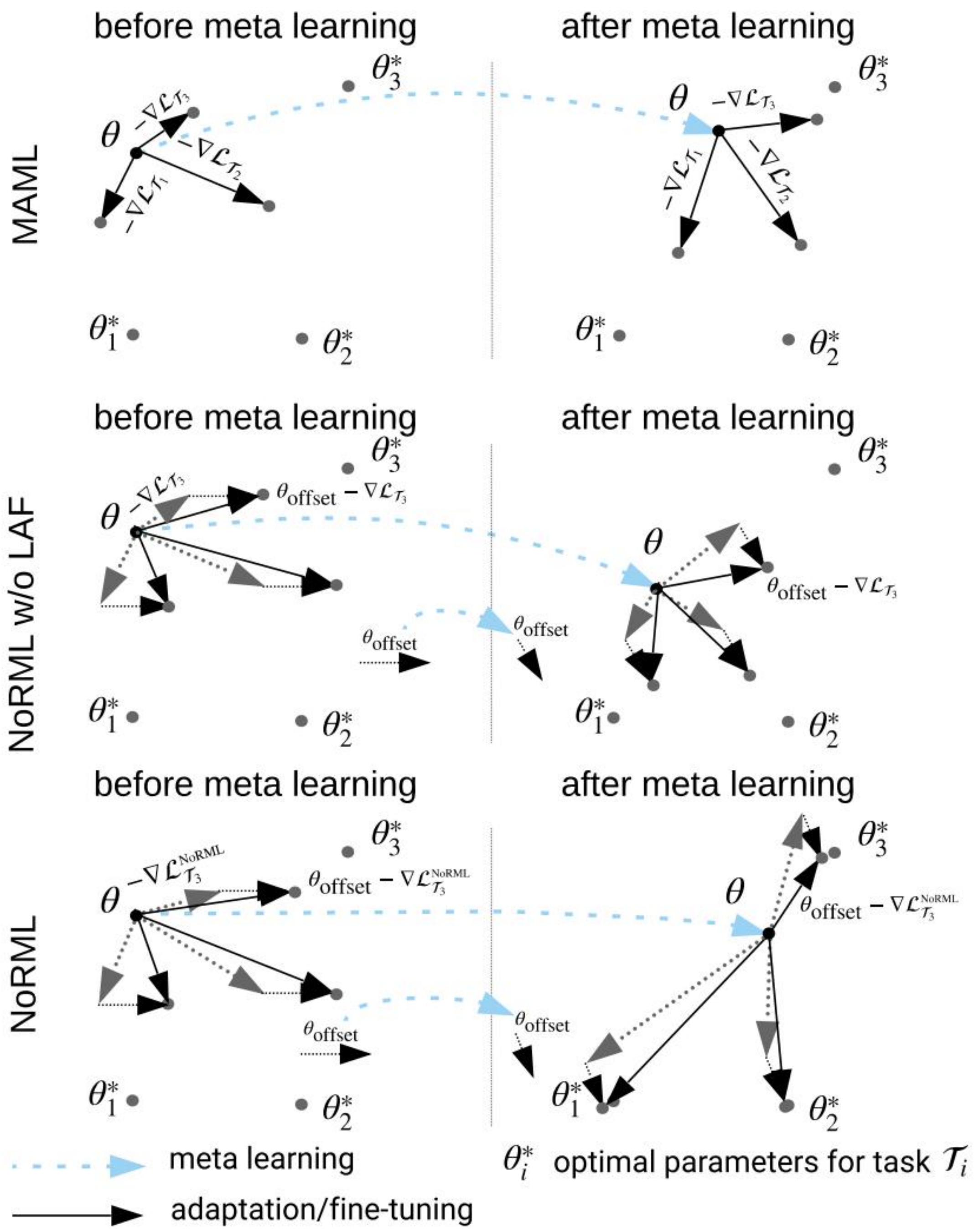
Learned Offset

- One policy gradient step may be insufficient to adapt an exploratory meta-policy into a policy for the new task.
- A learned offset $\vartheta_{\text{offset}}$ that is added to the policy parameters ϑ when calculating an adapted policy:

$$\theta_i = \theta + \theta_{\text{offset}} - \alpha \sum_{D_i^{\text{train}}} A_{\psi}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}) \nabla_{\theta} \log \pi_{\theta}(\boldsymbol{a}_t | \boldsymbol{s}_t).$$

Learned Offset

Geometric interpretations of MAML, NoRML w/o Learned Advantage Function (LAF) and NoRML.



Algorithm

Algorithm 3 NoRML Training

Differences between MAML and NoRML are highlighted.

Require: $p(\mathcal{T})$: distribution over tasks

Require: α, β: step size hyperparameters

Require: θ_σ: initial log standard deviation of meta-policy

Randomly initialize θ_μ and set $\theta = \left[\theta_\mu^T\ \theta_\sigma^T\right]^T$

Randomly initialize ψ

Initialize θ_{offset} to $[0 \ldots 0]^T$

while not done **do**

Sample a batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$

for all $\mathcal{T}_i$ **do**

Sample K trajectories without rewards using π_θ on task $\mathcal{T}_i$ and store all state transitions as a set $D_i^{\text{train}} = \{(s_t, a_t, s_{t+1}) : \forall k < K, \forall t \leq H\}$.

Compute adapted parameters using D_i^{train}:

$\theta_i = \theta + \theta_{\text{offset}} - \alpha \sum_{D_i^{\text{train}}} \boxed{A_\psi(s_t, a_t, s_{t+1})} \nabla_\theta \log \pi_\theta(a_t | s_t)$.

Sample K trajectories D_i^{test} using π_{θ_i} on task $\mathcal{T}_i$.

end for

Update θ, θ_{offset}, and ψ using all D_i^{train}, and D_i^{test}:

$$\begin{bmatrix}\theta \\ \theta_{\text{offset}} \\ \psi\end{bmatrix} \leftarrow \begin{bmatrix}\theta \\ \theta_{\text{offset}} \\ \psi\end{bmatrix} - \beta \sum_{\mathcal{T}_i \sim \mathcal{T}} \nabla_{[\theta^T \theta_{\text{offset}}^T \psi^T]^T} \mathcal{L}_{\mathcal{T}_i}(\theta_i, D_i^{\text{test}})$$

with $\nabla \mathcal{L}_{\mathcal{T}_i}(\theta_i, D_i^{\text{test}}) = \sum_{D_i^{\text{test}}} \boxed{A^\pi(s_t, a_t)} \nabla \log \pi_{\theta_i}(a_t | s_t)$.

end while

Algorithm 4 NoRML Fine-tuning

Require: $\mathcal{T}_i$: a task

Require: θ, θ_{offset}, ψ, α: from NoRML training

Sample K trajectories without rewards using π_θ and store all state transitions as a set $D = \{(s_t, a_t, s_{t+1}) : \forall k < K, \forall t < H\}$.

Compute fine-tuned policy:

$\theta \leftarrow \theta + \theta_{\text{offset}} - \alpha \sum_D \boxed{A_\psi(s_t, a_t, s_{t+1})} \nabla_\theta \log \pi_\theta(a_t | s_t)$.

Note r_t is only used to train A^π. During meta-test fine-tuning, we only apply learned offset and LAF without the requirement of reward.

Outline

- Background & Motivation
- Review on MAML-RL
- No-Reward Meta Learning
  - Learned Advantage Function
  - Learned Offset
  - Algorithm
- Experiment
  - Point Agent
  - Continuous Control
- Conclusion

Experiment

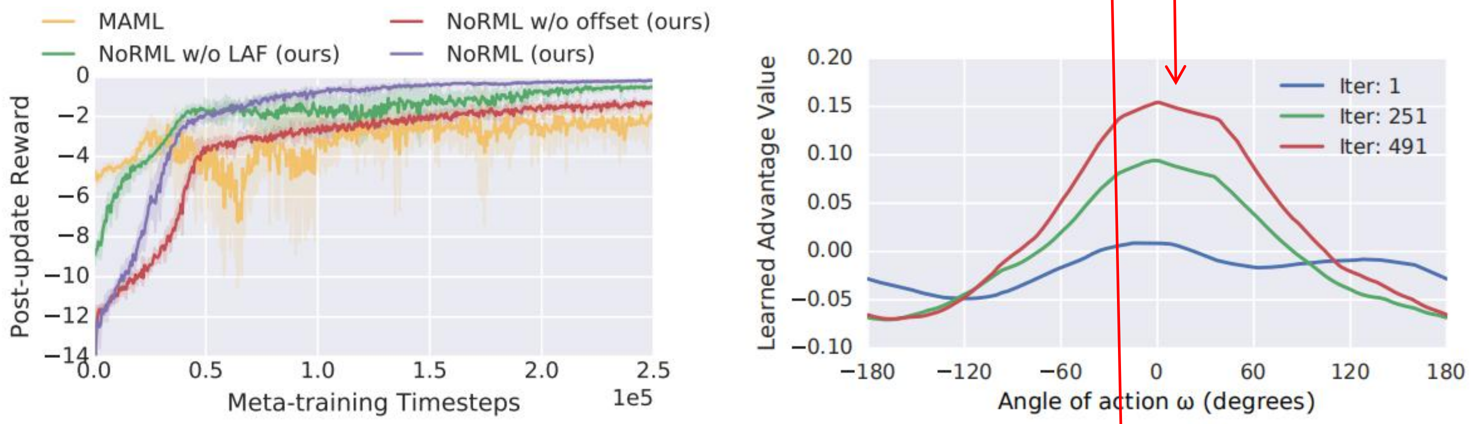
- Comparison objective:
  - vanilla MAML
  - Domain Randomization(DR): Implement DR by setting α and θ_{offset} to zero
  - NoRML w/o offset
  - NoRML w/o LAF
- Some details:
  - Policy: $\pi_\theta(\boldsymbol{a}_t|\boldsymbol{s}_t) = \mathcal{N}(f(\boldsymbol{s}_t|\boldsymbol{\theta}_\mu), \mathrm{diag}(e^{\boldsymbol{\theta}_\sigma})^2)$ $(\boldsymbol{\theta} = \left[\boldsymbol{\theta}_\mu^T \ \boldsymbol{\theta}_\sigma^T\right]^T)$
  - Apply Meta-SGD extension to MAML:replace the fixed inner learning rate of MAML with a learned vector of the same dimension as the policy parameter.
  - Apply PPO for MAML's adaptation and meta objective to improve performance. For NoRML PPO is only used for the meta objective.
  - we use polynomial regression to fit the value function

Point Agent

- Purpose: Study the adaptation to dynamics change and reward change.
- Task: move a point agent from (0,0) to (1,0);
- State: (x_t, y_t); Action:(dx_t, dy_t);
- Dynamics: $\begin{bmatrix} x_{t+1} \\ y_{t+1} \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} dx_t \\ dy_t \end{bmatrix} + \begin{bmatrix} x_t \\ y_t \end{bmatrix}$ (one φ per task);
- Reward setting:
  - Shaped reward case: negative Euclidean distance to (1, 0)
  - Sparse reward case: -1 for each step

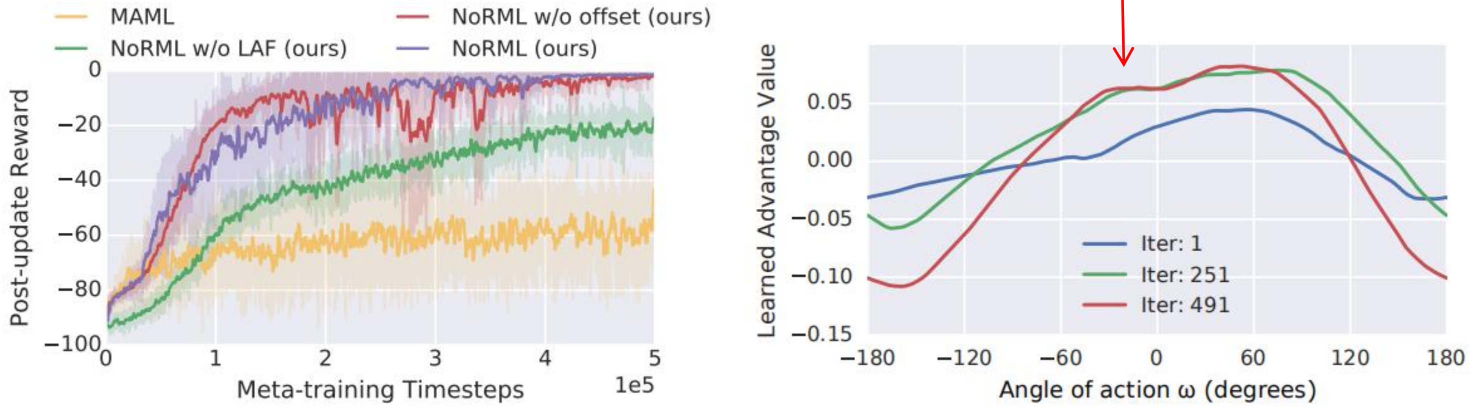
Point Agent

LAF with no rotation



(a) Shaped Reward: Learning Curve

(b) Shaped Reward: Learned Advantage Function



(c) Sparse Reward: Learning Curve

(d) Sparse Reward: Learned Advantage Function

Figure 2

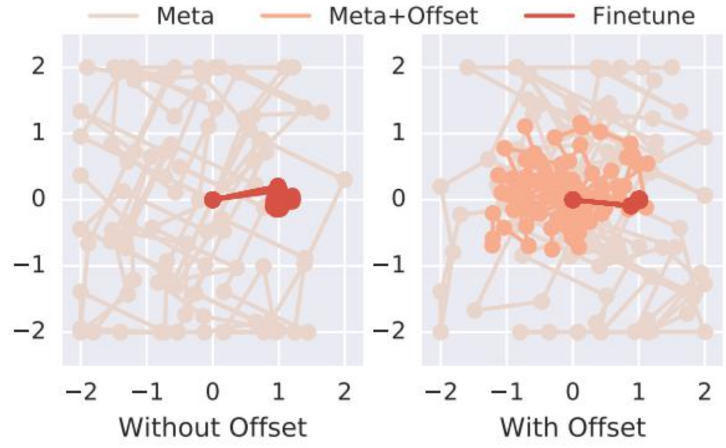


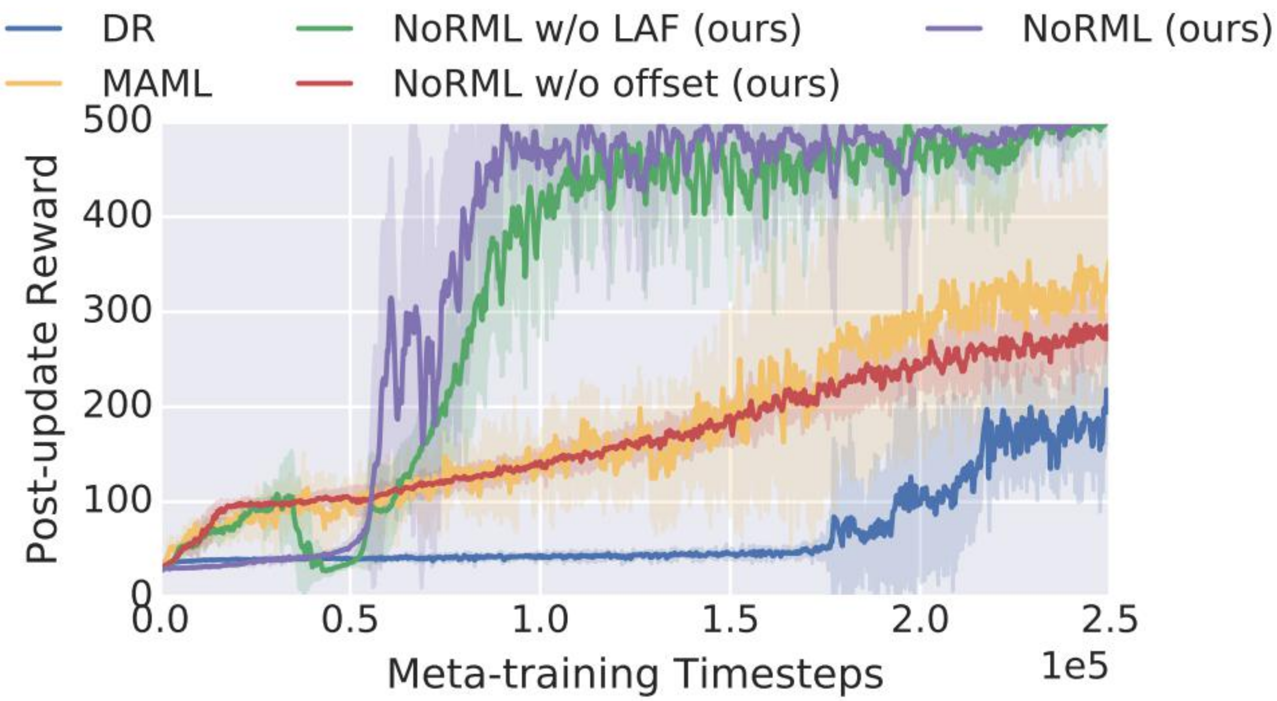
Figure 3: 10 rollout trajectories for NoRML policies trained with and without the offset. "Meta+Offset" means we only add the learned offset θ_{offset} to the meta-policy parameters θ without a gradient step and evaluate policy $\pi_{\theta+\theta_{\text{offset}}}$.

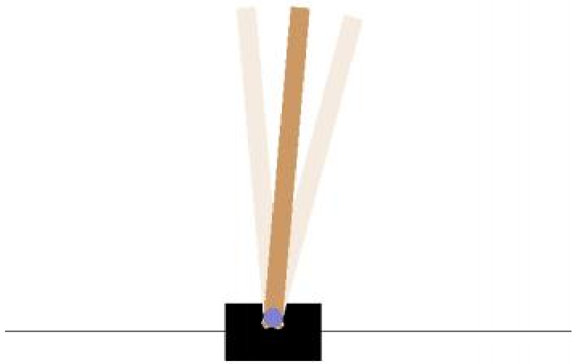
Point Agent

- Impact of LAF:
  - In the shaped-reward case, MAML and NoRML both perform well.(Fig.2a)
  - In the sparse-reward case, MAML's performance degrades dramatically but LAF in NoRML enables the agent to adapt in one policy gradient step.(Fig.2c)
- Impact of the learned offset:
  - Better fine-tuning performance(Fig2a&c)
  - Reduce variance(Fig.3)

Continuous Control

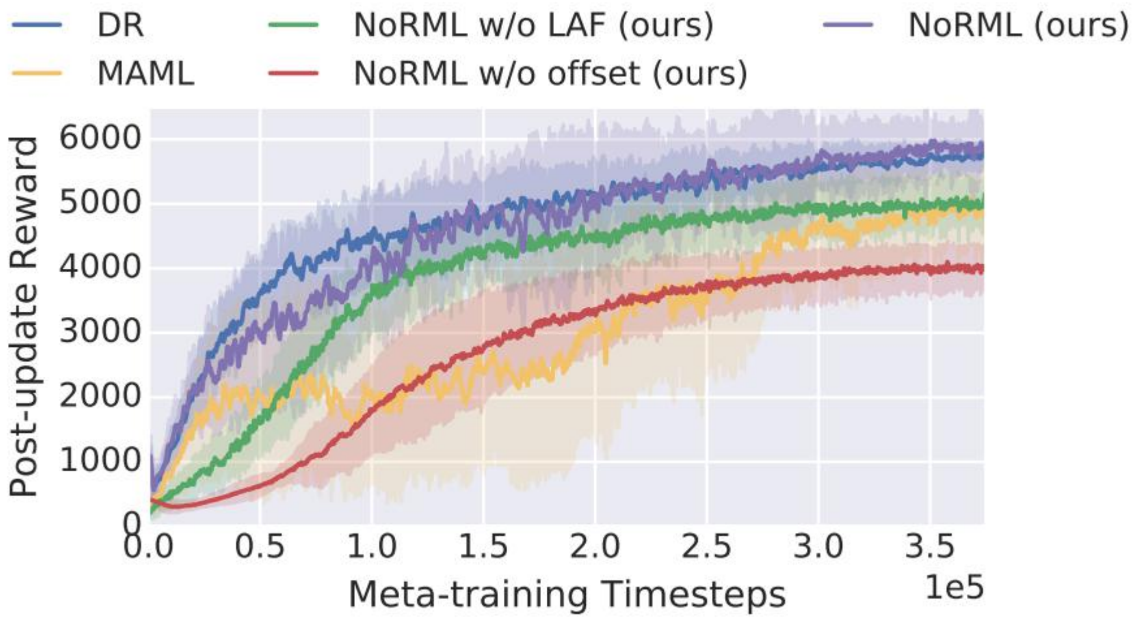
- Experiment 1: Cartpole with sensor bias.





Continuous Control

- Experiment 2: Half Cheetah with Swapped Actions
  - Allow the torque outputs of the two hip joints to be swapped
  - Remove the position and linear velocity from half cheetah’s observation space. (makes it difficult to compute the distance-based reward function)



Continuous Control

- Exp1:
  - NoRML converges faster despite having fewer assumptions——it does not require an external reward signal for adapation.
  - Without the offset, NoRML could not converge to a high final reward, and without the learned advantage function, convergence is slower.
- Exp2:
  - NoRML outperforms MAML both in convergence speed and final return.

Outline

- Background & Motivation
- Review on MAML-RL
- No-Reward Meta Learning
  - Learned Advantage Function
  - Learned Offset
  - Algorithm
- Experiment
  - Point Agent
  - Continuous Control
- Conclusion

Conclusion

- MAML cannot address dynamics change, sensor drift and missing reward signal well.
- By incorporating both a learned advantage function and a learned offset, NoRML can adapt to all these types of changes.
- LAF mainly transforms the policy gradient with more informed evaluation, leading faster and robust adaptation.
- Offset provides an additional adjustment to get better fine-tuning performance.