1803.02999 - On First-Order Meta-Learning Algorithms

On First-Order Meta-Learning Algorithms

Alex Nichol and Joshua Achiam and John Schulman
OpenAI
{alex, jachiam, joschu}@openai.com

- **Yunqiu Xu**

---

1. Introduction

- Challenges:
  - Human can learn a new task quickly through prior knowledge
  - Meta learning can be used to achieve this, however MAML with two-level derivatives can be computational expensive
- Our contribution
  - Expand the work of First-order MAML, which is simpler to implement
  - Reptile: similar to FOMAML but doesn't need to split training-test for each task
  - Analyse both FOMAML and Reptile theritically to show that they optimize for within-task generalization
  - Experiment on Mini-ImageNet and Omniglot

2. Reptile

- 符号:
  - ϕ : 初始参数

- $\tilde{\phi}$: 梯度下降更新后参数
- $\tilde{\phi} = U_\tau^k(\phi)$: 使用任务 τ 更新 k 次参数 ϕ, 即进行 k 次梯度下降, 得到更新后的参数 $\tilde{\phi}$

- MAML:

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
end while

- MAML基于假设: 对于内循环(inner-loop)某个任务, 使用训练集 A 进行训练, 并使用测试集 B 进行测试, 其测试误差会被当作外循环(meta-loop)
- 可以看出MAML在内循环和外循环都需要计算梯度导数 $\rightarrow$ 二阶导数

$$g_{\text{MAML}} = \frac{\partial}{\partial \phi} L_{\tau,B}(U_{\tau,A}(\phi)) \tag{3}$$
$$= U'_{\tau,A}(\phi) L'_{\tau,B}(\widetilde{\phi}), \quad \text{where} \quad \widetilde{\phi} = U_{\tau,A}(\phi) \tag{4}$$

- FOMAML: 仅仅对外循环梯度求导, 不求取内循环梯度的导数, 将 $U'_{\tau,A}$ 看作常数

- Reptile:

Algorithm 1 Reptile (serial version)

Initialize ϕ, the vector of initial parameters
for iteration $= 1, 2, \ldots$ **do**
  Sample task τ, corresponding to loss L_τ on weight vectors $\widetilde{\phi}$
  Compute $\widetilde{\phi} = U_\tau^k(\phi)$, denoting k steps of SGD or Adam
  Update $\phi \leftarrow \phi + \epsilon(\widetilde{\phi} - \phi)$
end for

- 在内循环进行 k 次更新, 得到更新后的参数 $\tilde{\phi}$
- 然后用 $\tilde{\phi}$ 对 ϕ 进行更新

3. Experiment

- Few-shot classification:
  - Transductive setting: 通过batch normalization在test sample之间共享信息
  - Non-transductive setting: 仅仅在training sample上使用batch normalization, 使用single test sample

Algorithm	1-shot 5-way	5-shot 5-way
MAML + Transduction	$48.70 \pm 1.84\%$	$63.11 \pm 0.92\%$
1^{st}-order MAML + Transduction	$48.07 \pm 1.75\%$	$63.15 \pm 0.91\%$
Reptile	$47.07 \pm 0.26\%$	$62.74 \pm 0.37\%$
Reptile + Transduction	$49.97 \pm 0.32\%$	$65.99 \pm 0.58\%$

Table 1: Results on Mini-ImageNet. Both MAML and 1^{st}-order MAML results are from [4].

Algorithm	1-shot 5-way	5-shot 5-way	1-shot 20-way	5-shot 20-way
MAML + Transduction	$98.7 \pm 0.4\%$	$99.9 \pm 0.1\%$	$95.8 \pm 0.3\%$	$98.9 \pm 0.2\%$
1^{st}-order MAML + Transduction	$98.3 \pm 0.5\%$	$99.2 \pm 0.2\%$	$89.4 \pm 0.5\%$	$97.9 \pm 0.1\%$
Reptile	$95.39 \pm 0.09\%$	$98.90 \pm 0.10\%$	$88.14 \pm 0.15\%$	$96.65 \pm 0.33\%$
Reptile + Transduction	$97.68 \pm 0.04\%$	$99.48 \pm 0.06\%$	$89.43 \pm 0.14\%$	$97.12 \pm 0.32\%$

Table 2: Results on Omniglot. MAML results are from [4]. 1^{st}-order MAML results were generated by the code for [4] with the same hyper-parameters as MAML.

- Different inner-loop and outer-loop gradient combinations
  - g_1, g_2, g_3, g_4 : inner-loop gradients with different minibatches
  - $g_1 + g_2$: outer-loop update for two-step Reptile
  - g_2 : outer-loop update for two-step FOMAML

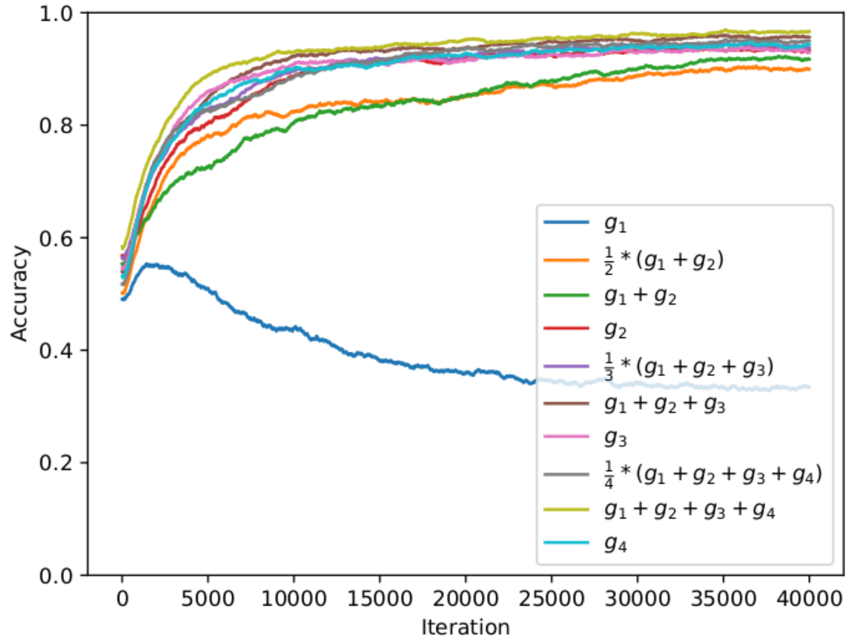
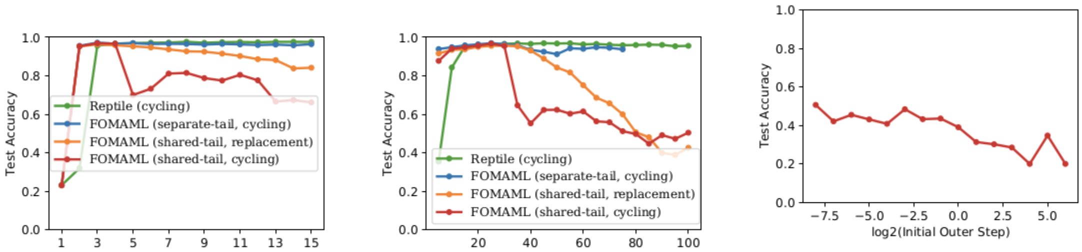


Figure 3: Different inner-loop gradient combinations on 5-shot 5-way Omniglot.

- Overlap between inner-loop minibatches:
  - Aim: check whether small changes to optimization procedure can lead to large changes in performance
  - shared-tail (cycling): final inner-loop mini-batch comes from the same set as earlier inner-loop batches
  - separate-tail (**more correct**) : final mini-batch comes from a disjoint of data



(a) Final test performance vs. number of inner-loop iterations.

(b) Final test performance vs. inner-loop batch size.

(c) Final test performance vs. outer-loop step size for shared-tail FOMAML with batch size 100 (full batches).

Figure 4: The results of hyper-parameter sweeps on 5-shot 5-way Omniglot.

  - result: Reptile and FOMAML with cycling / separate-tail are not sensitive to inner-loop hyper-parameters

4. Summary

- Reptile:
    - 类似joint training和FOMAML
    - 将MAML简化为一阶
    - 和MAML差不多的性能, 但是效率更高
    - 和FOMAML相比, 对每个子任务不需要再split training-testing
- Future work:
    - Reptile在RL中的效果不大好, 作者这里提出原因可能为"joint training is a strong baseline", 未来需要对Reptile进行改进
    - 还有一些未来工作是关于few-shot classification的
- Further reading:
    - Paper repro: Deep Metalearning using “MAML” and “Reptile”
    - Understanding Reptile: A Scalable Meta-learning Algorithm By OpenAI
- Implementations:
    - Pytorch implementation in "Paper repro: Deep Metalearning using 'MAML' and 'Reptile'"
    - TF implementation on new dataset
    - Another pytorch implementation