Meta-Learning with Implicit Gradients

Aravind Rajeswaran, Chelsea Finn, Sham Kakade, Sergey Levine (2019)

Mayank Mittal

Model-Agnostic Meta-Learning

- Learn general purpose internal representation that is transferable across different tasks

$$\mathcal{T}_i \sim P(\mathcal{T}), \forall i = \{1, ..., M\} \Longrightarrow \left.\begin{array}{l} \mathcal{D}_i^{\text{tr}} = \{(x_i^k, y_i^k)\}_{k=1}^{K} \\ \mathcal{D}_i^{\text{test}} = \{(x_i^k, y_i^k)\}_{k=K+1}^{K+N} \end{array}\right\} \text{drawn from } \mathcal{T}_i \text{ task}$$

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

Meta-Learning objective:

$$\overbrace{\boldsymbol{\theta}^*_{\text{ML}} := \underset{\boldsymbol{\theta}\in\Theta}{\text{argmin}}\, F(\boldsymbol{\theta})}^{\text{outer-level}}, \text{ where } F(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}\left(\overbrace{\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}})}^{\text{inner-level}}, \mathcal{D}_i^{\text{test}}\right)$$

$\mathcal{L}$: loss function

$\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}})$: task-specific parameters

Images from: **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks**, Finn et al. (2017)

Model-Agnostic Meta-Learning

$$\overbrace{\boldsymbol{\theta}^*_{\mathrm{ML}} := \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}} F(\boldsymbol{\theta})}^{\mathrm{outer-level}}, \text{ where } F(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}\left(\overbrace{\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}})}^{\mathrm{inner-level}}, \mathcal{D}_i^{\mathrm{test}}\right)$$

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \frac{1}{M}\sum_{i=1}^{M} \frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}(Alg(\boldsymbol{\theta}^k, \mathcal{D}_i^{\mathrm{tr}}), \mathcal{D}_i^{\mathrm{test}})$$

$$\boldsymbol{\phi}_i \equiv \mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}}) = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}})$$

Algorithm 1 Model-Agnostic Meta-Learning

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

Inner Level/ Adaptation Step

Outer Level/ Meta-Optimization Step

Images from: **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks**, Finn et al. (2017)

Model-Agnostic Meta-Learning

$$\overbrace{\boldsymbol{\theta}^*_{\text{ML}} := \underset{\boldsymbol{\theta}\in\Theta}{\text{argmin}}\, F(\boldsymbol{\theta})}^{\text{outer-level}}, \text{ where } F(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}\Big(\overbrace{\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}})}^{\text{inner-level}}, \mathcal{D}_i^{\text{test}}\Big)$$

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \frac{1}{M}\sum_{i=1}^{M} \frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}(\underbrace{\mathcal{A}lg(\boldsymbol{\theta}^k, \mathcal{D}_i^{\text{tr}})}_{\mathcal{A}lg_i(\boldsymbol{\theta}) := \mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}})}, \mathcal{D}_i^{\text{test}})$$

$$\mathcal{L}_i(\boldsymbol{\phi}) := \mathcal{L}(\boldsymbol{\phi}, \mathcal{D}_i^{\text{test}})$$

$$\boldsymbol{\phi}_i \equiv \mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}}) = \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \underbrace{\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}_i^{\text{tr}})}_{\hat{\mathcal{L}}_i(\boldsymbol{\phi}) := \mathcal{L}(\boldsymbol{\phi}, \mathcal{D}_i^{\text{tr}})}$$

$$\implies \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{d}_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta}^k))$$

$$\boldsymbol{d}_{\boldsymbol{\theta}} \mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta})) = \frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}} \nabla_{\boldsymbol{\phi}} \mathcal{L}_i(\boldsymbol{\phi})\,|_{\boldsymbol{\phi}=\mathcal{A}lg_i(\boldsymbol{\theta})} = \underbrace{\frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}}_{(d\times d)\text{–size Jacobian matrix}} \nabla_{\boldsymbol{\phi}} \mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta}))$$

Images from: **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks**, Finn et al. (2017)

Model-Agnostic Meta-Learning

$$\overbrace{\boldsymbol{\theta}^*_{\mathrm{ML}} := \operatorname*{argmin}_{\boldsymbol{\theta}\in\Theta} F(\boldsymbol{\theta})}^{\mathrm{outer-level}}, \text{ where } F(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}\Big(\overbrace{\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}})}^{\mathrm{inner-level}}, \mathcal{D}_i^{\mathrm{test}}\Big)$$

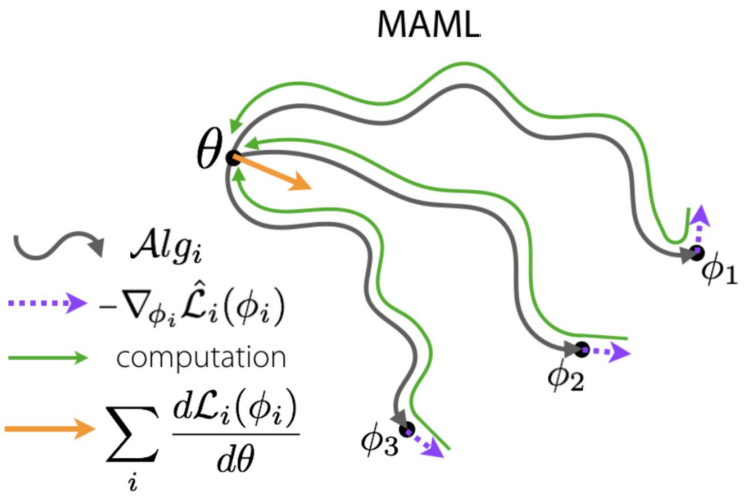
$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{d_\theta}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta}^k))$$

$$\boldsymbol{\phi}_i \equiv \mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}}) = \boldsymbol{\theta} - \alpha\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}})$$

$$\boldsymbol{d_\theta}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta})) = \frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\boldsymbol{\phi})\,|_{\boldsymbol{\phi}=\mathcal{A}lg_i(\boldsymbol{\theta})} = \boxed{\frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta}))$$

$(d \times d)$–size Jacobian matrix

source of problem



Gradients flow over iterative updates of inner-level!

- Need to store optimization path
- Computationally expensive

Hence, heuristics such as first-order MAML and Reptile are used

Images from: **Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks**, Finn et al. (2017)

Motivation

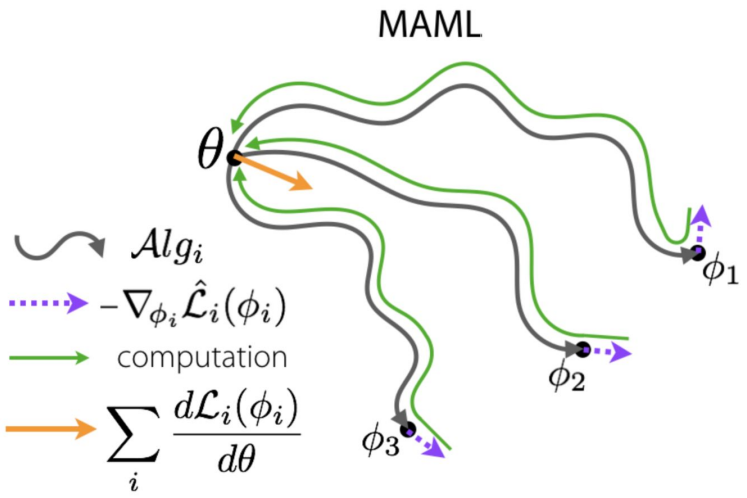
$$\overbrace{\boldsymbol{\theta}^*_{\mathrm{ML}} := \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\, F(\boldsymbol{\theta})}^{\mathrm{outer-level}}, \text{ where } F(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}\left(\overbrace{\mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}})}^{\mathrm{inner-level}}, \mathcal{D}_i^{\mathrm{test}}\right)$$

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{d_\theta}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta}^k))$$

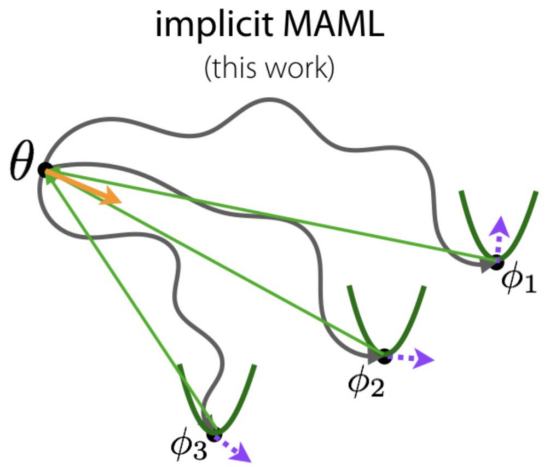
$$\boldsymbol{\phi}_i \equiv \mathcal{A}lg(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}}) = \boldsymbol{\theta} - \alpha\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}_i^{\mathrm{tr}})$$

$$\boldsymbol{d_\theta}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta})) = \frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\boldsymbol{\phi})\,|_{\boldsymbol{\phi}=\mathcal{A}lg_i(\boldsymbol{\theta})} = \boxed{\frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta}))$$

$(d \times d)$–size Jacobian matrix



make this path independent?



Implicit MAML (iMAML)

- In MAML, $\phi_i \equiv Alg(\theta, \mathcal{D}_i^{\text{tr}})$ is an iterative gradient descent
  - dependence of model-parameters $\{\phi_i\}$ on meta-parameters θ shrinks and vanishes as number of gradient steps increases
- Instead, iMAML uses an optimum solution of the regularized loss

$$Alg^{\star}(\theta, \mathcal{D}_i^{\text{tr}}) = \underset{\phi' \in \Phi}{\text{argmin}}\, \mathcal{L}(\phi', \mathcal{D}_i^{\text{tr}}) + \frac{\lambda}{2} ||\phi' - \theta||^2$$

encourages dependency between model-parameters and meta-parameters

Implicit MAML (iMAML)

$$\boldsymbol{\theta}^*_{\mathrm{ML}} := \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\, F(\boldsymbol{\theta})\,, \text{ where } F(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} \mathcal{L}_i\big(\mathcal{A}lg_i^\star(\boldsymbol{\theta})\big), \text{ and}$$

$$\mathcal{A}lg_i^\star(\boldsymbol{\theta}) := \underset{\boldsymbol{\phi}'\in\Phi}{\operatorname{argmin}}\, G_i(\boldsymbol{\phi}', \boldsymbol{\theta}), \text{ where } G_i(\boldsymbol{\phi}', \boldsymbol{\theta}) = \hat{\mathcal{L}}_i(\boldsymbol{\phi}') + \frac{\lambda}{2}\,||\boldsymbol{\phi}' - \boldsymbol{\theta}||^2$$

Issue: Still the meta-gradient requires to computation of the Jacobian, i.e.

$$\boldsymbol{d_\theta}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta})) = \frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\boldsymbol{\phi})\,|_{\boldsymbol{\phi}=\mathcal{A}lg_i(\boldsymbol{\theta})} = \boxed{\frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta}))$$

$(d \times d)$–size Jacobian matrix

Soln: Use implicit Jacobian! 🙂

$$\frac{d\mathcal{A}lg_i^\star(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \left(\boldsymbol{I} + \frac{1}{\lambda}\,\nabla^2_{\boldsymbol{\phi}}\hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)^{-1}$$

Implicit Jacobian Lemma

Lemma 1. *(Implicit Jacobian) Consider $\mathcal{A}lg_i^\star(\boldsymbol{\theta})$ as defined in Eq. 4 for task $\mathcal{T}_i$. Let $\boldsymbol{\phi}_i = \mathcal{A}lg_i^\star(\boldsymbol{\theta})$ be the result of $\mathcal{A}lg_i^\star(\boldsymbol{\theta})$. If $\left(\boldsymbol{I} + \frac{1}{\lambda}\nabla^2_{\boldsymbol{\phi}}\hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)$ is invertible, then the derivative Jacobian is*

$$\frac{d\mathcal{A}lg_i^\star(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \left(\boldsymbol{I} + \frac{1}{\lambda}\nabla^2_{\boldsymbol{\phi}}\hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)^{-1}. \tag{6}$$

where $\mathcal{A}lg_i^\star(\boldsymbol{\theta}) := \underset{\boldsymbol{\phi}' \in \Phi}{\operatorname{argmin}}\, G_i(\boldsymbol{\phi}', \boldsymbol{\theta}),$ where $G_i(\boldsymbol{\phi}', \boldsymbol{\theta}) = \hat{\mathcal{L}}_i(\boldsymbol{\phi}') + \frac{\lambda}{2}\,||\boldsymbol{\phi}' - \boldsymbol{\theta}||^2$

Proof. Apply stationary point condition:

$$\nabla_{\boldsymbol{\phi}'} G(\boldsymbol{\phi}', \boldsymbol{\theta})\,|_{\boldsymbol{\phi}'=\boldsymbol{\phi}} = 0 \implies \nabla\hat{\mathcal{L}}(\boldsymbol{\phi}) + \lambda(\boldsymbol{\phi} - \boldsymbol{\theta}) = 0 \implies \boldsymbol{\phi} = \boldsymbol{\theta} - \frac{1}{\lambda}\nabla\hat{\mathcal{L}}(\boldsymbol{\phi}),$$

which is an implicit equation that often arises in proximal point methods. When the derivative exists, we can differentiate the above equation to obtain:

$$\frac{d\boldsymbol{\phi}}{d\boldsymbol{\theta}} = \boldsymbol{I} - \frac{1}{\lambda}\nabla^2\hat{\mathcal{L}}(\boldsymbol{\phi})\frac{d\boldsymbol{\phi}}{d\boldsymbol{\theta}} \implies \left(\boldsymbol{I} + \frac{1}{\lambda}\nabla^2\hat{\mathcal{L}}(\boldsymbol{\phi})\right)\frac{d\boldsymbol{\phi}}{d\boldsymbol{\theta}} = \boldsymbol{I}.$$

Practical Algorithm

Meta gradient: $$d_{\theta}\mathcal{L}_i(\mathcal{A}lg_i(\theta)) = \frac{d\mathcal{A}lg_i(\theta)}{d\theta}\nabla_{\phi}\mathcal{L}_i(\phi)\mid_{\phi=\mathcal{A}lg_i(\theta)} = \boxed{\frac{d\mathcal{A}lg_i(\theta)}{d\theta}}\nabla_{\phi}\mathcal{L}_i(\mathcal{A}lg_i(\theta))$$

$$\frac{d\mathcal{A}lg_i^{\star}(\theta)}{d\theta} = \left(\boldsymbol{I} + \frac{1}{\lambda}\nabla_{\phi}^2\hat{\mathcal{L}}_i(\phi_i)\right)^{-1}$$

Issues:

- obtaining exact solution to inner level optimization problem may be not feasible, in practice
- computing Jacobian by explicitly forming and inverting the matrix may be intractable for large networks

Soln: Use approximations! 🙂

Practical Algorithm

Meta gradient: $d_{\boldsymbol{\theta}}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta})) = \frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\boldsymbol{\phi})\mid_{\boldsymbol{\phi}=\mathcal{A}lg_i(\boldsymbol{\theta})} = \frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta}))$

$\boldsymbol{g}_i$

$$\frac{d\mathcal{A}lg_i^{\star}(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \left(\boldsymbol{I} + \frac{1}{\lambda}\nabla^2_{\boldsymbol{\phi}}\hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)^{-1}$$

Soln: Use approximations! 🙂

1. Approximate algorithm solution:

   Let $\mathcal{A}lg_i(\boldsymbol{\theta})$ *be a* δ*–accurate approximation of* $\mathcal{A}lg_i^{\star}(\boldsymbol{\theta})$*, i.e.* $\|\mathcal{A}lg_i(\boldsymbol{\theta}) - \mathcal{A}lg_i^{\star}(\boldsymbol{\theta})\| \leq \delta$

2. Approximate matrix inversion product:

   Let $\boldsymbol{g}_i$ *be a vector such that* $\|\boldsymbol{g}_i - \left(\boldsymbol{I} + \frac{1}{\lambda}\nabla^2_{\boldsymbol{\phi}}\hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)^{-1}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\boldsymbol{\phi}_i)\| \leq \delta'$

   where $\boldsymbol{\phi}_i = \mathcal{A}lg_i(\boldsymbol{\theta})$ *and* $\mathcal{A}lg_i$ *is based on definition 1.*

   ⇕

   Solved using Conjugate Gradient (CG) method ← $\min_{\boldsymbol{w}} \boldsymbol{w}^{\top}\left(\boldsymbol{I} + \frac{1}{\lambda}\nabla^2_{\boldsymbol{\phi}}\hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)\boldsymbol{w} - \boldsymbol{w}^{\top}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\boldsymbol{\phi}_i)$

Practical Algorithm

Implicit gradient accuracy: (valid under other regularity assumptions)

Assuming that $\delta < \mu/(2\rho)$, *we have that:*

$$\|\boldsymbol{g}_i - \boldsymbol{d}_{\boldsymbol{\theta}}\mathcal{L}_i(\mathcal{A}lg_i^\star(\boldsymbol{\theta}))\| \leq \left(2\frac{\lambda\rho}{\mu^2}B + \frac{\lambda L}{\mu}\right)\delta + \delta'$$

Soln: Use approximations! 🙂

1. Approximate algorithm solution:

   Let $\mathcal{A}lg_i(\boldsymbol{\theta})$ *be a* δ*–accurate approximation of* $\mathcal{A}lg_i^\star(\boldsymbol{\theta})$, *i.e.* $\|\mathcal{A}lg_i(\boldsymbol{\theta}) - \mathcal{A}lg_i^\star(\boldsymbol{\theta})\| \leq \delta$

2. Approximate matrix inversion product:

   Let $\boldsymbol{g}_i$ *be a vector such that* $\|\boldsymbol{g}_i - \left(\boldsymbol{I} + \frac{1}{\lambda}\nabla^2_{\boldsymbol{\phi}}\hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)^{-1}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\boldsymbol{\phi}_i)\| \leq \delta'$

   where $\boldsymbol{\phi}_i = \mathcal{A}lg_i(\boldsymbol{\theta})$ *and* $\mathcal{A}lg_i$ *is based on definition 1.*

   ⇕

   Solved using Conjugate Gradient (CG) method ← $\min_{\boldsymbol{w}} \boldsymbol{w}^\top\left(\boldsymbol{I} + \frac{1}{\lambda}\nabla^2_{\boldsymbol{\phi}}\hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)\boldsymbol{w} - \boldsymbol{w}^\top\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\boldsymbol{\phi}_i)$

Implicit MAML Algorithm

Algorithm 1 Implicit Model-Agnostic Meta-Learning (iMAML)

Require: Distribution over tasks $P(\mathcal{T})$, outer step size η, regularization strength λ,
while not converged **do**
  Sample mini-batch of tasks $\{\mathcal{T}_i\}_{i=1}^{B} \sim P(\mathcal{T})$
  for Each task $\mathcal{T}_i$ **do**
    Compute task meta-gradient $\boldsymbol{g}_i = \texttt{Implicit-Meta-Gradient}(\mathcal{T}_i, \boldsymbol{\theta}, \lambda)$
  end for
  Average above gradients to get $\hat{\nabla} F(\boldsymbol{\theta}) = (1/B) \sum_{i=1}^{B} \boldsymbol{g}_i$
  Update meta-parameters with gradient descent: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \hat{\nabla} F(\boldsymbol{\theta})$ *// (or Adam)*
end while

Outer Level/ Meta-Optimization Step

Algorithm 2 Implicit Meta-Gradient Computation

Input: Task $\mathcal{T}_i$, meta-parameters $\boldsymbol{\theta}$, regularization strength λ
Hyperparameters: Optimization accuracy thresholds δ and δ'
Obtain task parameters $\boldsymbol{\phi}_i$ using iterative optimization solver such that: $\|\boldsymbol{\phi}_i - \mathcal{A}lg_i^{\star}(\boldsymbol{\theta})\| \leq \delta$
Compute partial outer-level gradient $\boldsymbol{v}_i = \nabla_{\boldsymbol{\phi}} \mathcal{L}_{\mathcal{T}}(\boldsymbol{\phi}_i)$
Use an iterative solver (e.g. CG) along with reverse mode differentiation (to compute Hessian vector products) to compute $\boldsymbol{g}_i$ such that: $\|\boldsymbol{g}_i - \left(\boldsymbol{I} + \frac{1}{\lambda} \nabla^2 \hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)^{-1} \boldsymbol{v}_i\| \leq \delta'$
Return: $\boldsymbol{g}_i$

Inner Level/ Adaptation Step

Computation Bounds

Theory: iMAML computes meta-gradient by small error in memory efficient way

Algorithm	Compute	Memory	Error
MAML (GD + full back-prop)	$\kappa \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla\hat{\mathcal{L}}_i) \cdot \kappa \log\left(\frac{D}{\delta}\right)$	0
MAML (Nesterov's AGD + full back-prop)	$\sqrt{\kappa} \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla\hat{\mathcal{L}}_i) \cdot \sqrt{\kappa} \log\left(\frac{D}{\delta}\right)$	0
Truncated back-prop [53] (GD)	$\kappa \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla\hat{\mathcal{L}}_i) \cdot \kappa \log\left(\frac{1}{\epsilon}\right)$	ϵ
Implicit MAML (this work)	$\sqrt{\kappa} \log\left(\frac{D}{\delta}\right)$	$\text{Mem}(\nabla\hat{\mathcal{L}}_i)$	δ

Experiment:

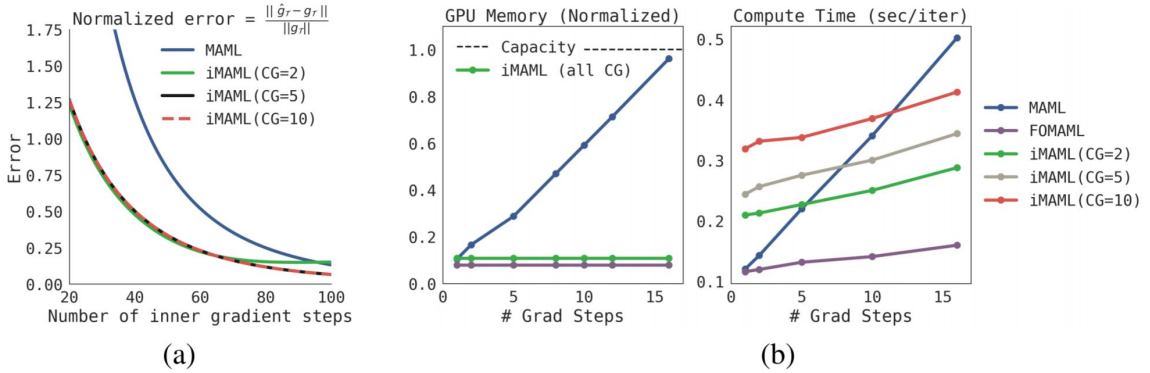


Figure 2: Accuracy, Computation, and Memory tradeoffs of iMAML, MAML, and FOMAML. (a) Meta-gradient accuracy level in synthetic example. Computed gradients are compared against the exact meta-gradient per Def 3. (b) Computation and memory trade-offs with 4 layer CNN on 20-way-5-shot Omniglot task. We implemented iMAML in PyTorch, and for an apples-to-apples comparison, we use a PyTorch implementation of MAML from: https://github.com/dragen1860/MAML-Pytorch

Results on few-shots learning

Table 2: Omniglot results. MAML results are taken from the original work of Finn et al. [15], and first-order MAML and Reptile results are from Nichol et al. [43]. iMAML with gradient descent (GD) uses 16 and 25 steps for 5-way and 20-way tasks respectively. iMAML with Hessian-free uses 5 CG steps to compute the search direction and performs line-search to pick step size. Both versions of iMAML use $\lambda = 2.0$ for regularization, and 5 CG steps to compute the task meta-gradient.

Algorithm	5-way 1-shot	5-way 5-shot	20-way 1-shot	20-way 5-shot
MAML [15]	$98.7 \pm 0.4\%$	$\mathbf{99.9 \pm 0.1\%}$	$95.8 \pm 0.3\%$	$98.9 \pm 0.2\%$
first-order MAML [15]	$98.3 \pm 0.5\%$	$99.2 \pm 0.2\%$	$89.4 \pm 0.5\%$	$97.9 \pm 0.1\%$
Reptile [43]	$97.68 \pm 0.04\%$	$99.48 \pm 0.06\%$	$89.43 \pm 0.14\%$	$97.12 \pm 0.32\%$
iMAML, GD (ours)	$99.16 \pm 0.35\%$	$99.67 \pm 0.12\%$	$94.46 \pm 0.42\%$	$98.69 \pm 0.1\%$
iMAML, Hessian-Free (ours)	$\mathbf{99.50 \pm 0.26\%}$	$99.74 \pm 0.11\%$	$\mathbf{96.18 \pm 0.36\%}$	$\mathbf{99.14 \pm 0.1\%}$

Table 3: Mini-ImageNet 5-way-1-shot accuracy

Algorithm	5-way 1-shot
MAML	48.70 ± 1.84 %
first-order MAML	48.07 ± 1.75 %
Reptile	49.97 ± 0.32 %
iMAML GD (ours)	48.96 ± 1.84 %
iMAML HF (ours)	49.30 ± 1.88 %

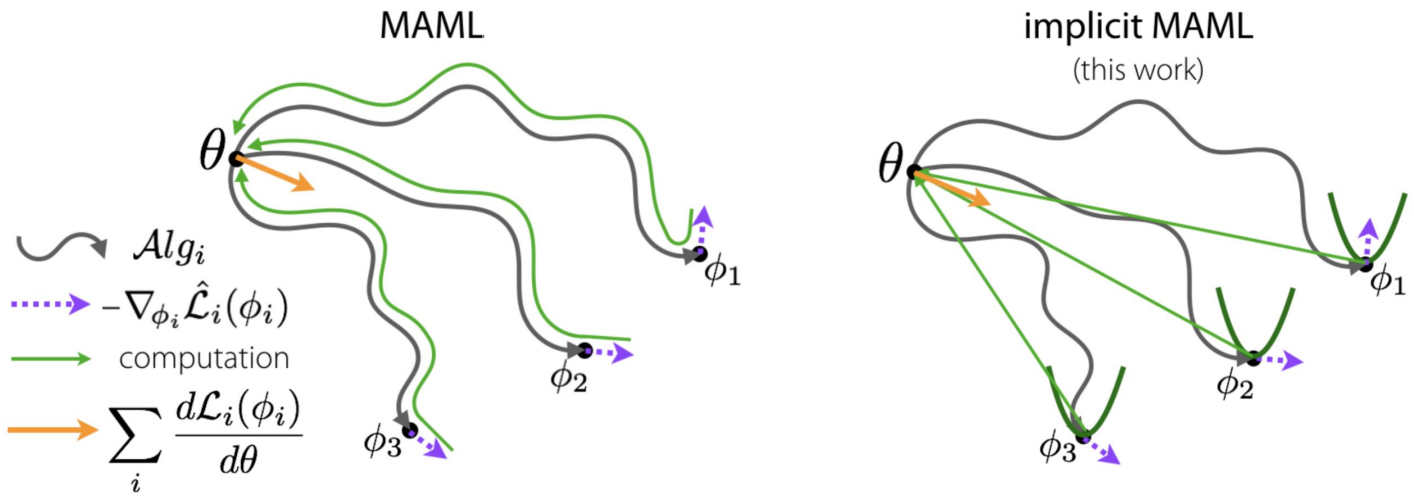
Future Work

- Broader classes for inner loop optimization
  - dynamic programming
  - energy-based models
  - actor-critic RL
- More flexible regularizers
  - Learn a vector- or matrix-valued λ (regularization coefficient)

References

- Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks, Finn et al. (2017)
- Meta-Learning with Implicit Gradients, Rajeswaran et al. (2019)
- Notes on iMAML, Ferenc Huszar (2019)

Thank you!


MAML
implicit MAML
(this work)
θ
ϕ_1
ϕ_2
ϕ_3
Alg_i
$-\nabla_{\phi_i}\hat{\mathcal{L}}_i(\phi_i)$
computation
$\sum_i \frac{d\mathcal{L}_i(\phi_i)}{d\theta}$