Meta-Learning with Implicit Gradients

Aravind Rajeswaran[*,1] Chelsea Finn[*,2] Sham Kakade[1] Sergey Levine[2]
[1] University of Washington Seattle [2] University of California Berkeley

【机器学习 94】iMAML

张楚珩
清华大学 交叉信息院博士在读

iMAML 全称是 implicit model-agnostic meta learning。

原文传送门

Rajeswaran, Aravind, et al. "Meta-Learning with Implicit Gradients." arXiv preprint arXiv:1909.04630 (2019).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.（重要的前序工作 MAML）

Duan, Yan, et al. "Rl $^ 2$: Fast reinforcement learning via slow reinforcement learning." arXiv preprint arXiv:1611.02779 (2016).（另一种比较有代表性的 meta-learning 的算法 RL2）

特色

首先，model-agnostic meta learning （MAML）代表了一类重要的 meta-learning 的算法。这类算法分为内层和外层两部分：内层为普通的学习算法，即学习一个模型 ϕ 来最小化某个损失函数 L ；外层从不同的任务中学习一个能供内层使用的一个先验 θ 。本专栏一直没来得及写 MAML，这里顺带记录一下。

其次，这篇文章旨在解决 MAML 中存在的一些问题：由于 MAML 需要通过 $\theta \leftrightarrow \phi \leftrightarrow L$ 的路径来优化先验 θ （通常是计算 θ 的梯度），因此 MAML 的内层算法需要有额外的计算和存储来追踪 $\theta \leftrightarrow \phi$ 之间的关系，同时也限定了内层算法能够使用的优化方法；同时，通常把 θ 作为神经网络的参数初始化，如果 gradient descent （GD） 的步骤数目较多，则 $\theta \leftrightarrow \phi$ 之间的关系会比较弱，这会导致梯度消失。为了解决上述问题，这篇文章提出了一个新的损失函数 L' 和相应的计算 θ 梯度的方法，使得只需要知道关于该损失函数的解，而不需要知道其具体优化的方法，就可以得到 θ 的梯度。

过程

1. 背景

Meta-learning 的大体想法是通过见识不同的任务，来从中学习一个先验，使得结合该先验的模型在一个新任务到来的时候能够学得更快（用更少的样本）、更好（具有更好的泛化能力）。通常，有以下几大类方法：

- 学习一个好的 embedding space，好的 embedding space 能够使得 nearest neighbors 类（non-parametric）的方法在其中也能较好地工作。
- 学习一个好的模型参数初始化，一个好的参数初始化能够 fast adapt，并且产生更好的泛化。MAML 就属于这一类。
- 学习一个『黑盒子』，这类黑盒子一般是 RNN 网络或者 attention-based NN。以 RNN 为例，这类算法的外层为 RNN 的训练，学习到的先验为 RNN 的网络参数，内层为一个固定参数的 RNN 接受一个序列的输入并且给出相应的输出，内层的学习反映在 RNN 的 hidden state 中。比较具有代表性的算法是 RL2。

2. 一个数学问题

我们先从一个简单的数学问题开始（这个数学问题是从 Math StackExchange 上看来的）。该问题描述如下：考虑 $h(x)=arg\min_y g(x,y)$ ，求 $h'(x)$ 。其中有一些关于函数 g 的 regularity 条件（比如连续、存在最小值等）略去。

解法也简单，由于是最小值，因此对于任意 x，都有 $\partial_2 g(x,h(x))=0$ 。其中， ∂_1,∂_2 分别代表对于函数的第一个、第二个参数求偏导。因此， $\partial_2 g(x,h(x))$ 对于 x 是『平』的，因此，上述函数对 x 求导，应该也为零。即

$$\partial_1\partial_2 g(x,h(x))+h'(x)\partial_2^2 g(x,h(x))=0$$

由此，可以求得 $h'(x)=\frac{\partial_1\partial_2 g(x,h(x))}{\partial_2^2 g(x,h(x))}$ 。

3. 问题设定

考虑一个 meta-learning 的设定，假定一个任务的分布 $\mathcal{P}(\mathcal{T})$ 。在 meta-train 的时候，从该分布中采样一些任务 $\{\mathcal{T}_i\}_{i=1}^M=\{(\mathcal{D}_i^{tr},\mathcal{D}_i^{test})\}_{i=1}^M$ 。给定一个任务，内层算法根据先验 θ 和训练数据集 $\mathcal{D}_i^{tr}$ ，训练得到一个模型；外层算法寻找一个先验，使得在不同的任务上学习到的模型在相应的测试数据集 $\mathcal{D}_i^{test}$ 表现都比较好。即，

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

内层算法的优化目标很重要。比如，如果像 MAML 那样使用 GD 去最小化 MSE，如果做多步 GD，内层算法得到的结果就和先验 θ 关系不大了，这就可能导致关于 θ 的梯度消失。为了解决此问题，文章直接显式地在内层的优化目标上加上了关于先验 θ 的正则项：

$$\mathcal{A}lg^\star(\boldsymbol{\theta},\mathcal{D}_i^{\mathrm{tr}}) = \underset{\boldsymbol{\phi}'\in\Phi}{\operatorname{argmin}}\ \mathcal{L}(\boldsymbol{\phi}',\mathcal{D}_i^{\mathrm{tr}}) + \frac{\lambda}{2}\,||\boldsymbol{\phi}'-\boldsymbol{\theta}||^2. \tag{3}$$

总结一下，该 meta-learning 方法主要流程如下：

$$\begin{aligned}&\boldsymbol{\theta}_{\mathrm{ML}}^* := \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmin}}\,\boxed{F(\boldsymbol{\theta})},\ \text{where } F(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M}\boxed{\mathcal{L}_i\big(\mathcal{A}lg_i^\star(\boldsymbol{\theta})\big)},\ \text{and}\\ &\mathcal{A}lg_i^\star(\boldsymbol{\theta}) := \underset{\boldsymbol{\phi}'\in\Phi}{\operatorname{argmin}}\ G_i(\boldsymbol{\phi}',\boldsymbol{\theta}),\ \text{where } G_i(\boldsymbol{\phi}',\boldsymbol{\theta}) = \hat{\mathcal{L}}_i(\boldsymbol{\phi}') + \frac{\lambda}{2}\,||\boldsymbol{\phi}'-\boldsymbol{\theta}||^2.\end{aligned} \tag{4}$$

其中

$$\mathcal{L}_i(\boldsymbol{\phi}) := \mathcal{L}\big(\boldsymbol{\phi},\ \mathcal{D}_i^{\mathrm{test}}\big),\quad \hat{\mathcal{L}}_i(\boldsymbol{\phi}) := \mathcal{L}\big(\boldsymbol{\phi},\mathcal{D}_i^{\mathrm{tr}}\big),\quad \mathcal{A}lg_i(\boldsymbol{\theta}) := \mathcal{A}lg\big(\boldsymbol{\theta},\mathcal{D}_i^{\mathrm{tr}}\big).$$

通常而言，希望用梯度方法来学习 θ ，因此需要求红框（即，蓝框）部分关于 θ 的梯度。根据链式法则，有

$$\boldsymbol{d}_{\boldsymbol{\theta}}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta}))=\frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\boldsymbol{\phi})\mid_{\boldsymbol{\phi}=\mathcal{A}lg_i(\boldsymbol{\theta})}=\frac{d\mathcal{A}lg_i(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\mathcal{A}lg_i(\boldsymbol{\theta}))$$

注意到，最右边乘积第一项为 $d_\theta \times d_\phi$ 的 Jacobian 矩阵，光是把它写出来空间复杂度就比较高（考虑到 ϕ 代表神经网络参数）。

因此，如果用梯度方法来学习 θ ，更新公式为

$$\boldsymbol{\theta}\leftarrow\boldsymbol{\theta}-\eta\,\frac{1}{M}\sum_{i=1}^{M}\frac{d\mathcal{A}lg_i^\star(\boldsymbol{\theta})}{d\boldsymbol{\theta}}\,\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\mathcal{A}lg_i^\star(\boldsymbol{\theta})). \tag{5}$$

4. Implicit MAML

下面这一步是文章最为精华的：即，把 $G_i(\theta,\phi')$ 和前面那个数学问题中的 $g(x,y)$ 相对应，就能够直接得到以下引理。

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 $\left(\boldsymbol{I}+\frac{1}{\lambda}\nabla_{\boldsymbol{\phi}}^2\hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)$ is invertible, then the derivative Jacobian is*

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

参考 张楚珩：【数值分析 2.2】Conjugate Gradient ，可以知道 $A^{-1}b$ 类问题可以转化为如下形式，

$$\min_{\boldsymbol{w}}\;\boldsymbol{w}^\top\left(\boldsymbol{I}+\frac{1}{\lambda}\,\nabla_{\boldsymbol{\phi}}^2\hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)\boldsymbol{w}-\boldsymbol{w}^\top\nabla_{\boldsymbol{\phi}}\mathcal{L}_i(\boldsymbol{\phi}_i) \tag{7}$$

然后使用 conjugate gradient （CG）方法来解决。

总体算法如下：

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

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: $\boxed{\|\boldsymbol{\phi}_i - \mathcal{A}lg_i^{\star}(\boldsymbol{\theta})\| \leq \delta}$
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: $\boxed{\|\boldsymbol{g}_i - \left(\boldsymbol{I} + \frac{1}{\lambda} \nabla^2 \hat{\mathcal{L}}_i(\boldsymbol{\phi}_i)\right)^{-1} \boldsymbol{v}_i\| \leq \delta'}$
Return: $\boldsymbol{g}_i$

5. Inexact case

前面都假设内层算法能够找到准确的解，同时 CG 也能找到准确的关于 θ 的梯度，文章还分析了如果这些步骤存在误差时的情形。

发布于 2019-09-19