Model-Agnostic Meta-Learning （MAML）模型介绍及算法详解

徐不知
生如逆旅，一苇以航。

1,218 人赞同了该文章

MAML在学术界已经是非常重要的模型了，论文**Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks**自2017年发表至今已经收获了400+的引用。由于当前网上关于MAML的中文介绍少之又少，可能很多小伙伴对其还不是特别理解。所以今天我整理了这段时间来的学习心得，与大家分享自己对MAML的认识与理解。MAML可以用于Supervised Regression and Classification以及Reinforcement Learning。由于我对强化学习不是特别了解，因此这篇文章，均是基于MAML在Supervised Regression and Classification中的运用。

一、一些相关概念的介绍

在原论文中，作者直接引用了许多元学习相关的概念，例如 **meta-learning, model-agnostic, N-way K-shot, tasks**等等，其中有些概念在MAML中还有特殊的含义。在此，我尽量用通俗易懂的方式对这些概念为大家做一个介绍。

(1) meta-learning

meta-learning即元学习，也可以称为“**learning to learn**”。常见的深度学习模型，目的是学习一个用于预测的数学模型。而元学习面向的不是学习的结果，而是学习的过程。其学习的不是一个直接用于预测的数学模型，而是学习“如何更快更好地学习一个数学模型”。

举一个现实生活的例子。我们教小朋友读英语时，可以直接让他们模仿apple、banana的发音。但是他们很快又会遇到新的单词，例如strawberry，这是小朋友就需要重新听你的发音，才能正确地读出这个新单词。我们换一种方式，这一次我们不教每个单词的发音，而是教音标的发音。从此小朋友再遇见新单词，他们只要根据音标，就可以正确地读出这个单词。学习音标的过程，正是一个元学习的过程。

在深度学习中，已经被提出的元学习模型有很多，大致上可以分类为learning good weight initializations，meta-models that generate the parameters of other models 以及learning transferable optimizers。其中MAML属于第一类。MAML学习一个好的初始化权重，从而在新任务上实现fast adaptation，即在小规模的训练样本上迅速收敛并完成fine-tune。

(2) model-agnostic

model-agnostic即**模型无关**。MAML与其说是一个深度学习模型，倒不如说是一个框架，提供一个meta-learner用于训练base-learner。这里的meta-learner即MAML的精髓所在，用于learning to learn；而base-learner则是在目标数据集上被训练，并实际用于预测任务的真正的数学模型。绝大多数深度学习模型都可以作为base-learner无缝嵌入MAML中，而MAML甚至可以用于强化学习中，这就是MAML中model-agnostic的含义。

(3) N-way K-shot

N-way K-shot是few-shot learning中常见的实验设置。few-shot learning指利用很少的被标记数据训练数学模型的过程，这也正是MAML擅长解决的问题之一。N-way指训练数据中有N个类别，K-shot指每个类别下有K个被标记数据。

(4) task

MAML的论文中多次出现名词**task**，模型的训练过程都是围绕task展开的，而作者并没有给它下一个明确的定义。要正确地理解task，我们需要了解的相关概念包括$\mathcal{D}_{meta-train}$，$\mathcal{D}_{meta-test}$，**support set**, **query set**, **meta-train classes**, **meta-test classes**等等。是不是有点眼花缭乱？不要着急，举个简单的例子，大家就可以很轻松地掌握这些概念。

我们假设这样一个场景：我们需要利用MAML训练一个数学模型模型 $M_{fine-tune}$ ，目的是对未知标签的图片做分类，类别包括 $P_1 \sim P_5$ （每类5个已标注样本用于训练。另外每类有15个已标注样本用于测试）。我们的训练数据除了 $P_1 \sim P_5$ 中已标注的样本外，还包括另外10个类别的图片 $C_1 \sim C_{10}$ （每类30个已标注样本），用于帮助训练元学习模型 M_{meta} 。我们的实验设置为**5-way 5-shot**。

关于具体的训练过程，会在下一节MAML算法详解中介绍。这里我们只需要有一个大概的了解：MAML首先利用 $C_1 \sim C_{10}$ 的数据集训练元模型 M_{meta}，再在 $P_1 \sim P_5$ 的数据集上精调（fine-tune）得到最终的模型 $M_{fine-tune}$ 。

此时， $C_1 \sim C_{10}$ 即**meta-train classes**， $C_1 \sim C_{10}$ 包含的共计300个样本，即 $\mathcal{D}_{meta-train}$ ，是用于训练 M_{meta} 的数据集。与之相对的， $P_1 \sim P_5$ 即**meta-test classes**， $P_1 \sim P_5$ 包含的共计100个样本，即 $\mathcal{D}_{meta-test}$ ，是用于训练和测试 $M_{fine-tune}$ 的数据集。

根据5-way 5-shot的实验设置，我们在训练 M_{meta} 阶段，从 $C_1 \sim C_{10}$ 中随机取5个类别，每个类别再随机取20个已标注样本，组成一个**task** $\mathcal{T}$ 。其中的5个已标注样本称为 $\mathcal{T}$ 的**support set**，另外15个样本称为 $\mathcal{T}$ 的**query set**。这个**task** $\mathcal{T}$ ， 就相当于普通深度学习模型训练过程中的一条训练数据。那我们肯定要组成一个batch，才能做随机梯度下降SGD对不对？所以我们反复在训练数据分布中抽取若干个这样的**task** $\mathcal{T}$ ，组成一个batch。在训练 $M_{fine-tune}$ 阶段，**task**、**support set**、**query set**的含义与训练 M_{meta} 阶段均相同。

赞同 1218
分享

作者在论文中给出的算法流程如下：

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**

知乎 @徐啊歪

MAML算法

该算法实质上是MAML预训练阶段的算法，目的是得到模型 M_{meta} 。不要被这些数学符号吓到喔，这个算法的思路其实很简单。接下来，我们来一行一行地分析这个算法。

首先来看两个**Require**。

第一个Require指的是在 $\mathcal{D}_{meta-train}$ 中**task**的分布。结合我们在上一小节举的例子，这里即反复随机抽取**task** $\mathcal{T}$ ，形成一个由若干个（e.g., 1000个） $\mathcal{T}$ 组成的**task**池，作为MAML的训练集。有的小伙伴可能要纳闷了，训练样本就这么多，要组合形成那么多的task，岂不是不同task之间会存在样本的重复？或者某些task的**query set**会成为其他task的**support set**？没错！就是这样！我们要记住，MAML的目的，在于**fast adaptation**，即通过对大量task的学习，获得足够强的泛化能力，从而面对新的、从未见过的task时，通过fine-tune就可以快速拟合。task之间，只要存在一定的差异即可。再强调一下，MAML的训练是基于**task**的，而这里的每个**task**就相当于普通深度学习模型训练过程中的一条训练数据。

第二个Require就很好理解啦。step size其实就是学习率，读过MAML论文的小伙伴一定会对**gradient by gradient**这个词有印象。MAML是基于二重梯度的，每次迭代包括两次参数更新的过程，所以有两个学习率可以调整。

接下来，就是激动人心的算法流程。

步骤1，随机初始化模型的参数，没什么好说的，任何模型训练前都有这一步。

步骤2，是一个循环，可以理解为一轮迭代过程或一个epoch，当然啦预训练的过程是可以有多个epoch的。

步骤3，相当于pytorch中的DataLoader，即随机对若干个（e.g., 4个）**task**进行采样，形成一个batch。

步骤4～步骤7，是第一次梯度更新的过程。注意这里我们可以理解为copy了一个原模型，计算出新的参数，用在第二轮梯度的计算过程中。我们说过，MAML是gradient by gradient的，有两次梯度更新的过程。步骤4～7中，利用batch中的每一个task，我们分别对模型的参数进行更新（4个task即更新4次）。注意这一个过程**在算法中是可以反复执行多次的**，伪代码没有体现这一层循环，但是作者再分析的部分明确提到" using multiple gradient updates is a straightforward extension"。

步骤5，即对利用batch中的某一个task中的support set，计算每个参数的梯度。在N-way K-shot的设置下，这里的**support set**应该有**NK**个。作者在算法中写with respect to K examples，默认对每一个class下的K个样本做计算。实际上参与计算的总计有NK个样本。这里的loss计算方法，在回归问题中，就是MSE；在分类问题中，就是**cross-entropy**。

步骤6，即第一次梯度的更新。

步骤4～步骤7结束后，MAML完成了**第一次**梯度更新。接下来我们要做的，是根据第一次梯度更新得到的参数，通过gradient by gradient，计算第二次梯度更新。第二次梯度更新时计算出的梯度，直接通过SGD作用于原模型上，也就是我们的模型真正用于更新其参数的梯度。换句话说，第一次梯度更新是为了第二次梯度更新，而第二次梯度更新才是为了更新模型参数。

关于以上过程，这里再补充一下解释：假设原模型是 θ_a ，我们复制了它，得到 θ_b 。在 θ_b 上，我们做了反向传播及更新参数，得到第一次梯度更新的结果 θ_b' 。接着，在 θ_b' 上，我们将计算第二次梯度更新。此时需要先在 θ_b' 上计算梯度（计算方法如接下来的步骤8所述），**但是梯度更新的并非是 θ_b' ，而是原模型 θ_a** 。这就是二重梯度在代码中的实现。

步骤8即对应第二次梯度更新的过程。这里的loss计算方法，大致与步骤5相同，但是不同点有两处。一处是我们不再是分别利用每个task的loss更新梯度，而是像常见的模型训练过程一样，计算一个batch的loss总和，对梯度进行随机梯度下降SGD。另一处是这里参与计算的样本，是task中的**query set**，在我们的例子中，即5-way*15=75个样本，目的是增强模型在task上的泛化能力，避免过拟合**support set**。步骤8结束后，模型结束在该batch中的训练，开始回到步骤3，继续采样下一个batch。

以上即时MAML预训练得到 M_{meta} 的全部过程。事实上，MAML正是因为其简单的思想与惊人的表现，在元学习领域迅速流行了起来。接下来，应该是面对新的**task**，在 M_{meta} 的基础上，精调得到 $M_{fine-tune}$ 的方法。原文中没有介绍**fine-tune**的过程，这里我简单介绍一下。

个task的support set训练模型，利用query set测试模型。实际操作中，我们会在 $\mathcal{D}_{meta-test}$ 上随机抽取许多个task（e.g., 500个），分别微调模型 M_{meta} ，并对最后的测试结果进行平均，从而避免极端情况。

- fine-tune没有步骤8，因为task的query set是用来**测试**模型的，标签对模型是未知的。因此**fine-tune过程没有第二次梯度更新，而是直接利用第一次梯度计算的结果更新参数。**

以上就是MAML的全部算法思路。我也是在摸索学习中，如有不足之处，敬请指正。

由于精力有限，私信无法一一回复，抱歉。另外知乎专栏的数学公式真是反人类啊，太难用了。。

参考资料：

- MAML论文：Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks]
- MAML Pytorch实现：dragen1860/MAML-Pytorch 或 katerakelly/pytorch-maml
- MAML Tensorflow实现：cbfinn/maml

编辑于 2021-07-27 15:32

meta-learning 元学习器 深度学习（Deep Learning）

166 条评论 切换为时间排序

写下你的评论...

一名 2021-05-31

记录下自己的理解，不对请指正。“这里我们可以理解为copy了一个原模型”意思是 copy出一个临时模型，然后求了loss，在临时模型上更新了梯度，但是没有作用于 原模型，只是在copy出来的模型上作用了，属于临时的。步骤4-7一直在临时模型上作用。且一个task对应一个临时模型，不然就会造成使用上一个task的更新过的参数了。

第二次梯度更新 在copy出来临时模型上计算 loss，但是不作用于 copy出来的模型，而是去作用于 原模型，这才是真正的更新！

从原模型的角度看，只进行了一次梯度更新，也就是在第二次即步骤8。但第二次更新是依赖于第一次(步骤4-7)的。这也是 “gradient by gradient”的本质。
（总结：先算一次梯度，但不作用于原模型，再算一次，才作用于原模型）

20

知乎用户 回复 **一名** 2021-10-14

感觉就像是：我和我朋友（对应copy的原模型）都不会区分猫狗，然后我朋友拿着猫狗照片各5张（有标签，即support set，对应2-way 5-shot）学会了（第1次gradient），接着ta再找了一堆的猫狗照片（即querry set）更深入学习了，最后把ta学到的知识教给我了（第2次gradient）

12

折夏夏 回复 **一名** 2021-11-23

我的理解和你类似。原模型可以看作一个pretrain模型，我们给他喂一个batch的Task，可以快速fine tune（进行第一次梯度更新）出一个新模型，如何评价原模型的好坏？我们注意到，在不同原模型基础上fine tune出来的新模型好坏是不一样的。所以将新模型在训练数据的query set上测试得到的损失，用于原模型的梯度更新。

4

邓冇 2019-03-12

你好，最近在看元学习，感觉看完了很困惑，主要在于meta-learning中对任务的定义都是N-way K-shot的形式，我能用这种技术解决传统分类问题吗？如果就考虑图像分类场景，假如我有一个10类的数据，每个类别10张图片，共10*10=100张。那meta-learning的过程就是，首先设置实验，比如10-way 5-shot，meta-train的过程拿imagenet这种数据集pre-train，然后meta-test就是在我自己的数据（100张图）上面finetune，最后拿到那套参数θ，然后推理的时候每次就拿这个θ算一下前向进行分类。请问是这么理解的吗？

15

徐不知 (作者) 回复 **邓冇** 2019-03-15

是的哦，就是这样理解的。完全正确！不过毕竟是小样本学习，效果肯定没有那么理想。

5

老冰 回复 **邓冇** 2021-10-26

是这样做的，建议你去看下maml论文的代码，看完你就明白了。他论文代码就是这么写的

1

知乎用户 2019-09-15

这个和迁移学习有区别吗？不都是找个好的初值？