CSL7530
Report on Meta-Learning with Implicit Gradients

Sanyam Jain
P20QC001
Indian Institute of Technology, Jodhpur
sanyam.1@iitj.ac.in

Abstract

This report explains the experiments performed to reproduce the results of the research paper "Meta-Learning with Implicit Gradients" , published by Rajeswaran et al at the Thirty-third Conference on Neural Information Processing Systems(NeurIPS) 2019. The original code for the project is available at the following https URL: https://github.com/aravindr93/imaml_dev

Introduction

As we already know machine learning is resource and data hungry. Being intelligent comes from experiences that is important part of becoming intelligent and adapt tasks. Meta-learning is subset of machine learning that tries to address the problem of machine learning being data hungry. The motivation of meta-learning comes from being able to learn from small data. From previous research, we have already known that supervised ML models are data-hungry. If you give large datasets to train large models, you learn a brilliant decision boundary (broad generalization). Meta-learning (or learning to learn) uses bi-level optimization criteria. In the multiple time scales of iterative optimization, the meta-learning can be divided into 2 parts, inner loop, and outer loop. Inner loop may optimize some cost function however, outer loop optimizes environment. For example, in an individual's lifetime of learning, whatever learns from the outside world is the outer loop and whatever learns inside that adapts to the outer environment is the inner loop. Traditional ML algorithms start to fall when situations tend to change fast and your model runs out of training data. Essentially we want our model to classify with large prior experience and small training knowledge because you don't have the luxury of having huge training data. In other words, can we explicitly

learn priors from previous experiences that lead to an efficient entirely data-driven approach to learning. There are several different ideas regarding how to perform meta-learning in the machine learning community. One of the prevalent approaches is using metric-learning. In metric-learning, the meta-learner learns an embedding where the projected features will be close to the projected features for new tasks. Nearest-neighbor approaches are then used on this embedding space to perform few-shot learning. Another approach to performing meta-learning is to use recurrent/recursive neural networks which are trained on individual samples from across tasks. These networks are then used to generate parameter updates or predictions for new tasks. Finally, meta-learning can also be treated as a two-level optimization problem which I discuss in further detail in the next sub-section.

Meta Learning with Bi-Level Optimization

Meta-learning can be considered as a two level optimization problem where the two levels optimize for different objectives as follows:

1. The inner objective corresponding to the learning problem of optimizing the cost function for a particular task (i.e. the traditional supervised/ reinforcement learning objective)
2. The outer objective corresponding to the meta-learning problem of optimizing the learning across tasks

In previous research, the authors address the meta-learning as bi-level optimization problem. In particular, they use the Model-Agnostic Meta-Learning (MAML)set-up and extend it to decouple the dependency in the outer optimization step from the gradients in the inner optimization loop. This is extremely helpful because this dependency leads to calculation of higher order gradients of the parameters in the inner objective function which have a high computational as well as storage complexity.

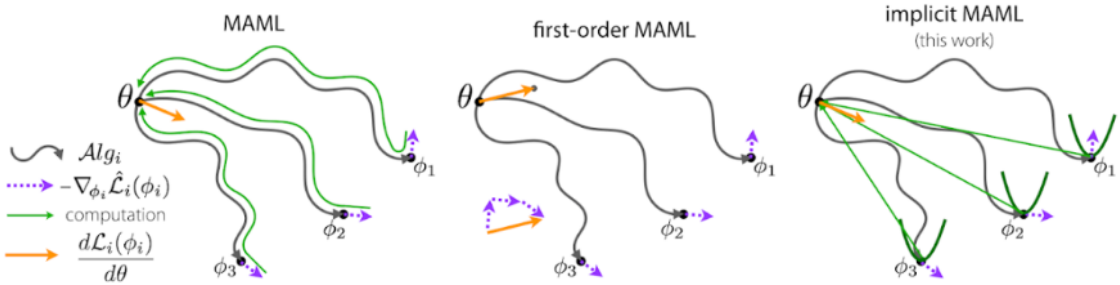


Figure 1 Variants of MAML for meta-learning as bi-level optimization visualized

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

Figure 2 This is a bi-level optimization problem asAlg(θ,Dtri)solves the inner optimization problem for each task

Implicit MAML

Model-Agnostic Meta-learning (MAML) was one of the first attempts to explicitly formalize the meta-learning problem as bi-level optimization (even though others had been using the approach before). Their aim was to develop an iterative algorithm to update the meta parameters as $\vartheta \leftarrow \vartheta - \eta d\vartheta F(\vartheta)$. MAML uses gradient descent (and its variants) based optimization for both the inner and the outer optimization problem to demonstrate the usefulness of algorithms which interpret meta-learning as bi-level optimization. However, MAML explicitly uses the gradient descent dynamics of the inner loop optimization in the outer loop which makes it compute and memory expensive. Also, instead of explicitly regularizing the task specific parameters, MAML uses early stopping (in terms of limited gradients updates in the inner loop) as a regularizer. First-order MAML approximates the inner derivative (i.e. task specific parameters w.r.t. the meta-parameters) with an identity matrix. This reduces the memory storage requirement of MAML but introduces an error in the inner loop. The authors of [9] introduce the method of using implicit gradients to solve the bi-level optimization problem discussed above. We next discuss it in further detail. Figure 1 provides a visual illustration of the these three MAML variants.

While the update equation in MAML corresponds to the gradient descent update rule, the approach can be easily extended to any other optimizer as we will shortly see. Using the chain rule, the update equation can be expanded as:

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

The first term is the gradient of the inner-optimization problem wrt the meta parameters. This can be tricky to compute because of both the nested gradient as well as the fact it depends on the optimum solution to the inner optimization problem. The $\nabla\varphi Li(Alg\star i(\vartheta))$ term is simply $\nabla\varphi Li(\varphi)|\varphi=Alg\star i(\vartheta)$ which can be obtained using auto diff like in standard machine learning practice. One of the major drawbacks of MAML is that the computation of the meta gradient depends on the actual inner optimization process. Thus, if gradient descent was used to calculate the gradients in the inner loop we need to store and propagate gradients from the inner loop to the outer loop. Furthermore, if second-order algorithms like Newton's method are used then then third derivatives are required which are computationally expensive to get. However, the meta-gradient requires only the optimum solution of the inner loop not the entire optimization process. The Implicit MAML paper goes into further details with regards to the memory footprint and mathematical sanity behind the algorithm. These proofs are beyond the scope of this report but the readers are referred to the excellent appendix section of the original paper for a better understanding. The practical algorithm is summarized above in Algorithm 1 (from the paper). The table below

summarizes the memory footprint and accuracy of the three MAML variants discussed before in terms of Big-O notation. While MAML is the most accurate of the methods discussed it has large memory requirement which scales linearly with condition number K of the inner optimization problem and logarithmically with search space **D** of the inner loop. If the Nesterov's momentum update is used instead of gradient descent then MAML scales linearly with **√K** both for compute and memory. First-order MAML does better in terms of memory footprint but still has a similar order of compute (in terms of **∇^L** computations). Implicit-MAML has the best memory requirements as well as compute among all the variants. By allowing for an error of **δ** in the computation of the solution to the inner optimization problem, the only memory requirement is to store a single derivative Mem (**∇Li**). The compute also scales linearly in terms of **√K** which is joint best among all variants.

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

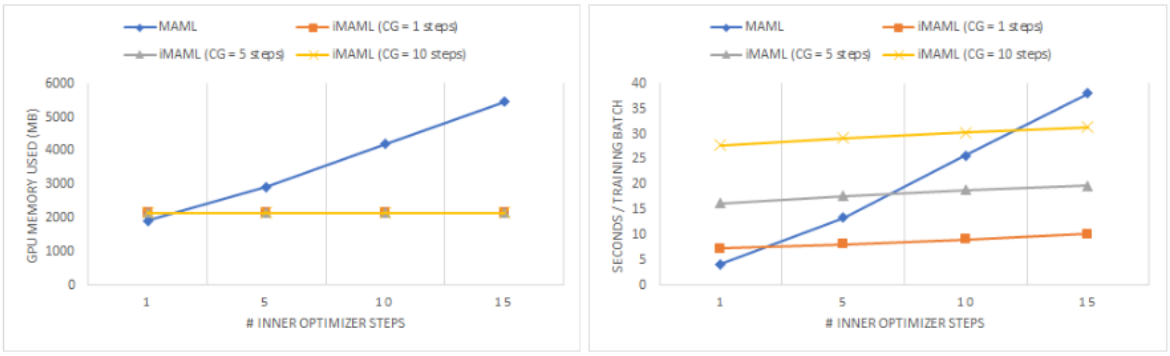
(a) Different characters in the Omniglot dataset (b) Different classes in the Omniglot dataset

Figure 3 The Omniglot dataset

Experimentation and Results Discussion

The Omniglot dataset is a standard dataset of handwritten characters for few-shot learning(analogous to how the MNIST dataset is for machine learning). The dataset is made up of 20instances (all drawn by different individuals) of 1623 characters from 50 different alphabets. The experimental procedure followed is to evaluate fast learning of N-way classification using few shots(number of instances used to train the model). The problem set-up as described in [10] is: "select N unseen classes, provide the model with K different instances of each of the N unseen classes, and evaluate the model's ability to classify new instances within the N classes." 1200 characters are randomly

picked as training and the remaining are used for testing. Since the Omniglot dataset is in gray-scale and quite simple, in order to perform data pre-processing we take random crops of size 80x80 pixels with a padding of 8 pixels while training. While testing we take a center-crop of the same size. For model architecture we use a convolution neural network made up of 4 convolution layers as our parameterized model. Each convolution layer is made up of 64 channels. Each convolution layer uses a convolution window of 3x3 with a padding of 1 with ReLU activation.



(a) Memory Requirements (b) Compute Requirements

Figure 4 Performance Comparisons of MAML and iMAML

Conclusions

In this report, I summarized the paper "Meta-Learning with Implicit Gradients" [9], published by Rajeswaran et al at the Thirty-third Conference on Neural Information Processing Systems (NeurIPS)2019. The paper introduces a method to perform meta-learning as bi-level optimization using implicit gradients. This is done by regularizing the task-specific parameters to be close to the meta parameters and then using an implicit gradient approximation of the solution for the optimization problem. The authors demonstrate the effectiveness of said method on regression and classification tasks. I reproduce the results of the paper for few-shot image classification on the Omniglot dataset to confirm the authors' findings.

References

1. Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In Advances in neural information processing systems, pages 3630–3638, 2016.
2. Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In Advances in Neural Information Processing Systems, pages 4077–4087, 2017.
3. Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In International Conference on Artificial Neural Networks, pages 87–94. Springer, 2001.
4. Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.
5. Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shilling ford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In Advances in neural information processing systems, pages 3981–3989, 2016.
6. Ke Li and Jitendra Malik. Learning to optimize. arXiv preprint arXiv:1606.01885, 2016.
7. Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In International conference on machine learning, pages 1842–1850, 2016.
8. Tsendsuren Munkhdalai and Hong Yu. Meta networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 2554–2563. JMLR. org, 2017.
9. Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In Advances in Neural Information Processing Systems, pages 113–124, 2019.
10. Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages1126–1135. JMLR. org, 2017.
11. Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.
12. James Martens. Deep learning via hessian-free optimization. In Proceedings of the 27th International Conference on Machine Learning, 2010.
13. John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policyoptimization. InInternational conference on machine learning, pages 1889–1897, 2015.

14. Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. The omniglot challenge: a 3-yearprogress report. Current Opinion in Behavioral Sciences, 29:97–104, 2019.
15. Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprintarXiv:1412.6980, 2014.
16. Ferenc Huszár. Notes on imaml: Meta-learning with implicit gradients in FERENCe, 2019.
17. Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-basedmeta-learning as hierarchical bayes. arXiv preprint arXiv:1801.08930, 2018.
18. Yutian Chen, Abram L. Friesen, Feryal Behbahani, David Budden, Matthew W. Hoffman, Arnaud Doucet, and Nando de Freitas. Modular meta-learning with shrinkage, 2019.
19. Sung-Yub Kim. GBML.https://github.com/sungyubkim/GBML, 2019.
20. Tristan Deleu, Tobias Würfl, Mandana Samiei, Joseph Paul Cohen, and Yoshua Bengio. Torchmeta: AMeta-Learning library for PyTorch, 2019. Available at: https://github.com/tristandeleu/pytorch-meta.
21. Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier,Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. arXivpreprint arXiv:1910.01727, 2019.