Teaching Your MAML

Gleb Shevchuk
Stanford Computer Science

***Abstract*—Meta learning is incredibly exciting because it promises to learn generalized policies across tasks. However, meta learning is held back by the practical assumption that meta-training tasks are sampled from a known distribution. In practice, it is very hard to define this distribution. If made too small, tasks are too similar to meaningfully generalize. If made too large, meta-learning becomes incredibly difficult. We argue that both problems can be alleviated by introducing a teacher model whose objective it is to control this meta-task distribution. This teacher model is incentivized to start the student meta-model on simple tasks then adaptively increase task difficulty in response to the student's progress. In turn, we can use knowledge from easier tasks to learn harder tasks, allowing us to learn generalizable policies faster. While this approach has been generally studied in curriculum learning, our key contribution is to automate the teacher model and update task-space parameters in response to student performance.**

I. Introduction

Humans are incredibly good at generalizing to unseen tasks. But, humans are only able to do so because they lean on a vast history of experience. Within a single lifespan, we begin by learning simple tasks: crawling, walking, talking. As we age, we learn progressively more and more difficult tasks, borrowing from the simpler to inform the more complex.

In order for machines to exhibit this same behavior, they have to learn how to generalize and borrow from previous experiences. Once they can do so reliably, we move a step closer to the holy grail of Artificial General Intelligence. However, artificial intelligence systems are incredibly brittle. Because we have an incomplete understanding of how to best learn from past experiences, it is unclear how we can create robust, generalizable AI agents.

We propose to tackle this problem by combining curriculum learning and meta learning to teach meta-AI agents to start from easy tasks, progressively learn harder tasks, and use information about easy tasks to inform the harder ones. In order to do so, we introduce a teacher that updates the difficulty of the current task in response to student progress.

Using this formulation on three classification tasks, we show that our approach leads to better generalization. Before doing so, we would like to discuss related works and provide preliminary information for attacking this problem.

II. Related Works

Meta-learning, life-long, curriculum, few-shot, one-shot and incremental learning are all concerned with one core challenge: when a model is given a problem it has never seen before, how can it use its prior knowledge to solve that problem?

Algorithm 1 Adaptive Meta-Teaching

Require: $p(\mathcal{T})$: Task-parameter space
Require: $\mathcal{U}$: Find approximate gradient procedure
Require: $\mathcal{L}$: Loss function
Require: θ: Student parameters
Require: ω: Parent parameters
while not done **do**
  Sample meta-batch of tasks from perturbed task-space distribution $\mathcal{T}_i \sim p(\mathcal{T}) + \mathcal{N}$
  for all $\mathcal{T}_i$ **do**
    Evaluate $\nabla_{\theta,T_i}\mathcal{L}_{\mathcal{T}_i}$ with respect to K examples
    Compute adapted parameters with gradient descent: $\theta_{i+1} \leftarrow \theta_i - \nabla_{\theta,T_i}\mathcal{L}_{\mathcal{T}_i}(\theta_i)$.
  end for
  Teacher step - update task space parameters
  Approximate difficulty gradient $\nabla_\psi \approx \mathcal{U}(\mathcal{L}_i, \omega_i)$
  Compute adapted task-space parameters with gradient descent: $\omega_{i+1} \leftarrow \omega_i - \nabla_\psi$.
end while

Meta-learning, or learning to learn, first began with the intent of learning the best way to pre-update models by learning learning rules or update functions [20][3]. Many of these earlier approaches used random search to perform this pre-update, [1], and showed that pre-training a model on a previous set of similar tasks improved performance on current tasks. Recently, approaches like MAML [6], FOMAML [2], and REPTILE [18] started using first or second order gradient information to perform this meta-learning step. Generally, we can broadly divide meta-learning approaches into three categories: those that use random search or evolutionary methods (Baldwinian/Lamarckian Evolution [5]), those that use gradient information for gradient descent or optimization (MAML [6]), and those that explicitly use past information (RNNs [26])

Curriculum learning addresses a similar problem, but centers on reusing previous experience either sequentially or in a manner that maximizes student progress[4]. The term self-paced learning has been used recently to describe this, [10][9][27][12], but the problem boils down to the same concept: how can a student most effectively learn and guide its learning? This is done with the hope that we can build systems that are continuously learning and using past experience, a concept which itself has been exxtensively explored [17][22][23][15][21][25][11]. Multiple approaches have also been proposed to use curriculum learning for transferring information between tasks [19][7].

The closest approach to our own is detailed in [16] and is referred to as teacher-student curriculum learning. In this set up, the teacher chooses to train the student on N discrete tasks. Depending on how well the student learns a task (using the learning curve), the teacher updates the probability with which it samples that task at the next training iteration. This method borrows from non-stationary multi-arm bandit literature and incentivizes the teacher to choose tasks that the student has previously achieved high reward progress on. Although this method can be extended to continuous task-space problems, it is unable to take advantage of progressive task knowledge and does not address either of the core problems in meta-learning.

Another similar approach shown in [8] also uses diversity to perform unsupervised meta-learning. However, this work again does not attempt to perform curriculum learning nor does it transfer from simpler policies to more difficult policies.

III. Preliminaries

We follow from the setup given by Model Agnostic Meta Learning. Here, we seek to learn an initial set of parameters $\theta^\star$ for a neural network across a distribution of tasks $p(\mathcal{T})$ such that a task $\mathcal{T}_i$ sampled from the distribution can be solved in a small number of gradient steps.

The meta-learner's objective is defined as:

$$\min_{\theta} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(\theta_i') \tag{1}$$

where we take an expectation over the task space, $L_{\mathcal{T}_i}$ refers to the loss on the ith example of the current task, and θ_i' are the parameters adapted to fit K examples of the current task.

Following the MAML setup, we obtain θ_i' by performing gradient descent:

$$\theta_i' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta) \tag{2}$$

We can use $\omega_{p(\mathcal{T})}$ to refer to the parameters that define the distribution of tasks $p(\mathcal{T})$. These can either be continuous or discrete. Furthermore, we assume to have access to $\psi(\omega)$, which defines the difficulty of the distribution of tasks. ψ can be changed by modulating our ω, although the relationship between ω and ψ is assumed to be nonlinear.

IV. Learning to Meta-Teach

A. The Static Teacher

We begin by exploring a meta-learning version of a static teacher model that has been previously used in curriculum learning.

The static teacher operates as follows. Before training, we define a static set of progressively more difficult task distributions. These task distributions can either be manually defined or drawn from another distribution. Then, during meta-training time, whenever the student model's meta validation loss goes below a specific threshold, we switch to the next, slightly harder set of tasks. We continue doing so until we reach the the final, most difficult set of tasks.

Though this approach is simple to implement, it heavily relies on knowledge of task-space parameters. In turn, it can fail catastrophically. For example, if we switch to a harder task distribution and it becomes too difficult for our model to solve the task, it will never reach our threshold and will never progress to the next set of tasks. Furthermore, in the chance that our model records a meta validation loss below the threshold on a set of tasks that do not encompass the full difficulty of the current task distribution, it would move on to more difficult tasks when, in reality, it was not able to fully solve the easier tasks.

We can alleviate these problems by creating more interpretable task space parameterizations and using higher-order information about student progress, but this puts more burden on task design and, ultimately, only adds to model brittleness.

B. The Adaptive Teacher

We now introduce our key contribution, the adaptive teacher. The adaptive teacher's goal is to decrease student loss while increasing task complexity. Ideally, we would like our teacher to observe several behaviors.

1) If the student is consistently over-performing on the current set of tasks, the teacher should increase the task difficulty.
2) If the student is consistently under-performing on the current set of tasks, the teacher should decrease task difficulty.
3) The teacher should update in the direction of $\nabla\psi$ at ω. In other words, the teacher should capture how much a change in each task distribution parameter affects the difficulty of the task distribution.
4) The teacher should eventually push the student to explore more difficult tasks.

In order to meet these requirements, we introduce a teacher model that updates task parameters by approximating how much its previous parameter changes affected task difficulty. To do so, we use information about student progress in order to estimate the gradient of the task space difficulty with respect to task space parameters.

In the next section, we will describe how we estimate this gradient and how it can be used to change our task space distribution.

1) Estimating $\nabla\psi$: In practice, it is difficult to define our difficulty ψ, since it requires domain-specific knowledge about how each task distribution parameter affects the difficulty of the task distribution. In turn, we assume that the student's learning progress at the current time-step is equivalent to the difficulty of the task distribution.

Once we have the student's meta validation loss at each timestep, we would like to use it to determine how much we should update the teacher's parameters in order to accomplish requirement 3. Effectively, this requires us to estimate $\nabla\psi$, or the gradient of our difficulty, with respect to the task distribution parameters ω. In order to approximate $\nabla\psi$, we perturb the current parameters ω with uniform noise. Then, by subtracting the current meta-learning loss from the previous meta-learning loss and dividing by change in parameter, we can approximate the derivation of $\mathcal{L}$ with respect to each task

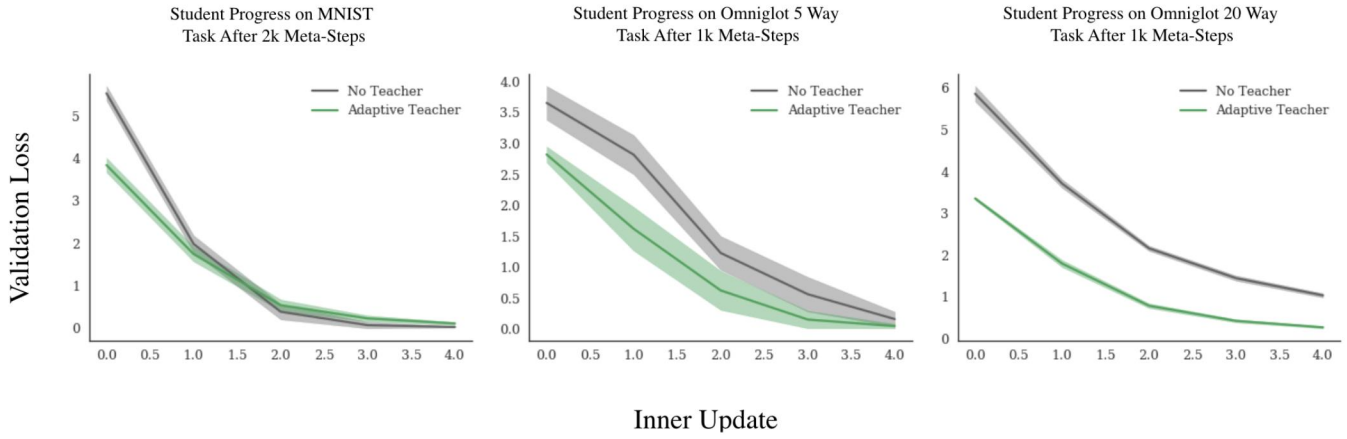

Fig. 1. Performance of the adaptive teacher compared to standard MAML on 10 tasks sampled from the hardest task distribution across three experiments. In all three preliminary experiments, we find that using an adaptive teacher improves MAML performance on unseen tasks.

distribution parameter. In order to refine this gradient, we can use information over more than two timesteps. Furthermore, in order to better capture our requirements, we only take gradient steps that are larger than a certain threshold, which is specified as a problem-specific hyperparameter. Once we have this, we can perform gradient descent over the task distribution parameters. In practice, because we assume this approximation is noisy, we only pay attention to the sign of each partial derivative and update each parameter in that direction by a pre-specified amount. One advantage of this approach is that the meta-teacher can use the same information as the meta-optimizer in order to update the task distribution.

TABLE I
HYPERPARAMETERS

Hyperparameter	Value
Inner Learning Rate	0.1
Meta Learning Rate	0.001
Inner Updates	5
Meta Updates	2000
Teacher Update Rate	4
Teacher Gradient Step	2
Inner batch size	1
Meta Batch Size	8

V. Experiments, Datasets, and Features

In order to ascertain the efficacy of the adaptive teacher approach, we compare it to standard meta-learning on three tasks, an MNIST classification task and two Omniglot classification tasks.

Ideally, we would like to show that after the same number of meta-steps, if we roll out each model on a set of unseen tasks, the teacher-aided model will be closer to optimal, meaning that it will start with a lower loss before performing inner gradient descent and end with a lower loss after performing the same number of gradient steps.

In all three tasks, we use an open-source PyTorch implementation of MAML and Cross Entropy Loss. We also use the same neural network structure specified in [6], consisting of 3 64 by 64 convolution layers, each followed by a batch normalization layer, a 2 by 2 max pooling layer and a ReLU activation. All other hyperparameters used are listed in Table I.

VI. MNIST Classification Task

MNIST is a classification dataset consisting of 28 by 28 pixel grayscale input images of hand-written numbers and labels from 0-9 [14]. In order to adapt it to a meta-learning setting, we randomly relabel the images so that each task might be considered unique. For our teacher context, the task space parameters ω simply consist the number of classes from 1 to 10 used for the current task.

VII. Omniglot Classification Tasks

Omniglot is a classification dataset consisting of 20 unique handwritten samples of 1623 characters from 50 languages [13] that is often used for few-shot learning experiments. We use the same protocol as in [6] and [24] of N-Way K-shot training, where N unseen classes are selected for each task and K instances of each class are provided for training. We then evaluate the model on new instances from within these N classes. We consider two versions of this task, a 5 way 1 shot version and a 20 way 1 shot version. In the 5 way experiment, the task space parameters ω consist of the number of classes from 1 to 5 used for the current task, while in the 20 way experiment, ω consists of the number of classes from 1 to 20.

VIII. Results

Results for all three tasks are summarized in Figure 1. In the MNIST experiment, we find that after 2000 meta-steps, the teacher-assisted model begins with a lower loss

across 10 unseen tasks. But, because this is a relatively simple experiment, both the teacher-trained version and normal MAML converge to a similar validation loss after 4 gradient steps. Similarly, in the two Omniglot experiments, after 1000 meta-steps the teacher-assisted model starts and ends with a lower loss than the standard model, indicating that teacher assisted training might have led models to better generalize to unseen tasks in the hardest task distribution. However, because we had little time to fine-tune the hyperparameters for the adaptive teacher, it is unclear what degree of benefit teacher training provides.

IX. Discussion and Future Work

Our initial results on these three classification tasks provide encouraging validation of our adaptive approach to teaching meta-learning models. Because our chosen method, MAML, is already notoriously difficult to train [2],it might be even harder to add an additional layer of training on top of it. Furthermore, because we only used discrete task space parameters, we were unable to explore what effect this method could have on continuous task space problems, where this method might have been more useful. Nevertheless, we believe that our method, an unsupervised manner of updating task difficulty in response to student progress, can be beneficial in long-term settings, where we might want our models to explore more diverse and disconnected task spaces. This type of approach might be useful for unifying meta-learning, curriculum learning, and lifelong learning, and make it more accessible to use past knowledge to guide future tasks.

We are excited to see improvements to the difficulty gradient approximation and to manners for tuning both student and teacher hyperparameters. We also hope to extend this approach to meta learning in continuous task parameter space and apply it to regression and reinforcement learning problems. Ultimately, we believe that this kind of approach will be instrumental in creating lifelong, never-ending agents, and are motivated to see it working in lifelong settings.

To reproduce these results, code is available at https://github.com/glebshevchukk/morph.

References

[1] Ajith Abraham. "Meta learning evolutionary artificial neural networks". In: *Neurocomputing* 56 (2004), pp. 1–38.

[2] Antreas Antoniou, Harrison Edwards, and Amos Storkey. "How to train your MAML". In: *arXiv preprint arXiv:1810.09502* (2018).

[3] Samy Bengio et al. "On the optimization of a synaptic learning rule". In: *Preprints Conf. Optimality in Artificial and Biological Neural Networks*. Univ. of Texas. 1992, pp. 6–8.

[4] Yoshua Bengio et al. "Curriculum learning". In: *Proceedings of the 26th annual international conference on machine learning*. ACM. 2009, pp. 41–48.

[5] Chrisantha Fernando et al. "Meta-learning by the baldwin effect". In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM. 2018, pp. 1313–1320.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *arXiv preprint arXiv:1703.03400* (2017).

[7] Chen Gong et al. "Multi-modal curriculum learning for semi-supervised image classification". In: *IEEE Transactions on Image Processing* 25.7 (2016), pp. 3249–3260.

[8] Abhishek Gupta et al. "Unsupervised Meta-Learning for Reinforcement Learning". In: *arXiv preprint arXiv:1806.04640* (2018).

[9] Lu Jiang et al. "Self-paced curriculum learning". In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[10] Lu Jiang et al. "Self-paced learning with diversity". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2078–2086.

[11] Faisal Khan, Bilge Mutlu, and Jerry Zhu. "How do humans teach: On curriculum learning and teaching dimension". In: *Advances in Neural Information Processing Systems*. 2011, pp. 1449–1457.

[12] M Pawan Kumar, Benjamin Packer, and Daphne Koller. "Self-paced learning for latent variable models". In: *Advances in Neural Information Processing Systems*. 2010, pp. 1189–1197.

[13] Brenden Lake et al. "One shot learning of simple visual concepts". In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 33. 33. 2011.

[14] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* 2 (2010), p. 18.

[15] David Lopez-Paz et al. "Gradient episodic memory for continual learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6467–6476.

[16] Tambet Matiisen et al. "Teacher-student curriculum learning". In: *arXiv preprint arXiv:1707.00183* (2017).

[17] Tom Mitchell et al. "Never-ending learning". In: *Communications of the ACM* 61.5 (2018), pp. 103–115.

[18] Alex Nichol and John Schulman. "Reptile: a scalable metalearning algorithm". In: *arXiv preprint arXiv:1803.02999* 2 (2018).

[19] Anastasia Pentina, Viktoriia Sharmanska, and Christoph H Lampert. "Curriculum learning of multiple tasks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5492–5500.

[20] Jürgen Schmidhuber. "Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook". PhD thesis. Technische Universität München, 1987.

[21] Sainbayar Sukhbaatar et al. “Intrinsic motivation and automatic curricula via asymmetric self-play”. In: *arXiv preprint arXiv:1703.05407* (2017).

[22] Sebastian Thrun. “Lifelong learning algorithms”. In: *Learning to learn*. Springer, 1998, pp. 181–209.

[23] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[24] Oriol Vinyals et al. “Matching networks for one shot learning”. In: *Advances in neural information processing systems*. 2016, pp. 3630–3638.

[25] Tianjun Xiao et al. “Error-driven incremental learning in deep convolutional neural network for large-scale image classification”. In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM. 2014, pp. 177–186.

[26] Wojciech Zaremba and Ilya Sutskever. “Learning to execute”. In: *arXiv preprint arXiv:1410.4615* (2014).

[27] Qian Zhao et al. “Self-paced learning for matrix factorization”. In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.