Meta-Teaching: Curriculum Generation for Lifelong Learning

Anonymous Authors[1]

Abstract

Meta-learning will be crucial to creating lifelong, generalizable AI. In practice, however, it is hard to define the meta-training task distribution that is used to train meta-learners. If made too small, tasks are too similar for a model to meaningfully generalize. If made too large, generalization becomes incredibly difficult. We argue that both problems can be alleviated by introducing a teacher model that controls the sequence of tasks that a meta-learner is trained on. This teacher model is incentivized to start the student meta-learner on simple tasks then adaptively increase task difficulty in response to student progress. While this approach has been previously studied in curriculum generation, our main contribution is in extending it to meta-learning.

1. 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 into an approach called meta-teaching. We aim 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.

Before describing meta-teaching, we will discuss related works and provide preliminary information.

Algorithm 1 Adaptive Meta-Teaching

Require: $p(\mathcal{T})$: Task-space distribution
Require: $\psi(p(\mathcal{T}))$: Task-difficulty
Require: $\mathcal{L}_S$: Meta-Student Loss function
Require: $\mathcal{L}_T$: Meta-Teacher Loss function
Require: θ: Meta-learner parameters
Require: ω: Meta-teacher parameters

```
while not done do
    Sample meta-batch of tasks from perturbed task-space distribution T_i ~ p(T) + N
    for all T_i do
        Evaluate L_S T_i with respect to K samples from task
        Compute adapted student parameters using either a gradient based or gradient free update
    end for
    Approximate difficulty gradient of current task-space ∇_ψ L_T(L_S)
    Update adapted task-space parameters with gradient descent over i student losses: ω' ← ω_i − ∇_ψ L_T(L_S).
end while
```

2. 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 task it has never seen before, how can it use prior knowledge to solve that problem?

Meta-learning, or learning to learn, first began by learning the best way to pre-update models by learning update rules (Schmidhuber, 1987)(Bengio et al., 1992). Some of these earlier approaches used random search to perform this pre-update, (Abraham, 2004), and showed that pre-training a model on a previous set of similar tasks improved performance on current tasks. Recently, approaches like MAML (Finn et al., 2017), FOMAML (Antoniou et al., 2018), and

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

REPTILE (Nichol & Schulman, 2018) started using first or second order gradient information to perform this meta-learning update. Generally, we can divide meta-learning approaches into three categories: those that use random search or evolutionary methods (Baldwinian/Lamarckian Evolution (Fernando et al., 2018)), those that use gradient information for gradient descent or optimization (MAML (Finn et al., 2017)), and those that explicitly use past information (RNNs (Zaremba & Sutskever, 2014))

Curriculum learning addresses a similar problem, but centers on reusing previous experience either sequentially or in a manner that maximizes learner progress(Bengio et al., 2009). The term self-paced learning has been used recently to describe this, (Jiang et al., 2014)(Jiang et al., 2015)(Zhao et al., 2015)(Kumar et al., 2010), but the approach boils down to the same concept: how can a student most effectively 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 extensively explored (Mitchell et al., 2018)(Thrun, 1998)(Thrun & Pratt, 2012)(Lopez-Paz et al., 2017)(Sukhbaatar et al., 2017)(Xiao et al., 2014)(Khan et al., 2011). Multiple approaches have also been proposed to use curriculum learning for transferring information between tasks (Pentina et al., 2015)(Gong et al., 2016).

The closest approach to our own is detailed in (Matiisen et al., 2017) 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, the teacher updates the probability with which it samples that task at the next training iteration. This method borrows from 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 directly address either of the core problems in meta-learning.

Another similar approach shown in (Gupta et al., 2018) 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.

3. Preliminaries

We follow the setup proposed in (Finn et al., 2017). We seek to learn an initial set of parameters $\theta^\star$ for a model across a distribution of tasks $p(\mathcal{T})$ such that a task $\mathcal{T}_i$ sampled from the distribution can be solved in the smallest number of algorithm updates.

The meta-learner's objective is defined as the following:

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

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. We can then use any meta-learning algorithm, whether it be gradient-based or gradient-free, to update towards this objective.

We can use $\omega_{p(\mathcal{T})}$ to refer to the parameters that define the distribution of tasks $p(\mathcal{T})$ and these can either be continuous or discrete. Furthermore, Our next insight is that all tasks have a level of difficulty and can therefore be directly compared by their difficulties. We assume to have access to $\psi(\omega)$, which defines the difficulty of the distribution of tasks currently being used. The current difficulty ψ can be changed by modulating our distribution of tasks ω, and we make no assumption on the relationship between ω and ψ.

4. Meta-Teaching

Borrowing from our discussion about task space difficulty above, we would like our meta-learner to effectively navigate task difficulty and apply insight from simpler tasks to more difficult tasks. To that end, we introduce a meta-teacher model parameterized as a neural network. Given the task space parameters as input and the student's learning curve as loss, the teacher model must modulate task space difficulty.

This approach is similar in spirit to (Ha et al., 2016) and (**?**), but uses this external model not to update the student's weights, but rather to update the task space parameters that define the tasks that the student is trained on. This model acts as a teacher that uses insight about student progress to update task difficulty in an unsupervised manner.

4.1. Static Meta-Teaching

To highlight the novelty of this approach, we will first describe the type of curriculum learning that is most commonly used, and has previously been applied to the meta-learning setting (Bengio et al., 2009).

This "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 relies heavily 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.

4.2. Adaptive Meta-Teaching

Having discussed several of the issues with static meta-teaching, we 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 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 implicitly use information about student progress in order to estimate the gradient of the task space difficulty with respect to task space parameters. This is then used to update the teacher model.

4.2.1. Teacher Loss

Next, we describe the modified loss function that is used to train the teacher model and why this loss sufficiently encodes our four requirements.

The teacher's loss can be any kind of absolute value function on the average slope of the student's training loss. This means that we penalize the teacher whenever the student's performance is either too high or too low. In addition, we add an additional term on the total change in task space parameters to encourage the teacher to explore more difficult tasks. This combination of terms, in turn, allows us to progressively and continuously update our task difficulty in response to student performance.

4.2.2. Algorithm Overview

We now present a summary of the adaptive meta-teaching algorithm.

We start by initializing with the simplest set of task space parameters. During meta-training, the meta-learner samples from the task distribution that is parameterized by these parameters. Next, during each meta-training step, we store the meta-learner's loss on a set of meta-validation tasks drawn from the current task distribution. After N meta-training update steps, we calculate the modified loss described above, and use this to update the teacher model via gradient descent. Then, the teacher model produces a new set of task parameters. We continue to do so until termination.

The full approach is shown in Algorithm 1.

5. Challenges

There are several challenges to implementing a meta-teaching approach in a meaningful manner.

First, this approach relies on the notion that our task space can be described using a continuous difficulty function. This might be simple to implement when difficulty is analogous to domain size. For example, we might assume that a larger task distribution is equivalent to a harder one. However, in more meaningful task spaces, it is unclear how this difficulty function would be applied. For example, the space of tasks that humans can perform is characterized by difficulty across an enormous amount of difficulty parameters. Would we have to define the difficulty of each task manually or fall back manual labelling? Ideally, could we learn a task difficulty space that encodes difficulty relationships between all pairwise tasks?

Second, because we assume that our task space distribution can be updated using gradient descent, we have to carefully bound the meta-teacher's outputs so that they correspond to valid task distribution parameters.

Third, adding a meta-teaching adds an additional layer to meta-optimization, which is already notoriously difficult to train. Making meta-teaching practical requires simultaneously addressing the underlying difficulty of current meta-learning algorithms.

6. Future Work

We are incredibly excited to apply this type of approach to a wide variety of meta learning problems. In the future, we hope to apply it to meta-learning distributions that cannot be solved by learning on the entire distribution at once.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 believe that this kind of approach will be instrumental in creating lifelong agents, and are motivated to see it working in lifelong settings.

References

Abraham, A. Meta learning evolutionary artificial neural networks. *Neurocomputing*, 56:1–38, 2004.

Antoniou, A., Edwards, H., and Storkey, A. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

Bengio, S., Bengio, Y., Cloutier, J., and Gecsei, J. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, pp. 6–8. Univ. of Texas, 1992.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Fernando, C., Sygnowski, J., Osindero, S., Wang, J., Schaul, T., Teplyashin, D., Sprechmann, P., Pritzel, A., and Rusu, A. Meta-learning by the baldwin effect. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1313–1320. ACM, 2018.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Gong, C., Tao, D., Maybank, S. J., Liu, W., Kang, G., and Yang, J. Multi-modal curriculum learning for semi-supervised image classification. *IEEE Transactions on Image Processing*, 25(7):3249–3260, 2016.

Gupta, A., Eysenbach, B., Finn, C., and Levine, S. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018.

Ha, D., Dai, A., and Le, Q. V. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Jiang, L., Meng, D., Yu, S.-I., Lan, Z., Shan, S., and Hauptmann, A. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pp. 2078–2086, 2014.

Jiang, L., Meng, D., Zhao, Q., Shan, S., and Hauptmann, A. G. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Khan, F., Mutlu, B., and Zhu, J. How do humans teach: On curriculum learning and teaching dimension. In *Advances in Neural Information Processing Systems*, pp. 1449–1457, 2011.

Kumar, M. P., Packer, B., and Koller, D. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pp. 1189–1197, 2010.

Lopez-Paz, D. et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

Matiisen, T., Oliver, A., Cohen, T., and Schulman, J. Teacher-student curriculum learning. *arXiv preprint arXiv:1707.00183*, 2017.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Yang, B., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., et al. Never-ending learning. *Communications of the ACM*, 61(5):103–115, 2018.

Nichol, A. and Schulman, J. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2, 2018.

Pentina, A., Sharmanska, V., and Lampert, C. H. Curriculum learning of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5492–5500, 2015.

Schmidhuber, J. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., and Fergus, R. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

Thrun, S. Lifelong learning algorithms. In *Learning to learn*, pp. 181–209. Springer, 1998.

Thrun, S. and Pratt, L. *Learning to learn*. Springer Science & Business Media, 2012.

Xiao, T., Zhang, J., Yang, K., Peng, Y., and Zhang, Z. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 177–186. ACM, 2014.

Zaremba, W. and Sutskever, I. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

Zhao, Q., Meng, D., Jiang, L., Xie, Q., Xu, Z., and Hauptmann, A. G. Self-paced learning for matrix factorization. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.