ML3: meta Learning via Learned Loss

Apostolos Psaros

1 Introduction

This is a review of Bechtle et al. (2020). In general, following a gradient-based learning approach, the parameters of an optimizee $f_\theta(x)$ are updated as

$$\theta_{new} = h(\theta, \nabla_\theta \mathcal{L}(y, f_\theta(x))) \tag{1}$$

where $\mathcal{L}$ is a loss function that takes as input the current prediction $f_\theta(x)$ and the target y, and h is a function that takes as input the current parameters θ and the gradient of the loss function. For example, for simple gradient descent we update θ as

$$\theta_{new} = \theta - \alpha \nabla_\theta \mathcal{L}(y, f_\theta(x)) \tag{2}$$

where α is the learning rate. Also, for regression the loss function is typically the squared error $(y - f_\theta(x))^2$.

Two types of meta-learning (ML) approaches for improving learning speeds and generalization to new tasks are 1) learning optimization policies, i.e., parametrizing and optimizing the update function h and 2) learning loss functions, i.e., parametrizing and optimizing the loss function $\mathcal{L}$. The work of Bechtle et al. (2020) addresses the problem of learning loss functions. Similar works can be found in Houthooft et al. (2018) and Sung et al. (2017).

2 ML3 algorithm

The aim of this paper is to learn a meta-loss function $\mathcal{L}_\phi(y, f_\theta(x))$ to be used in Eq. (2) for training models f_θ with parameters θ. In this regard, $\mathcal{L}_\phi$ is a meta-loss network parametrized by ϕ that takes as input the current prediction $f_\theta(x)$ and the target y and outputs the corresponding loss. The meta-loss $\mathcal{L}_\phi$ is optimized at "meta-train" time and then is used for solving new regression tasks at "meta-test" time.

At each step of meta-training, the current parameters ϕ are used for updating θ via Eq. (2). Next, the parameters ϕ are evaluated based on the progress they have made towards minimizing a task-loss denoted by $\mathcal{L}_T$, which can be the squared error

$(y - f_\theta(x))^2$. Specifically, the parameters ϕ are updated as

$$\phi_{new} = \phi - \eta \nabla_\phi \mathcal{L}_T(y, f_{\theta_{new}}(x)) \tag{3}$$

where

$$\nabla_\phi \mathcal{L}_T = \nabla_f \mathcal{L}_T \nabla_{\theta_{new}} f_{\theta_{new}} \nabla_\phi [\theta - \alpha \nabla_\theta \mathbb{E}_{x,y} \mathcal{L}_\phi(y, f_\theta(x))] \tag{4}$$

The above computation can be alternatively performed using automatic differentiation. Further, if $M > 1$ inner gradient steps (updates of θ) are performed, then the gradient of Eq. (4) requires back-propagation through a chain of all the inner update steps.

Overall, we attempt to obtain a meta-loss $\mathcal{L}_\phi$ that will give *better updates towards minimizing* $\mathcal{L}_T$ *as compared to minimizing* $\mathcal{L}_T$ *directly.* This is shown in the meta-train Algorithm 1.

Algorithm 1: ML3 at meta-train

initialize ϕ

while *not converged* **do**

  initialize θ

  sample task samples x, y from T

  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathbb{E}_{x,y} \mathcal{L}_\phi(y, f_\theta(x))$ ▷ `can be M inner steps`

  $\phi \leftarrow \phi - \eta \nabla_\phi \mathbb{E}_{x,y} \mathcal{L}_T(y, f_\theta(x))$

end

return meta-loss parameters ϕ

Note that we can provide additional information about the task at meta-train time, which can influence the learning procedure. For example, the task-loss $\mathcal{L}_T$ can be

$$\mathcal{L}_T = \beta(y - f_{\theta_{new}}(x))^2 + \gamma \mathcal{L}_{extra} \tag{5}$$

This is referred to as shaping the loss in Bechtle et al. (2020) because in a way it shapes the loss landscape.

Finally, after meta-training, each meta-test task is trained independently using the learned loss function $\mathcal{L}_\phi$. According to Bechtle et al. (2020), the obtained meta-loss can be used across multiple architectures with a mild variation in performance.

3 Examples

Although there are examples related to reinforcement learning as well, in this section only the examples related to supervised learning are presented.

3.1 Supervised learning

Here we present an experiment pertaining to sine function regression. The tasks are different sine functions given as $y = Asin(x - \omega)$, where $A \in [0.2, 5.0]$, $\omega \in [-\pi, \pi]$, and $x \in [-2.0, 2.0]$. At meta-train time just the task given by $y = sin(x - \pi)$ is used (see Fig. 1). The model f_θ is a feedforward NN with 2 hidden layers and 40 hidden units and the learning rates are fixed to $\alpha = \eta = 0.001$.

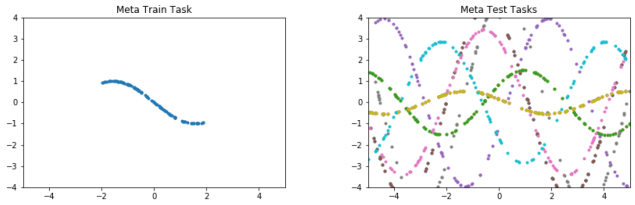

Fig. 1. Figure 2a-b from Bechtle et al. (2020): Meta-train/test tasks.

For each update at meta-train time the task function is sampled at random locations (although it is not very clear in the paper whether it is indeed SGD or GD), the parameters θ are updated given the gradients $\nabla_\theta \mathbb{E}_{x,y} \mathcal{L}_\phi(y, f_\theta(x))$ and then the parameters ϕ are updated given the gradients $\nabla_\phi \mathbb{E}_{x,y} \mathcal{L}_T(y, f_\theta(x))$. In this example $\mathcal{L}_T(y, f_\theta(x)) = (y - f_\theta(x))^2$.

Fig. 2 shows how the final MSE on the training task (after 100 gradient steps) changes depending on the loss function it is trained with. For example, for the blue line, in order to obtain the y-value corresponding to the value of 200 in the x-axis we have to train f_θ for 100 gradient steps with the loss function $\mathcal{L}_\phi$ corresponding to the meta-train iteration 200. What is not clear from the paper is whether $M = 100$ in Algorithm 1 and thus, ϕ is updated on the whole chain of 100 updates of θ, or $M = 1$ and thus ϕ is updated based on a single update of θ, but in order to produce Fig. 2 f_θ is trained separately for 100 iterations. Finally, the orange line corresponds to the final MSE on the training task if the loss function used is $\mathcal{L}_T$ directly. Obviously, this MSE does not depend on the meta-train iteration and this is why it is constant.

Fig. 2. Figure 2c from Bechtle et al. (2020): MSE on train-task for training f_θ with $\mathcal{L}_T$ directly (orange) and for training with $\mathcal{L}_\phi$ corresponding to the meta-train iteration (blue). The uncertainty comes from the initialization of ϕ and θ and the random samples for the train-task. In total 5 random seeds have been used for obtaining the results.

At meta-test time 10 test-tasks are drawn and 10 models are trained independently both with $\mathcal{L}_T$ directly and with $\mathcal{L}_\phi$ (which has been already trained at meta-train time). The results are shown in Fig. 3.

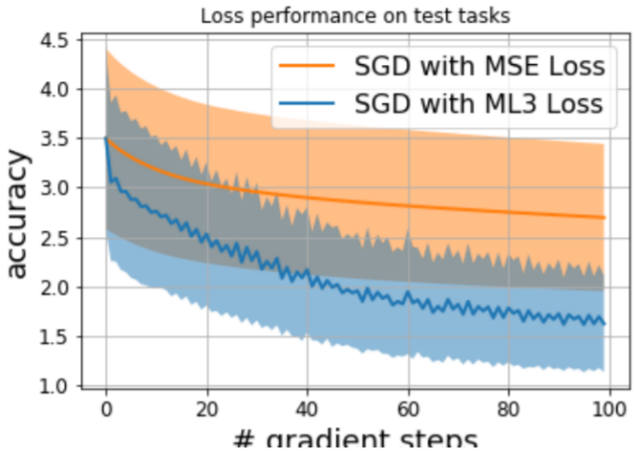

Fig. 3. Figure 2d from Bechtle et al. (2020): $\mathcal{L}_T$ on 10 test-tasks for making updates with $\mathcal{L}_T$ directly (orange) and for making updates with $\mathcal{L}_\phi$ (blue; which has been already trained at meta-train time). The uncertainty comes from the initialization of θ, the random test-tasks and the random samples for the test-tasks. In total 5 random seeds have been used for obtaining the results.

3.2 Toy example: Shaping loss

Suppose we have 1000 samples from $y = sin(vx)$ for $x \in [-1, 1]$. We create a model $f_\omega(x) = sin(\omega x)$ and we aim to identify the true parameter v with the model parameter ω. An indicative MSE loss is shown in Fig. 4 (note the severe non-convexity).

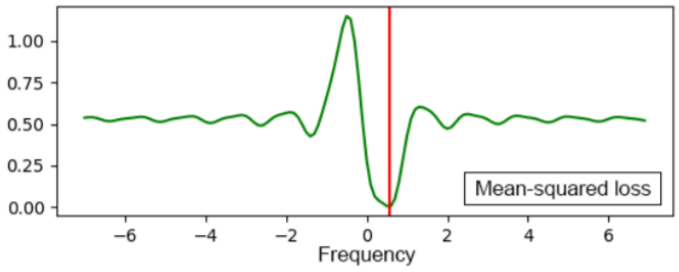

Fig. 4. Figure 5a bottom from Bechtle et al. (2020): Typical MSE loss for learning frequency v.

Suppose next that we use Algorithm 1 for training a meta-loss function $\mathcal{L}_\phi$ by using the task-loss $\mathcal{L}_T = (\omega - v)^2$, i.e., we assume we know the true v. Then the obtained loss function $\mathcal{L}_\phi$ is convex as shown in Fig. 5.

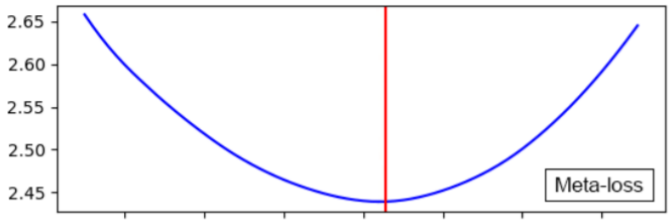

Fig. 5. Figure 5a top from Bechtle et al. (2020): Obtained meta-loss function with shaping loss as task-loss.

Further, Fig. 6 shows the regression loss (most likely they mean the MSE loss $\mathbb{E}_x[sin(vx) - sin(\omega x))^2]$) for the case where ω is updated via the MSE loss (orange), the case where it is updated via the meta-loss obtained with the MSE loss as task-loss (blue) and for the case where it is updated via the meta-loss obtained with the shaping loss as task-loss (green). The two cases of ML3 are delineated in Algorithm 2. It is evident that without shaping the loss landscape, the optimization is prone to getting stuck at a local minimum.

Algorithm 2: ML3 at meta-train: toy example

initialize ϕ

while *not converged* **do**

  initialize ω

  sample task samples x, y from T

  $\omega \leftarrow \omega - \alpha \nabla_\omega \mathbb{E}_{x,y} \mathcal{L}_\phi(sin(vx), sin(\omega x))$

  if *MSE task-loss (blue)* **then**

    $\mathcal{L}_T = (sin(vx) - sin(\omega x))^2$

  else if *Shaping loss (green)* **then**

    $\mathcal{L}_T = (v - \omega)^2$

  $\phi \leftarrow \phi - \eta \nabla_\phi \mathbb{E}_{x,y} \mathcal{L}_T$

end

return meta-loss parameters ϕ

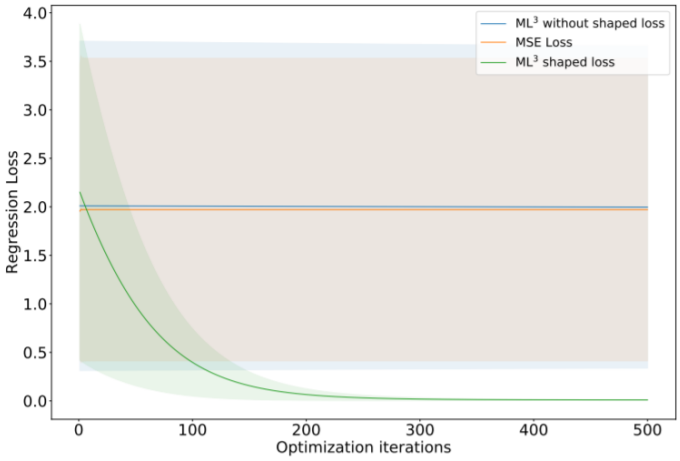

Fig. 6. Figure 5b from Bechtle et al. (2020): MSE loss at meta-test time for 3 cases of loss functions (see also Algorithm 2).

It seems that when the task-loss is so pathological, trying to learn a meta-loss based on it will not help at meta-test time. As a final note, it does not seem that we can use the learned loss function of this example for test-tasks different than the train-task. It was just a toy example.

3.3 Shaping loss via physics prior for inverse dynamics learning

Consider the dynamical system

$$M(q)\ddot{q} + F(q, \dot{q}) = u \tag{6}$$

where data for $q, \dot{q}, \ddot{q}, u$ are provided. Indicative ways to learn the map $u \mapsto q, \dot{q}, \ddot{q}$ include estimating $M(q)$ from analytical expressions (reference provided in the paper), using a black box map, and using a parametrized inertia matrix $M_\theta(q)$ (see Lutter et al., 2019). The latter uses a physics-based MSE loss defined using Eq. (6).

For improving sample efficiency Bechtle et al. (2020) proposes to learn a meta-loss $\mathcal{L}_\phi$ by using Algorithm 1 with task-loss given as $\mathcal{L}_T = (M(q) - M_\theta(q))^2$, where $M(q)$ (seems to be; it is not very clear) the analytical estimate mentioned above (see Algorithm 3). Thus, it acts as a physics-based prior for learning the loss. What is not very clear in the paper is whether $(M(q) - M_\theta(q))^2$ is added to the MSE loss to form the task-loss (like in Eq. (5)) or the task-loss is equal to just $(M(q) - M_\theta(q))^2$. From the paper the latter seems true, but then how is $M_\theta(q)$ faithful to the data as well?

Algorithm 3: ML3 at meta-train: shaping loss via physics prior

initialize ϕ

while *not converged* **do**

  initialize θ

  sample task samples $q, \dot{q}, \ddot{q}, u$ from T

  $\theta \leftarrow \theta - \alpha \nabla_\theta \mathbb{E}_{q,u} \mathcal{L}_\phi(u, M_\theta(q)\ddot{q} + F(q, \dot{q}))$

  if *MSE task-loss (blue)* **then**

    $\mathcal{L}_T = (u - M_\theta(q)\ddot{q} - F(q, \dot{q}))^2$

  else if *Shaping task-loss (green)* **then**

    $\mathcal{L}_T = (M(q) - M_\theta(q))^2$

  $\phi \leftarrow \phi - \eta \nabla_\phi \mathbb{E}_{q,u} \mathcal{L}_T$

end

return meta-loss parameters ϕ

In Fig. 7 the regression loss is shown for the case where θ is updated via the MSE loss (orange), via the meta-loss learned with MSE as task-loss (blue) and via the meta-loss learned with the physics prior as task-loss (green). The two cases of ML3 are delineated in Algorithm 3.

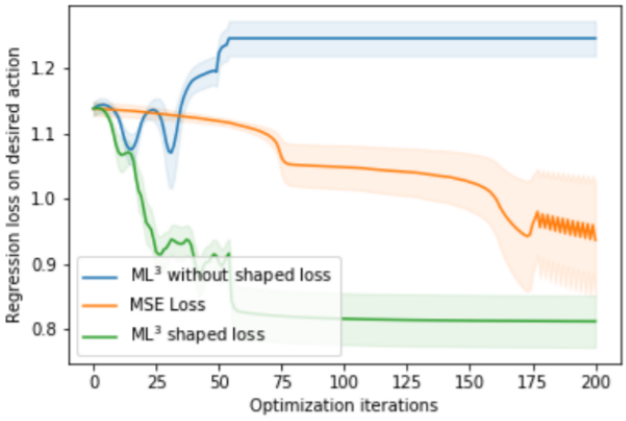

Fig. 7. Figure 5c from Bechtle et al. (2020): MSE loss at meta-test time for 3 cases of loss functions.

References

Bechtle, S. et al. (2020). "Meta-Learning via Learned Loss". *arXiv:1906.05374 [cs, stat]*. arXiv: 1906.05374 [cs, stat].

Houthooft, R. et al. (2018). "Evolved Policy Gradients". *Advances in Neural Information Processing Systems*, pp. 5400–5409.

Lutter, M., Ritter, C., and Peters, J. (2019). "Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning". *arXiv preprint arXiv:1907.04490*. arXiv: 1907.04490.

Sung, F., Zhang, L., Xiang, T., Hospedales, T., and Yang, Y. (2017). "Learning to Learn: Meta-Critic Networks for Sample Efficient Learning". *arXiv preprint arXiv:1706.09529*. arXiv: 1706.09529.