ENHANCING META LEARNING VIA MULTI-OBJECTIVE SOFT IMPROVEMENT FUNCTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Meta-learning tries to leverage information from similar learning tasks. In the commonly-used bilevel optimization formulation, the shared parameter is learned in the outer loop by minimizing the average loss over all tasks. However, the converged solution may be compromised in that it only focuses on optimizing on a small subset of tasks. To alleviate this problem, we consider meta-learning as a multi-objective optimization (MOO) problem, in which each task is an objective. However, existing MOO solvers need to access all the objectives' gradients in each iteration, and cannot scale to the huge number of tasks in typical meta-learning settings. To alleviate this problem, we propose a scalable gradient-based solver with the use of mini-batch. We provide theoretical guarantees on the Pareto optimality or Pareto stationarity of the converged solution. Empirical studies on various machine learning settings demonstrate that the proposed method is efficient, and achieves better performance than the baselines, particularly on improving the performance of the poorly-performing tasks and thus alleviating the compromising phenomenon.

1 INTRODUCTION

Meta-learning, also known as "learning to learn", aims to enable models to learn more effectively by leveraging information from many similar learning tasks (Hospedales et al., 2020). In recent years, meta-learning (Vilalta & Drissi, 2002) has received much attention for its fast adaptation to new learning scenarios with limited data (Kao et al., 2021; Finn et al., 2017; Snell et al., 2017; Lee et al., 2019; Nichol et al., 2018; Deleu et al., 2022; Rajeswaran et al., 2019). Meta-learning is usually formulated as a bi-level optimization problem (Franceschi et al., 2018; Hong et al., 2020), which finds task-specific parameters in the inner level and minimizes the average (expected) loss over different tasks in the outer level.

Recently, Wang et al. (2021) reformulate meta-learning as a multi-task learning problem. From this perspective, minimizing the average loss in the outer level using (stochastic) gradient descent may not always be desirable. Specifically, it may suffer from the compromising (or conflicting) phenomenon (Liu et al., 2021a; Sener & Koltun, 2018), in which the converged solution only focuses on minimizing the losses of a small subset of tasks while ignoring the others (Yu et al., 2020; Liu et al., 2021a; Sener & Koltun, 2018). This compromised solution may thus lead to poor performance.

To alleviate this problem, we propose reformulating meta-learning as a multi-objective optimization (MOO) problem (Sener & Koltun, 2018), in which each task is an objective. The performance of all the tasks (objectives) are then considered during optimization (Emmerich & Deutz, 2018). A popular class of MOO solvers is the gradient-based approach (Liu et al., 2021a; Yu et al., 2020; Sener & Koltun, 2018; Navon et al., 2022; Liu et al., 2021b), with prominent examples such as the multiple-gradient descent algorithm (MGDA) (Désidéri, 2012; Sener & Koltun, 2018), PCGard (Yu et al., 2020), and CAGard (Liu et al., 2021a). In each iteration, they find a common descent direction among all objective gradients, instead of simply optimizing the average performance over all objectives.

Existing gradient-based MOO methods rely on using gradients from all the objectives. However, when formulating meta-learning as a MOO problem with each task being an objective, computing all these gradients in each iteration can become very expensive, as the number of objectives (i.e., tasks) can be huge. For example, in 5-way 1-shot classification on the MiniImagenet dataset, the total number of meta-training tasks is $\binom{64}{5} \approx 7 \times 10^6$.

To address this challenge, we design a scalable MOO solver by using the improvement function (Miettinen & Mäkelä, 1995; Mäkelä et al., 2016; Montonen et al., 2018) with the help of mini-batch. On the other hand, a trivial extension of existing gradient-based MOO methods with the use of mini-batch does not guarantee Pareto optimality and has poor performance in practice.

Our main contributions are as follows: (i) To alleviate the compromising phenomenon, we reformulate meta-learning as a multi-objective optimization problem in which each task is an objective; (ii) To handle the possibly huge number of tasks, we propose a scalable gradient-based solver with the use of mini-batch. (iii) We provide theoretical guarantees on the Pareto optimality or Pareto stationarity of the converged solution. (iv) Empirical studies on few-shot regression, few-shot classification, and reinforcement learning demonstrate that the proposed method achieves better performance, particularly in improving the performance of the poorly-performing tasks and thus alleviating the compromising phenomenon.

2 BACKGROUND

Multi-objective Optimization (MOO). In MOO (Marler & Arora, 2004), one aims to minimize[1] $m \geq 2$ objectives $f_1(x), \ldots, f_m(x)$:

$$\min_x [f_1(x), \ldots, f_m(x)]. \tag{1}$$

Definition 2.1. *(Global Pareto optimality) (Miettinen, 2012; Mäkelä et al., 2016) x^* is global Pareto optimal if there does not exist another x such that $f_\tau(x^*) \leq f_\tau(x)$ for all $\tau \in \{1, \ldots, m\}$, and $f_{\tau'}(x^*) < f_{\tau'}(x)$ for at least one $\tau' \in \{1, \ldots, m\}$.*

The *Pareto front* (PF) is the set of multi-objective values of all global Pareto-optimal solutions.

Definition 2.2. *(Pareto stationarity) (Miettinen, 2012; Désidéri, 2012) x^* is Pareto-stationary iff there exist $\{u_i\}_{i=1}^m$ such that $\| \sum_{\tau=1}^m u_\tau \nabla_x f_\tau(x) \| = 0, u_\tau \geq 0 \; \forall \tau$ and $\sum_{\tau=1}^m u_\tau = 1$.*

Note that global Pareto optimal solutions are also Pareto stationary (Désidéri, 2012). Analogous to the extension from a stationary point to an ϵ-stationary point (Lin et al., 2020), we extend Pareto stationarity to ϵ-Pareto stationarity. Obviously, 0-Pareto stationarity reduces to Pareto stationarity.

Definition 2.3. *(ϵ-Pareto stationarity). A point x is ϵ-Pareto-stationary iff there exist $\{u_i\}_{\tau=1}^m$ such that $\| \sum_{\tau=1}^m u_\tau \nabla_x f_\tau(x) \| \leq \epsilon, u_\tau \geq 0 \; \forall \tau$ and $\sum_{\tau=1}^m u_\tau = 1$.*

Definition 2.4. *(Improvement function) (Montonen et al., 2018) The improvement function of problem (1) is:*

$$H(x, x') = \max_{\tau=1\ldots,m} \{f_\tau(x) - f_\tau(x')\}. \tag{2}$$

Note that x^* satisfying $x^* = \arg\min_x H(x, x^*)$ (intuitively, x^* cannot be further improved) is a Pareto stationary point (Montonen et al., 2018). To find x^*, one can perform steepest descent on H:

$$x^{s+1} = x^s + \beta d^*, \quad d^* = \arg\min_d H(x^s + d, x^s) + \frac{\lambda'}{2}\|d\|^2, \tag{3}$$

where x^s is the iterate at iteration s, β is the learning rate satisfying $H(x^s + \beta d, x^s) < H(x^s, x^s)$, and λ' is a hyper-parameter. It can be shown that when $s \to \infty$, x^s is Pareto stationary (Montonen et al., 2018).

In this paper, we focus on gradient-based MOO methods, including MGDA (Désidéri, 2012; Sener & Koltun, 2018), PCGard (Yu et al., 2020), and CAGard (Liu et al., 2021a). They assign weights to each objective's gradient and find a common descent direction that decreases the losses of all objectives. For example, MGDA finds the direction $g^*(x) = \sum_{\tau=1}^m \gamma_\tau^* \nabla_x f_\tau(x)$ in each iteration, where

$$\{\gamma_\tau^*\} = \arg\min_{\{\gamma_\tau\}} \left\| \sum_{\tau=1}^m \gamma_\tau \nabla_x f_\tau(x) \right\|^2 \text{ s.t. } \sum_{\tau=1}^m \gamma_\tau = 1, \gamma_\tau \geq 0, \forall \tau. \tag{4}$$

[1] Without loss of generality, we consider minimization in this paper.

Meta-Learning. Meta-learning aims to achieve good performance with limited data and computation (Hospedales et al., 2020). Most of them are gradient-based (Nichol et al., 2018; Deleu et al., 2022; Rajeswaran et al., 2019; Zhou et al., 2019; Shu et al., 2019) or metric-based (Snell et al., 2017; Lee et al., 2019; Vinyals et al., 2016). Let $\mathcal{T}$ be the set of all m tasks, and w be the shared model parameter. For a task $\tau \in \mathcal{T}$, let D_τ be its dataset and $\mathcal{L}_\tau$ the corresponding loss function. It tries to obtain task-specific parameter w_τ from the shared w as $w_\tau^*(w)$. Meta-learning is usually formulated as the following bilevel optimization problem (Ji et al., 2021):

$$\min_w \sum_{\tau=1}^m \mathcal{L}_\tau(w_\tau) \text{ s.t. } w_\tau = w_\tau^*(w). \tag{5}$$

The inner-subproblem learns the task-specific parameter w_τ for each τ, while the outer-subproblem learns w by minimizing the average loss over tasks in $\mathcal{T}$. As m can be very large, usually a mini-batch B of tasks are uniformly sampled from $\mathcal{T}$, and w is then updated as $w^{s+1} = w^s - \beta \frac{1}{|B|} \sum_{\tau \in B} \mathcal{L}_\tau(w_\tau^*(w))$ (Finn et al., 2017).

By taking each $\mathcal{L}_\tau(w_\tau^*(w))$ in (5) as an objective, this can be regarded as a weighted sum in multi-task learning (Wang et al., 2021). As observed in (Sener & Koltun, 2018; Yu et al., 2020), gradient descent on this weighted sum can suffer from the *compromising* phenomenon, in which the loss obtained on some task τ' can be much larger than the losses on the other tasks.

3 Soft Improvement Multi-Objective meta-Learning (SIMOL)

To mitigate the compromising phenomenon, we take the view of meta-learning as multi-task learning in (Wang et al., 2021) one step further and consider the meta-learning problem as the following multi-objective optimization (MOO) problem:

$$\min_w (\mathcal{L}_1(w_1^*(w)), \ldots, \mathcal{L}_m(w_m^*(w))), \tag{6}$$

in each each task corresponds to an objective. This considers all the individual tasks instead of simply considering the total loss over all tasks (Liu et al., 2021a). Recently, Ye et al. (2021) also used multi-objective learning into meta-learning. However, their focus is not on addressing the compromising phenomenon and they do not treat each task as an objective. Instead, besides minimizing the average task loss in (5), they consider adding some other objectives such as robustness to adversarial attacks. Moreover, MGDA is then used to find the Pareto optimal solution. However, as in other gradient-based MOO methods (Yu et al., 2020; Liu et al., 2021a), MGDA requires collecting gradients from all m objectives in each iteration (as can be seen from its optimization problem (4)). This is computationally feasible only when there are a small number of objectives.[2] When each task is treated as an objective, the number of objectives can easily be in the millions (as in performing 5-way 1-shot classification on MiniImagenet).

To alleviate this problem, one solution is to use only a mini-batch of objectives in each iteration. For example, when a subset B of objectives is used, MGDA's optimization problem (4) becomes:

$$\min_{\{\gamma_\tau\}} \left\| \sum_{\tau \in B} \gamma_\tau \nabla_\tau (\mathcal{L}_\tau(w_\tau^*(w)) \right\|^2 \text{ s.t. } \sum_{\tau \in B} \gamma_\tau = 1, \gamma_\tau \geq 0, \forall \tau,$$

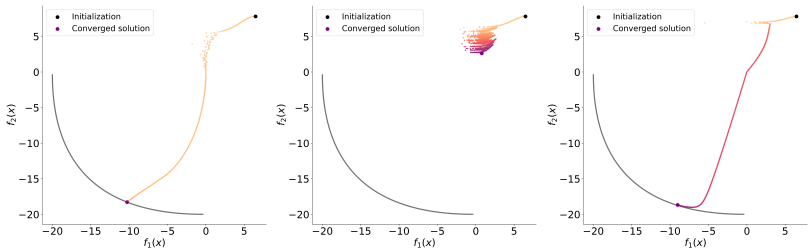
However, the descent direction then only considers objectives in B, and the original normalization constraint $\sum_{\tau=1}^m \gamma_\tau = 1$ in (4) is also changed to $\sum_{\tau \in B} \gamma_\tau = 1$. The obtained solution can no longer be guaranteed to be Pareto optimal. In the following, we demonstrate this empirically by using a simple toy example with two objectives ($f_1(x)$ and $f_2(x)$, where $x \in \mathbb{R}^2$) from (Liu et al., 2021a; Navon et al., 2022).[3] As can be seen from Figure 1, mini-batch MGDA (with a mini-batch of 1) cannot converge to the Pareto front.

3.1 Soft Improvement Function

In this section, we design a scalable MOO solver with the use of the improvement function (Montonen et al., 2018). The proposed solver is agnostic to the number of tasks, while still theoretically guaranteeing that the solution is Pareto optimal.

[2] In the meta-learning experiments of (Ye et al., 2021), they only consider two objectives.

[3] Definitions for f_1, f_2 and the environment setup are in Appendix A.1.



(a) MGDA. (b) mini-batch MGDA. (c) SIMOL.

Figure 1: Convergence of (batch) MGDA, mini-batch MGDA and the proposed SIMOL, with mini-batch size 1, one a two-objective toy dataset. The Pareto front is shown in black.

Using Definition 2.4, the improvement function for problem (6) is:

$$H(w, w') = \max_{\tau=1\ldots,m} \left\{ \mathcal{L}_\tau(w^*_\tau(w)) - \mathcal{L}_\tau(w^*_\tau(w')) \right\}. \tag{7}$$

We first rewrite the above as (the correctness will be proven in Lemma 3.1 below):

$$\max_\pi \mathbb{E}_{\tau\sim\pi} \left[\mathcal{L}_\tau(w^*_\tau(w)) - \mathcal{L}_\tau(w^*_\tau(w')) \right], \tag{8}$$

where π is a probability density function on τ. The following Lemma shows that (7) and (8) are equivalent, when π is the Dirac delta distribution concentrated on the task corresponding to the maximum in (7). All the proofs are in Appendix B.

Lemma 3.1. $H(w, w') = \max_\pi \mathbb{E}_{\tau\sim\pi} \left[\mathcal{L}_\tau(w^*_\tau(w)) - \mathcal{L}_\tau(w^*_\tau(w')) \right]$.

Using (3) and Lemma 3.1, w is updated as

$$w^{s+1} = w^s + \beta d^*, \tag{9}$$

$$d^* = \arg\min_d \left(\max_\pi \mathbb{E}_{\tau\sim\pi} \left[\mathcal{L}_\tau(w^*_\tau(w^s + d)) - \mathcal{L}_\tau(w^*_\tau(w^s)) \right] \right) + \frac{\lambda'}{2} \|d\|^2. \tag{10}$$

Taking the first-order approximation

$$\mathcal{L}_\tau(w^*_\tau(w^s + d)) \simeq \mathcal{L}_\tau(w^*_\tau(w^s)) + \nabla_w \mathcal{L}_\tau(w^*_\tau(w^s))^\top d, \tag{11}$$

the minimax theorem (Simons, 1995) can then be applied to swap the $\min$ and $\max$ operators in (10), leading to

$$(\pi^*, d^*) = \arg\max_\pi \left(\min_d \mathbb{E}_{\tau\sim\pi} \left[\mathcal{L}_\tau(w^*_\tau(w^s + d)) - \mathcal{L}_\tau(w^*_\tau(w^s)) \right] + \frac{\lambda'}{2} \|d\|^2 \right). \tag{12}$$

The following Proposition shows that the inner minimization problem has a closed-form solution.

Proposition 3.2. $\min_d \mathbb{E}_{\tau\sim\pi} \left[\mathcal{L}_\tau(w^*_\tau(w^s + d)) - \mathcal{L}_\tau(w^*_\tau(w^s)) \right] + \frac{\lambda'}{2} \|d\|^2 = \frac{-1}{2\lambda'} \|\mathbb{E}_{\tau\sim r} \nabla_w \mathcal{L}_\tau(w^*_\tau(w))|_{w=w^s}\|^2$, *and the optimal* d *is* $d^* = -\frac{1}{\lambda'} \mathbb{E}_{\tau\sim\pi} [\nabla_w \mathcal{L}_\tau(w^*_\tau(w)|_{w=w^s}]$.

The expectation in Proposition 3.2 requires sampling tasks from π. An easier alternative is to sample tasks from the uniform distribution $U(\cdot)$ over the set $\mathcal{T}$ of all tasks, and then weighting each sampled task τ with

$$r(\tau) \equiv \pi(\tau)/U(\tau). \tag{13}$$

Note that

$$\mathbb{E}_{\tau\sim U} r(\tau) = \sum_\tau U(\tau)\pi(\tau)/U(\tau) = \sum_\tau \pi(\tau) = 1. \tag{14}$$

We further parameterize r as a neural network r_θ with parameter θ. Using Proposition 3.2, we can then rewrite (12) as

$$\theta^* = \arg\max_\theta \frac{-1}{2\lambda'} \left[\mathbb{E}_{\tau\sim U} r_\theta(\tau) \nabla_w \mathcal{L}_\tau(w^*_\tau(w))|_{w=w^s} \right]^2 - \frac{\lambda''}{2} (\mathbb{E}_{\tau\sim U} r_\theta(\tau) - 1)^2, \tag{15}$$

where the last term (with another hyper-parameter λ'') is a penalty for enforcing the constraint in (14). For notational simplicity, we denote the objective in (15) by $K(\theta)$.

In principle, θ^* can be obtained from (15) by gradient ascent. However, problem (15) involves an expectation over tasks. Recall that we have a total of m tasks, and m can be huge. Hence, using all of them to compute this expectation may not be feasible. Instead, Let B be a mini-batch of k tasks, and denote the the mini-batched version of the objective in (15) as:

$$\tilde{K}_B(\theta) \equiv \frac{-1}{2\hat{\lambda}'}\left\|\frac{1}{|B|}\sum_{\tau\in B} r_\theta(\tau)\nabla_w \mathcal{L}_\tau(w_\tau^*(w))|_{w=w^s}\right\|^2 - \frac{\hat{\lambda}''}{2}\left(\frac{1}{|B|}\sum_{\tau\in B} r_\theta(\tau) - 1\right)^2,$$

where $\hat{\lambda}', \hat{\lambda}''$ are another set of hyper-parameters (which will be set in Proposition 3.3) corresponding to λ', λ'' in (15). Note that $\tilde{K}_{\mathcal{T}}(\theta) = K(\theta)$. Let $\mathcal{B}$ be the set of all size-k mini-batches ($k > 1$). The following Proposition bounds the difference between $K(\theta)$, the original objective in (15), and the version $\frac{1}{|\mathcal{B}|}\sum_B \tilde{K}_B(\theta)$ based on mini-batches.

Proposition 3.3. *Set* $\frac{\lambda'}{C_1|\mathcal{B}|} = \hat{\lambda}'$, $\lambda'' C_1|\mathcal{B}|k^2 = \hat{\lambda}''$. *We have:*

$$\left(K(\theta) - \frac{1}{|\mathcal{B}|}\sum_{B\in\mathcal{B}} \tilde{K}_B(\theta)\right)^2 \leq \frac{C_2 k|\mathcal{B}|G_1}{\lambda'} + C_2 k|\mathcal{B}|\lambda'' G_2,$$

where $C_1 \equiv \frac{k^2}{\binom{m-2}{k-2}m^2}$, *and* $C_2 \equiv \left[\binom{m-1}{k-1} - \frac{1}{2}\binom{m-2}{k-2}\right] / \binom{m-1}{k-1}m^2\binom{m-2}{k-2}$,$G_1 \equiv \frac{1}{k|\mathcal{B}|}\sum_{B\in\mathcal{B}}\sum_{\tau\in B}\|r_\theta(\tau)\nabla_w\mathcal{L}_\tau(w_\tau^*(w)|_{w=w^s}\|^2$, *and* $G_2 \equiv \frac{1}{k|\mathcal{B}|}\sum_{B\in\mathcal{B}}\sum_{\tau\in B}[r_\theta(\tau)-1]^2$.

Corollary 3.3.1. *When* $k \ll m$, $\left(K(\theta) - \frac{1}{|\mathcal{B}|}\sum_{B\in\mathcal{B}}\tilde{K}_B(\theta)\right)^2 \leq \frac{G_1}{k\lambda'} + \frac{G_2}{k}\lambda''$.

In the experiments, $m \geq 10^6$, $k \approx 10^2$, and $G_1, G_2 \leq 10^4$. When $\lambda' \geq 10^2, \lambda'' \leq 10^{-2}$, we have $(K(\theta) - \frac{1}{|\mathcal{B}|}\sum_{B\in\mathcal{B}}\tilde{K}_B(\theta))^2 \leq 0.2$, which is small. Thus, Proposition 3.3 shows that $K(\theta)$ can be decomposed into mini-batches as $\frac{1}{|\mathcal{B}|}\sum_{B\in\mathcal{B}}\tilde{K}_B(\theta)$. This allows us to update θ by SGD (with learning rate β') over the task mini-batches:

$$\theta^{s+1} = \theta^s + \beta'\nabla_\theta \tilde{K}_B(\theta^s). \tag{16}$$

Similarly, we approximate d^* by its mini-batch approximation $\tilde{d}^* = -\frac{1}{\lambda'|B|}[\sum_{\tau\in B}\nabla_w\mathcal{L}_\tau(w_\tau^*(w))|_{w=w^s}]$ and update w by: $w^{s+1} = w^s + \beta\tilde{d}^*$.

The whole procedure, which will be called Soft Improvement Multi-Objective Meta Learning (SIMOL), is shown in Algorithm 1. Step 4 trains the base learner. In the experiments, we use the two popular meta-learning algorithms MAML (Finn et al., 2017) and prototypical network (PN) (Snell et al., 2017) as the backbone. For MAML, the base learner is updated by

$$w_\tau^*(w) = w - \alpha\nabla_w\mathcal{L}_\tau(w),$$

where α is the base learner's learning rate. For the PN,

$$w_\tau^*(w) = \frac{1}{|Q_\tau|N_C}\sum_{x\in Q_\tau}\frac{\exp(-\|f_w(x)-c_k\|^2)}{\sum_{k'}\exp(-\|f_w(x)-c_{k'}\|^2)}, \quad c_k = \frac{1}{|S_k|}\sum_{(x_i,y_i)\in S_k} f_w(x_i),$$

where S_k is the set of examples belonging to class k, Q_τ is the set of query examples for task τ, N_C is the number of classes per epoch, and f_w is the model with parameter w.

Algorithm 1: Soft Improvement Multi-Objective Meta Learning (SIMOL)

Input: $\mathcal{T}$, batch size B, learning rates β and β' for w, $\tilde{d}^* = 0$ and θ, respectively.

1 **for** $s = 1, 2, \ldots, S$ **do**
2 Reset $\tilde{d}^* = 0$;
3 **for** $\tau = 1, 2, \ldots, B$ **do**
4 obtain $r_{\theta^s}(\tau)\nabla_{w^s}\mathcal{L}_\tau(w_\tau^*(w^s))$ for task τ;
5 $\tilde{d}^* = \tilde{d}^* - r_{\theta^s}(\tau)\nabla_{w^s}\mathcal{L}_\tau(w_\tau^*(w^s))$;
6 $w^{s+1} = w^s + \beta\frac{1}{B}\tilde{d}^*$;
7 $\theta^{s+1} = \theta^s + \beta'\nabla_{\theta^s}\tilde{K}_B(\theta^s)$;

4 CONVERGENCE ANALYSIS

Define

$$R(\theta, w) \equiv \mathbb{E}_{\tau\sim U}\left[\perp(r_\theta(\tau))[\mathcal{L}_\tau(w_\tau^*(w)]\right] + \Delta(\theta) - \frac{\lambda''}{2}(\mathbb{E}_{\tau\sim U} r_\theta(\tau) - 1)^2, \tag{17}$$

where $\Delta(\theta) \equiv -\frac{1}{\lambda'}\mathbb{E}_{\tau\sim U} r_\theta(\tau) \perp [(\mathcal{L}_\tau(w_\tau^*(w) + d^*) - \mathcal{L}_\tau(w_\tau^*(w))]]\cdot \perp [\mathbb{E}_{\tau\sim U} r_\theta(\tau)[\mathcal{L}_\tau(w_\tau^*(w) + d^*) - \mathcal{L}_\tau(w_{tau}^*(w))]]$, and $\perp$ is the stop gradient operator.[4]

Lemma 4.1. *$\nabla_w R(\theta, w)|_{w=w^s} = -d^*$, and $\nabla_\theta R(\theta, w^s) = \nabla_\theta K(\theta)$.*

This allows interpreting the updates in (16) and (9) as performing Gradient Descent Ascent (GDA) (Singh et al., 2000) on (17). Thus, we can leverage game theoretical tools (Lin et al., 2020) in the analysis.

Let $\tilde{R}(\theta, w; B) := \perp \frac{1}{|B|}\sum_{\tau\in B}[r_\theta(\tau)][\mathcal{L}_\tau(w_\tau^*(w)] - \frac{1}{|B|\lambda'}\sum_{\tau\in B} r_\theta(\tau)[\perp [(\mathcal{L}_\tau(w_\tau^*(w) + d^*)\ \mathcal{L}_\tau(w_\tau^*(w))]\cdot \perp \left[\frac{1}{|B|}\sum_{\tau\in B} r_\theta(\tau)[\mathcal{L}_\tau(w_\tau^*(w) + d^*) - \mathcal{L}_\tau(w_\tau^*(w))]\right] - \lambda''(\frac{1}{|B|}\sum_{\tau\in B} r_\theta(\tau) - 1)^2$ be the mini-batch version of R, and $U(\mathcal{B})$ be the uniform distribution over task mini-batches. The following Theorem shows that Algorithm 1 converges to an ϵ-Pareto stationary point of (6).

Theorem 4.2. *Assume that (i) $\mathcal{L}_\tau(w_\tau^*(\cdot))$ is L-smooth and $r_\theta(\cdot)$ is μ-strongly concave. (ii) The domain of θ is a convex and bounded set with diameter $D > 0$, (iii) $\mathbb{E}_{B\sim U(\mathcal{B})}[\nabla_\theta \tilde{R}(\theta, w; B) - \nabla_\theta R(\theta, w)] = 0$, and $\mathbb{E}_{B\sim U(\mathcal{B})}\|\nabla_\theta \tilde{R}(\theta, w, B) - \nabla_\theta R(\theta, w)\|^2 \leq \sigma^2$. Assume the first-order approximation in (11), and take $\beta = \Theta\left(1/D^2\sigma^2(L^2 + \sigma^2)\right)$, $\beta' = \Theta(1/L\sigma^2)$. Algorithm 1 converges to an ϵ-Pareto stationary point of (6) with a rate of $O(1/\epsilon^8)$. If $\mathcal{L}_\tau(w_\tau^*(w))$ is also μ'-convex w.r.t. w and $\epsilon = 0$, the 0-Pareto stationary point is also global Pareto optimal.*

Assumption (i) is standard (Collins et al., 2020; Zhou et al., 2021; Finn et al., 2019); while (ii) and (iii) are from (Lin et al., 2020). Theorem 4.2 shows that the proposed method can obtain an ϵ-Pareto stationary point (or global Pareto optimal point for convex objectives) regardless of m.

Corollary 4.2.1. *Consider the MAML base learner update $w_\tau^*(w) = w - \alpha\nabla_w \mathcal{L}_\tau(w)$. Assume that $\nabla_w \mathcal{L}_\tau(w)$ is Hessian-Lipschitz, bounded and Lipschitz-continuous, w is bounded. then, Algorithm 1 converges to an ϵ-Pareto stationary point of (6) with a rate of $O(1/\epsilon^8)$.*

Corollary 4.2.1 is an application of Theorem 4.2 revealing that SIMOL with MAML could also converge to Pareto point. Convergence of the outer loop is slower than the $O(\frac{1}{\epsilon^2})$ rate of standard MAML (Fallah et al., 2020). However, standard MAML only guarantees convergence to stationary points but not to Pareto-stationary points. Moreover, as will be seen in Section 5.1, empirically, the proposed method has comparable or even slightly faster convergence speed than MAML and other baselines. Among the gradient-based MOO approaches, most of them, except CAGrad (Liu et al., 2021a), do not provide convergence rate analysis. However, the analysis in CAGrad relies on the assumption that all the task gradients are used in every epoch, which can have a very high per-epoch complexity, as will be empirically demonstrated in Section 5.2.

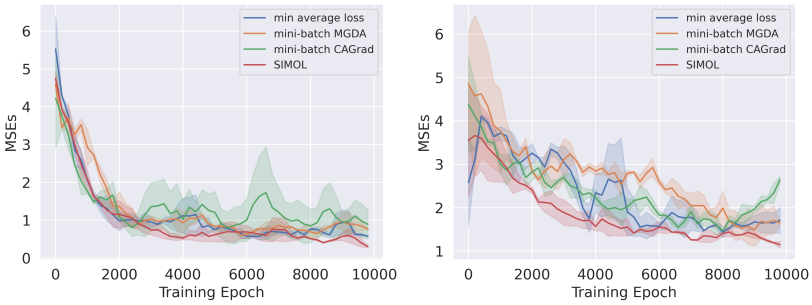
5 EXPERIMENTS

In this section, we perform experiments on few-shot regression (Section 5.1), few-shot classification (Section 5.2), and reinforcement learning (Section 5.3). All experiments are run on a single GeForce RTX 2080 Ti GPU and Intel(R) Xeon(R) CPU E5-2680. Our implementations are based on the Learn2Learn framework (Arnold et al., 2020), a popular open-source meta-learning library.

5.1 FEW-SHOT REGRESSION

In this section, we follow the setup in (Finn et al., 2017; Li et al., 2017). Experiments are performed on 5-shot and (the harder) 2-shot regression using a synthetic dataset. The target function for task τ is

[4]The stop gradient operator satisfies $\perp(h(x) = h(x)$, $\nabla_x \perp(h(\boldsymbol{x})) = 0$, where $h(\cdot)$ is any differentiable function.



(a) 5-shot. (b) 2-shot.

Figure 2: Convergence of MSE with number of training epochs.

$y = a_\tau \sin(x + b_\tau)$, where a_τ and b_τ are sampled uniformly from $[0.1, 5.0]$ and $[0, \pi]$, respectively. We generate $160,000$ meta-training tasks and $1,000$ meta-testing tasks. A multilayer perceptron with 2 fully-connected (FC) layers (each of size 32) and ReLU activation is used as meta-learner and re-weighting network. The re-weighting network uses all instances in each mini-batch as input.

The backbone meta-learning algorithm is MAML (Finn et al., 2017). We use Adam (Kingma & Ba, 2014), with an initial learning rate of 0.01, to update the base learners for 5 steps in the inner-loop. Hyperparameters for the backbone meta-learning algorithms follow the original papers. For the outer-loop, we compare (i) minimizing the (single objective) of overall task loss as in original MAML; versus performing MOO with (ii) mini-batch MGDA (Désidéri, 2012; Ye et al., 2021), (iii) mini-batch CAGrad, using the same hyper-parameters as (Liu et al., 2021a), and (iv) the proposed SIMOL (with $\beta = 0.0008$, $\beta' = 0.03$, $\hat{\lambda}'' = 0.01$, and $\hat{\lambda}' = 1$). The mini-batch size is always 16. We do not compare with the batch versions of MGDA/CAGrad, as computing all task gradients takes extremely large memory and time. The initial learning rate for the outer loop is 0.001. The experiment is repeated three times with different random seeds. For performance evaluation, we use the mean-squared-error (MSE) over all meta-testing tasks. We also report the worst-10% MSE, which is the average MSE for the 10% worst-performing meta-testing tasks.

Table 1 shows the results. As can be seen, SIMOL consistently outperforms SGD, MGDA, and CAGrad in terms of both the overall and worst-10% MSEs. Indeed, (mini-batch) MGDA and CAGard are even worse than the original MAML. The good performance of SIMOL in terms of the worst-10% MSE also lends support to the hypothesis that improving the poorly performing tasks can also aid overall performance.

Table 1: MSE (with 95% confidence interval) for few-shot regression. The best results are in bold.

	overall		worst-10%	
MAML	5-shot	2-shot	5-shot	2-shot
min average loss	0.43 ± 0.11	1.70 ± 0.11	2.13 ± 0.21	7.75 ± 0.48
mini-batch MGDA	0.60 ± 0.02	1.73 ± 0.10	2.62 ± 0.10	7.04 ± 0.51
mini-batch CAGrad	1.90 ± 0.24	1.82 ± 0.44	8.18 ± 0.63	8.23 ± 2.33
SIMOL	$\mathbf{0.34 \pm 0.04}$	$\mathbf{1.24 \pm 0.08}$	$\mathbf{1.69 \pm 0.18}$	$\mathbf{5.66 \pm 0.33}$

Figure 2 shows the convergence of MSE with the number of training epochs. As can be seen, SIMOL converges slightly faster than the other baselines. Figure 3 shows the learned functions on two example meta-testing tasks[5] in the 5-shot and 2-shot settings. As can be seen, SIMOL fits the ground-truth better than the baselines.

5.2 Few-shot Image Classification

In this section, we perform 5-way-1-shot and 5-way-5-shot classification on miniImageNet (Ravi & Larochelle, 2016) and tieredImageNet (Ren et al., 2018). Following (Finn et al., 2017), we split the

[5]For the 5-shot task, $a_\tau = 0.89, b_\tau = 1.71$; for the 2-shot task, $a_\tau = 1.29, b_\tau = 2.18$.

miniImageNet dataset into a meta-training set with 64 classes, a meta-validation set with 16 classes, and a meta-test set with 20 classes. The total number of meta-training tasks is $\binom{64}{5} \approx 7.6 \times 10^6$. Similarly, as in (Zhou et al., 2019), we split the tieredImageNet dataset into a meta-training set with 351 classes, a validation set with 97 classes, and a test set with 160 classes. The total number of meta-training tasks is $\binom{351}{5} \approx 4.3 \times 10^{10}$. for both datasets, we randomly select $1,000$ meta-testing tasks for evaluation.

We use two backbone meta-learning algorithms, MAML and prototypical network (PN) (Snell et al., 2017). Hyperparameters for the backbone meta-learning algorithms follow the original papers. Following (Finn et al., 2017; Li et al., 2017; Zintgraf et al., 2019), we use the CNN4[6] (LeCun et al., 2015) as the meta-learner. The optimizer is Adam. The learning rates are 0.003 for MAML-based methods and 0.001 for PN-based methods, respectively. The mini-batch size is 32. The other hyper-parameters are the same as in Section 5.1. More details can be found in Appendix D.

The proposed SIMOL is compared with (i) minimizingt the overall task loss as in standard MAML (named as *min average loss*), (ii) MGDA, and (iii) CAGrad. We also compare with MAML variants including (iv) Reptile (Nichol & Schulman, 2018), (v) FOMAML (Finn et al., 2017), (vi) Meta-MinibatchProx (Zhou et al., 2019), (vii) TSA-MAML (Zhou et al., 2021), and (viii) IMAML (Rajeswaran et al., 2019)) and the standard prototypical network (Snell et al., 2017), IX) MTL (Wang et al., 2021), a multi-task learning based meta learning. The evaluation metrics are similar to those in Section 5.1, but with accuracy instead of MSE. Experiments are repeated three times with different random seeds.

Results on miniImageNet and tieredImageNet are shown in Tables 3 and 4, respectively. As can be seen, with MAML, SIMOL consistently outperforms all the other baselines in terms of both overall and worst-10% accuracies. The same is also observed when the backbone meta-learning algorithm is the prototypical network. This demonstrates that SIMOL is useful in both gradient-based and metric-based meta-learning approaches.

Note that the performance of (mini-batch) MGDA and CAGrad are not good in terms of both overall and worst-10% accuracies, showing that they cannot be straightforwardly extended to the use of mini-batch. On the other hand, the batch versions of MGDA and CAGrad are computationally impractical. Table 2 compares the per-epoch running time of standard MAML, SIMOL, and batch MGDA and CAgrad. experiment is performed on 5-way-5-shot classification with the MAML algorithm on miniImageNet. As can be seen, while SIMOL has comparable per-epoch running time as MAML, batch MGDA and CAgrad are much more computationally expensive (around $432,000$ times slower).

Table 2: per-epoch running time for 5-way 5-shot classification on miniImageNet.

standard MAML	SIMOL	batch MGDA	batch CAGrad
2.0 sec	2.3 sec	6.9 days	5.6 days

5.3 Reinforcement Learning

In this experiment, we demonstrate that the proposed SIMOL can also be used in reinforcement learning. Experiments are performed on the HalfCheetach-Dir and Walker-Dir environments (Rothfuss et al., 2018). In both environments, each task corresponds to a random direction in the XY-plane, and the agent (Walker/ HalfCheetach) learns to run in that direction as far as possible. The reward is equal to average velocity minus control costs. We again use MAML as the meta-learning algorithm, and the base RL algorithm is vanilla policy gradient (VPG) (Sutton et al., 1999). Following (Rothfuss et al., 2018), the policy network has 64×64 FC layers with tanh as the activation function, while the critic is a linear state-value function whose parameters are obtained by minimizing least-squares.

Figure 4 shows the convergence of the accumulated reward with number of iterations. As can be seen, SIMOL consistently outperforms MAML. Moreover, the convergence of MAML is less stable, which

[6]The CNN4 consists of four 3×3 convolution networks with batch normalization, 2×2 max-pooling and a ReLU activation layer.

Table 3: 5-way classification accuracies on miniImageNet (with 95% confidence interval). Results of Reptile, FOMAML, and Meta-MinibatchProx are from (Zhou et al., 2019), IMAML from (Deleu et al., 2022), and TSA-MAML from (Zhou et al., 2021). Results not reported in the original papers are denoted "-". The best results are in bold. The * denotes that the result are statistically significant compared with the corresponding backbone algorithm with significance level 0.1 using matched-pair t-test.

		overall		worst-10%	
		1-shot	5-shot	1-shot	5-shot
(MAML)	min average loss	49.24 ± 0.78	62.13 ± 0.72	13.33 ± 1.07	41.71 ± 1.02
	mini-batch MGDA	46.08 ± 0.78	60.15 ± 0.41	10.60 ± 1.33	39.67 ± 0.55
	mini-batch CAGrad	44.67 ± 0.75	60.05 ± 0.67	11.33 ± 1.12	40.01 ± 0.88
	SIMOL	**50.62*** $\pm$ **1.39**	**65.83*** $\pm$ **0.86**	**14.99*** $\pm$ **1.72**	**44.81***$\pm$ **0.58**
(MAML variants)	Reptile	47.07 ± 0.26	62.74 ± 0.37	-	-
	FOMAML	45.53 ± 1.58	61.02 ± 1.12	-	-
	IMAML	49.30 ± 1.88	59.77 ± 0.41	-	-
	Meta-MP	48.51 ± 0.92	64.15 ± 0.92	-	-
	TS-MAML	48.44 ± 0.91	65.52 ± 0.68	-	-
	MTL	49.87 ± 0.41	65.81 ± 0.33	13.64 ± 1.45	43.42 ± 0.47
(PN)	Standard	48.25 ± 0.95	65.29 ± 0.48	13.10 ± 1.32	45.03 ± 0.69
	SIMOL	**50.45***$\pm$ **0.93**	**66.67*** $\pm$ **0.47**	**15.00*** $\pm$ **1.21**	**46.27*** $\pm$ **0.70**

Table 4: 5-way classification accuracies on tieredImageNet (with 95% confidence interval).

		overall		worst-10%	
		1-shot	5-shot	1-shot	5-shot
(MAML)	min average loss	50.58 ± 1.44	69.33 ± 0.74	11.60 ± 1.96	46.95 ± 1.14
	mini-batch MGDA	22.92 ± 1.04	53.41 ± 0.74	7.12 ± 2.11	32.79 ± 0.88
	mini-batch CAGrad	49.04 ± 0.93	65.43 ± 0.73	11.40 ± 1.97	42.63 ± 0.95
	SIMOL	**51.42*** $\pm$ **1.50**	**70.13*** $\pm$ **0.74**	**12.00*** $\pm$ **1.95**	**47.51*** $\pm$ **1.46**
(MAML variants)	Reptile	49.12 ± 0.43	65.99 ± 0.42	-	-
	FOMAML	45.53 ± 1.58	61.02 ± 1.12	-	-
	IMAML	38.54 ± 1.37	60.24 ± 0.76	-	-
	Meta-MP	50.14 ± 1.37	68.30 ± 0.91	-	-
	TS-MAML	48.82 ± 0.88	67.82 ± 0.72	-	-
	MTL	51.02 ± 0.46	66.47 ± 0.39	13.60 ± 1.59	49.45 ± 0.58
(PN)	Standard	47.63 ± 0.93	68.92 ± 0.52	14.40 ± 1.25	46.67 ± 0.78
	SIMOL	**50.09*** $\pm$ **0.96**	**71.51*** $\pm$ **0.79**	**15.10*** $\pm$ **1.20**	**49.85*** $\pm$ **0.79**

is also reported by (Rothfuss et al., 2018). On the other hand, as SIMOL focuses more on the poorly performing tasks, its convergence is smoother and more stable.

6 Conclusion

In this paper, we propose to avoid the compromising phenomenon in meta-learning by reformulating it as a multi-objective optimization (MOO) problem, in which each task is an objective. However, current gradient-based MOO solvers cannot scale to a large number of objectives. With the use of improvement function and mini-batch, we propose a scalable gradient-based solver with theoretical guarantees to Pareto-optimality. Empirical studies on few-shot regression, few-shot classification, and reinforcement learning demonstrate that the proposed method is efficient, and has good generalization in terms of both overall performance and performance on the poorly-performing tasks.

REFERENCES

Sébastien MR Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas Zarkias. learn2learn: A library for meta-learning research. *arXiv preprint arXiv:2008.12284*, 2020.

Liam Collins, Aryan Mokhtari, and Sanjay Shakkottai. Task-robust model-agnostic meta-learning. *NeurIPS*, 33:18860–18871, 2020.

Tristan Deleu, David Kanaa, Leo Feng, Giancarlo Kerg, Yoshua Bengio, Guillaume Lajoie, and Pierre-Luc Bacon. Continuous-time meta-learning with forward mode differentiation. *arXiv preprint arXiv:2203.01443*, 2022.

Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

Michael Emmerich and André H Deutz. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing*, 17(3):585–609, 2018.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *AISTATS*, pp. 1082–1092. PMLR, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pp. 1126–1135, 2017.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *ICML*, pp. 1920–1930. PMLR, 2019.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, pp. 1568–1577. PMLR, 2018.

Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*, 2020.

Timothy M Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

Johannes Jahn. Scalarization in multi objective optimization. In *Mathematics of Multi Objective Optimization*, pp. 45–88. Springer, 1985.

Kaiyi Ji, Junjie Yang, and Yingbin Liang. Bilevel optimization: Convergence analysis and enhanced design. In *International Conference on Machine Learning*, pp. 4882–4892. PMLR, 2021.

Chia Hsiang Kao, Wei-Chen Chiu, and Pin-Yu Chen. Maml is a noisy contrastive learner in classification. In *ICLR*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, pp. 10657–10665, 2019.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Tianyi Lin, Chi Jin, and Michael Jordan. On gradient descent ascent for nonconvex-concave minimax problems. In *ICML*, pp. 6083–6093. PMLR, 2020.

Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. *NeurIPS*, 32, 2019.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *NeurIPS*, 34, 2021a.

Liyang Liu, Yi Li, Zhanghui Kuang, J Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Towards impartial multi-task learning. In *ICLR*, 2021b.

Marko M Mäkelä, Napsu Karmitsa, and Outi Wilppu. Proximal bundle method for nonsmooth and nonconvex multiobjective optimization. In *Mathematical modeling and optimization of complex structures*, pp. 191–204. Springer, 2016.

R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

K Miettinen and MM Mäkelä. Interactive bundle-based method for nondifferentiable multiobjeective optimization: NIMBUS. *Optimization*, 34(3):231–246, 1995.

Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 2012.

O Montonen, N Karmitsa, and MM Mäkelä. Multiple subgradient descent bundle method for convex nonsmooth multiobjective optimization. *Optimization*, 67(1):139–158, 2018.

Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Multi-task learning as a bargaining game. In *International Conference on Machine Learning*. PMLR, 2022.

Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2(3):4, 2018.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. *NeurIPS*, 32, 2019.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *ICLR*, 2016.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. Promp: Proximal meta-policy search. *arXiv preprint arXiv:1810.06784*, 2018.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *NeurIPS*, 31, 2018.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. *NeurIPS*, 32, 2019.

Stephen Simons. Minimax theorems and their proofs. In *Minimax and applications*, pp. 1–23. Springer, 1995.

Satinder Singh, Michael J Kearns, and Yishay Mansour. Nash convergence of gradient dynamics in general-sum games. In *UAI*, pp. 541–548. Citeseer, 2000.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *NeurIPS*, 30, 2017.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *NeurIPS*, 12, 1999.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *NeurIPS*, 29, 2016.

Haoxiang Wang, Han Zhao, and Bo Li. Bridging multi-task learning and meta-learning: Towards efficient training and effective adaptation. In *ICML*, pp. 10991–11002. PMLR, 2021.

Feiyang Ye, Baijiong Lin, Zhixiong Yue, Pengxin Guo, Qiao Xiao, and Yu Zhang. Multi-objective meta learning. *NeurIPS*, 34, 2021.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *NeurIPS*, 33:5824–5836, 2020.

Pan Zhou, Xiaotong Yuan, Huan Xu, Shuicheng Yan, and Jiashi Feng. Efficient meta learning via minibatch proximal update. *NeurIPS*, 32, 2019.

Pan Zhou, Yingtian Zou, Xiao-Tong Yuan, Jiashi Feng, Caiming Xiong, and Steven Hoi. Task similarity aware meta learning: Theory-inspired improvement on maml. In *UAI*, pp. 23–33. PMLR, 2021.

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *ICML*, pp. 7693–7702. PMLR, 2019.

A TOY EXAMPLE AND PSEUDO CODE

A.1 TOY EXAMPLE

The definition of f_1 and f_2 are:

$$f_1(x) = c_1(x)l_1(x) + c_2(x)g_1(x) \text{ and } f_2(x) = c_1(x)l_2(x) + c_2(x)g_2(x),$$

where

$$\begin{aligned}
l_1(x) &= \log\left(\max\left(\left|0.5\left(-x_1-7\right) - \tanh\left(-x_2\right)\right|, 0.000005\right)\right) + 6,\\
l_2(x) &= \log\left(\max\left(\left|0.5\left(-x_1+3\right) - \tanh\left(-x_2\right) + 2\right|, 0.000005\right)\right) + 6,\\
g_1(x) &= \left(\left(-x_1+7\right)^2 + 0.1 * \left(-x_2-8\right)^2\right)/10 - 20,\\
g_2(x) &= \left(\left(-x_1-7\right)^2 + 0.1 * \left(-x_2-8\right)^2\right)/10 - 20,\\
c_1(x) &= \max\left(\tanh\left(0.5x_2\right), 0\right) \text{ and } c_2(x) = \max\left(\tanh\left(-0.5x_2\right), 0\right).
\end{aligned}$$

For mini-batch MGDA, and SIMOL the probability to sample task 1 is $2/3$ and $1/3$ for task 2. The learning rates for MGDA and mini-batch MGDA are 0.01. The learning rate for SIMOL meta learner is 0.01, while for the reweight network is 0.1.

A.2 PSEUDO CODE

The code in blue is the major differences between standard methods and SIMOL based methods.

Algorithm 2: SIMOL for MAML. The major difference between original method and our modification is marked in blue.

Input: $\mathcal{T}$, and total epoch S. B is the batch size, β and β' are learning rates for w and θ. α is the learning rates for inner loop.

for *epoch* $s = 1, 2, 3, \ldots, S$ **do**
  Sample tasks $1, 2, \ldots B$;
  for *Every task* τ **do**
    Receive $r_{\theta^s}(\tau)$;
    Compute adapted parameters with gradient descent $w_\tau^*(w) = w^s - \alpha \nabla_w \mathcal{L}_\tau(w^s)$;
  $w^{s+1} = w^s - \beta^s \frac{1}{B} \sum_{\tau=1}^{B} r_{\theta^s}(\tau) \nabla_w \mathcal{L}_\tau(w_\tau^*(w))$;
  $\theta^{s+1} = \theta^s + \beta'^s \tilde{K}(\theta, B)$;

Algorithm 3: SIMOL for PN. The major difference between original method and our modification is marked in blue.

for *epoch* $s = 1, 2, 3, \ldots, S$ **do**
  Sample tasks $1, 2, \ldots B$;
  for *Every task* τ **do**
    for *Every class* c **do**
      Select the support S_τ^c and query set Q_τ;
      Compute prototype $c_\tau = \frac{1}{N_C} \sum_{(\mathbf{x}_i, y_i) \in S_\tau^c} f_\theta(\mathbf{x}_i)$;
    Calculate $w_\tau^*(w) = \frac{1}{|Q_\tau| N_C} \sum_{x \in Q_\tau} \frac{\exp(-\|f_w(x) - c_k\|^2)}{\sum_{k'} \exp(-\|f_w(x) - c_{k'}\|^2)}$;
    Receive $r_{\theta^s}(\tau)$;
  $w^{s+1} = w^s - \beta^s \frac{1}{B} \sum_{\tau=1}^{B} r_{\theta^s}(\tau) \nabla_w \mathcal{L}_\tau(w_\tau^*(w))$;
  $\theta^{s+1} = \theta^s + \beta'^s \tilde{K}(\theta, B)$;

B OMITTED PROOFS

B.1 PROOFS IN SEC. 3

Proof. (of Lemma 3.1) Observe that

$$\max_{\tau=1\ldots,m}\{\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\}=\sum p(\tau)\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))$$
$$s.t., p(\tau)=\begin{cases}1 & \tau=\arg\max_\tau\ \mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w')),\\ 0 & otherwise.\end{cases}$$

We firstly show that

$$\max_{\pi(\tau)}\ \mathbb{E}_{\pi(\tau)}\left[\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\right]\ -\ \sum p(\tau)\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\geq 0.$$

This can be done by setting $\pi(\tau)=p(\tau)$.

We then show that

$$\max_{\pi(\tau)}\ \mathbb{E}_{\pi(\tau)}\left[\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\right]\ -\ \sum p(\tau)\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\leq 0.$$

This is established since

$$\begin{aligned}
&\max_{\pi(\tau)}\ \mathbb{E}_{\tau\sim\pi(\tau)}\left[\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\right]\ -\ \sum p(\tau)\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\\
&\leq\sum[\pi(\tau)-p(\tau)]\left[\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\right]\\
&=[\pi(\tau')-1]\left[\mathcal{L}_{\tau'}(w_{\tau'}^*(w))-\mathcal{L}_{\tau'}(w_{\tau'}^*(w'))\right]+\sum_{\tau\in\{1:m\}/\tau*}[\pi(\tau)]\left[\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\right]\\
&\overset{(a)}{\leq}[\pi(\tau')-1]\left[\mathcal{L}_{\tau'}(w_{\tau'}^*(w))-\mathcal{L}_{\tau'}(w_{\tau'}^*(w'))\right]+\sum_{\tau\in\{1:m\}/\tau'}[\pi(\tau)]\left[\mathcal{L}_{\tau'}(w_{\tau'}^*(w))-\mathcal{L}_{\tau'}(w_{\tau'}^*(w'))\right]\\
&\leq\pi(\tau')\left[\mathcal{L}_{\tau'}(w_{\tau'}^*(w))-\mathcal{L}_{\tau'}(w_{\tau'}^*(w'))\right]+\sum_{\tau\in\{1:m\}/\tau'}[\pi(\tau)]\left[\mathcal{L}_{\tau'}(w_{\tau'}^*(w))-\mathcal{L}_{\tau'}(w_{\tau'}^*(w'))\right]\\
&-1*\left[\mathcal{L}_{\tau'}(w_{\tau'}^*(w))-\mathcal{L}_{\tau'}(w_{\tau'}^*(w'))\right]\\
&=1*\left[\mathcal{L}_{\tau'}(w_{\tau'}^*(w))-\mathcal{L}_{\tau'}(w_{\tau'}^*(w'))\right]-1*\left[\mathcal{L}_{\tau'}(w_{\tau'}^*(w))-\mathcal{L}_{\tau'}(w_{\tau'}^*(w'))\right]\\
&=0
\end{aligned}$$

where $\tau'=\arg\max_{\tau=1\ldots,m}\{\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\}$, and (a) is due to the fact that $\left[\mathcal{L}_{\tau'}(w_{\tau'}^*(w))-\mathcal{L}_{\tau'}(w_{\tau'}^*(w'))\right]\geq\left[\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\right],\forall\tau$ based on the property of τ'.

Therefore, both $\left[\mathcal{L}_{\tau'}(w_{\tau'}^*(w))-\mathcal{L}_{\tau'}(w_{\tau'}^*(w'))\right]\geq\left[\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\right]$ and $\left[\mathcal{L}_{\tau'}(w_{\tau'}^*(w))-\mathcal{L}_{\tau'}(w_{\tau'}^*(w'))\right]\leq\left[\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\right]$ are established.

Thus,

$$\max_{\tau=1\ldots,m}\{\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\}=\max_{\pi(\tau)}\ \mathbb{E}_{\tau\sim\pi(\tau)}\left[\mathcal{L}_\tau(w_\tau^*(w))-\mathcal{L}_\tau(w_\tau^*(w'))\right]. \tag{18}$$

□

Proof. (of Lemma 3.2) According to the assumptions,

$$\begin{aligned}
&\arg\min_d\mathbb{E}_{U(\tau)}[r_\theta(\tau)[\mathcal{L}_\tau\left(w_\tau^*\left(w^s+d\right)\right)-\left(\mathcal{L}_\tau\left(w_\tau^*\left(w^s\right)\right)\right]+\tfrac{\lambda'}{2}||d||^2\\
&=\ \arg\min_d\ \mathbb{E}_{U(\tau)}\left[r_\theta(\tau)\left[\nabla_w\mathcal{L}_\tau\left(w_\tau^*\left(w^s\right)\right)^\top d\right]\right]+\tfrac{\lambda'}{2}||d||^2
\end{aligned}$$

Taking the derivatives w.r.t d

$$\begin{aligned}
&\nabla_d\,\mathbb{E}_{U(\tau)}\left[r_\theta(\tau)\left[\nabla_w\mathcal{L}_\tau\left(w_\tau^*\left(w^s\right)\right)^\top d\right]\right]+\tfrac{\lambda'}{2}||d||^2\\
&=\mathbb{E}_{U(\tau)}\left[r_\theta(\tau)\left[\nabla_w\mathcal{L}_\tau\left(w_\tau^*\left(w^s\right)\right)\right]\right]+\lambda' d
\end{aligned}$$

Setting the above equation to zero $\mathbb{E}_{U(\tau)}\left[r_\theta(\tau)\nabla_w\mathcal{L}_{D_\tau}\left(w_\tau^*\left(w^s\right)\right)\right]+d\ =\ 0$ and we find the optimum:

$$d^*=\frac{-1}{\lambda'}\mathbb{E}_{U(\tau)}\left[r_\theta(\tau)\left[\nabla_w\mathcal{L}_\tau\left(w_\tau^*\left(w^s\right)\right)\right]\right]\ .$$

Putting d^* back to $\arg\min_d \ \mathbb{E}_{U(\tau)}\left[r_\theta(\tau)\left[\nabla_w \mathcal{L}_\tau\left(w_\tau^*\left(w^s\right)\right)^\top d\right]\right] + \frac{\lambda'}{2}||d||^2$, we have:

$$
\begin{aligned}
&\mathbb{E}_{\tau\sim\pi} r_\theta(\tau)\left[\mathcal{L}_\tau(w_\tau^*(w^s+d)) - \mathcal{L}_\tau(w_\tau^*(w^s))\right] + \frac{\lambda'}{2}\|d\|^2 \\
&= \left[\mathbb{E}_{\tau\sim U} r_\theta(\tau)\nabla_w \mathcal{L}_\tau(w_\tau^*(w))|_{w=w^s}^\top d\right] + \frac{\lambda'}{2}\| - \frac{1}{\lambda'}\mathbb{E}_{\tau\sim\pi}[\nabla_w \mathcal{L}_\tau(w_\tau^*(w)|_{w=w^s}]||^2 \\
&= \langle \mathbb{E}_{\tau\sim U} r_\theta(\tau)\nabla_w \mathcal{L}_\tau(w_\tau^*(w))|_{w=w^s}^\top, d\rangle \\
&+ \frac{1}{2\lambda'}\|\mathbb{E}_{\tau\sim\pi}[\nabla_w \mathcal{L}_\tau(w_\tau^*(w)|_{w=w^s}]||^2 \\
&= -\frac{1}{\lambda'}\langle \mathbb{E}_{\tau\sim U} r_\theta(\tau)\nabla_w \mathcal{L}_\tau(w_\tau^*(w))|_{w=w^s}^\top, \mathbb{E}_{\tau\sim U} r_\theta(\tau)\nabla_w \mathcal{L}_\tau(w_\tau^*(w))|_{w=w^s}^\top\rangle \\
&+ \frac{1}{2\lambda'}\|\mathbb{E}_{\tau\sim\pi}[\nabla_w \mathcal{L}_\tau(w_\tau^*(w)|_{w=w^s}]||^2 \\
&= \frac{-1}{\lambda'}\|\mathbb{E}_{\tau\sim\pi}[\nabla_w \mathcal{L}_\tau(w_\tau^*(w)|_{w=w^s}]||^2.
\end{aligned}
$$

□

The following two lemmas are not hard to obtain, and we put them here for completeness.

Lemma B.1. *When $\mathcal{L}_\tau\left(w_\tau^*\left(w^s+d\right)\right) \approx \mathcal{L}_\tau(w_\tau^*\left(w^s\right) \ + \nabla_w \mathcal{L}_\tau(w_\tau^*\left(w^s\right))^\top d$ Eq. (10) is convex w.r.t d and concave. w.r.t $r_\theta(\tau)$.*

Proof. Putting $\mathcal{L}_\tau\left(w_\tau^*\left(w^s+d\right)\right) \approx \mathcal{L}_\tau(w_\tau^*\left(w^s\right) \ + \nabla_w \mathcal{L}_\tau(w_\tau^*\left(w^s\right))^\top d$ into Eq. (10), we have:

$$
\begin{aligned}
&\min_d \max_{r(\tau)} \ \mathbb{E}_{U(\tau)}[r(\tau)[\mathcal{L}_\tau(w_\tau^*(w^s+d)) - (\mathcal{L}_\tau(w_\tau^*(w^s))] \\
&+ \frac{\lambda'}{2}||d||^2 - \frac{\lambda''}{2}(\mathbb{E}_{U(\tau)}[r(\tau)] \ - 1)^2], \\
&= \min_d \max_{r(\tau)} \ \mathbb{E}_{U(\tau)}[r(\tau)[\mathcal{L}_\tau(w_\tau^*\left(w^s\right)) \ + \nabla_w \mathcal{L}_\tau(w_\tau^*\left(w^s\right))^\top d - (\mathcal{L}_\tau(w_\tau^*(w^s))] \\
&+ \frac{\lambda'}{2}||d||^2 - \frac{\lambda''}{2}(\mathbb{E}_{U(\tau)}[r(\tau)] \ - 1)^2],
\end{aligned}
$$

Regarding d, for the $\nabla_w \mathcal{L}_\tau(w_\tau^*\left(w^s\right))^\top d + \frac{\lambda'}{2}||d||^2$ since $\nabla_w \mathcal{L}_\tau(w_\tau^*\left(w^s\right))^\top d$ is both convex and concave, and $\frac{\lambda'}{2}||d||^2$ is convex. Then the sum of two convex function $\nabla_w \mathcal{L}_\tau(w_\tau^*\left(w^s\right))^\top d + \frac{\lambda'}{2}||d||^2$ is convex. Thus, Eq. (10) is convex regarding d.

Similarly, regarding $r_\theta(\tau)$, since $-\frac{\lambda''}{2}(\mathbb{E}_{U(\tau)}[r_\theta(\tau)] \ - 1)^2$ is concave. Therefore, Eq. (10) is also concave regarding $r_\theta(\tau)$.

□

Lemma B.2. *When $\mathcal{L}_\tau\left(w_\tau^*\left(w^s+d\right)\right) \approx \mathcal{L}_\tau(w_\tau^*\left(w^s\right) \ + \nabla_w \mathcal{L}_\tau(w_\tau^*\left(w^s\right))^\top d$,*

$$
\begin{aligned}
&\min_d \max_{r_\theta(\tau)} \ \mathbb{E}_{U(\tau)}[r_\theta(\tau)[\mathcal{L}_\tau(w_\tau^*(w^s+d)) - \mathcal{L}_\tau(w_\tau^*(w^s))] \\
&+\tfrac{\lambda'}{2}||d||^2 - \tfrac{\lambda''}{2}(\mathbb{E}_p[r_\theta(\tau)] \ - 1)^2], \\
&= \max_{r_\theta(\tau)} \min_d \ \mathbb{E}_{U(\tau)}[r_\theta(\tau)[\mathcal{L}_\tau(w_\tau^*(w^s+d)) - \mathcal{L}_\tau(w_\tau^*(w^s))] \\
&+\tfrac{\lambda'}{2}||d||^2 - \tfrac{\lambda''}{2}(\mathbb{E}_p[r_\theta(\tau)] \ - 1)^2].
\end{aligned} \tag{19}
$$

Proof. It is worth noting that it is established according to the minimax theoremm(Simons, 1995).

And the remaining is to show that the above equation is Eq. (10) is convex w.r.t d and concave. w.r.t $r_\theta(\tau)$, which have been already proven in Lemma B.1. □

The following we are going to proof the Prop. 3.3.

Lemma B.3. *If we construct a set $\mathcal{B}$ such that every element $B \in \mathcal{B}$ is a set consists of a unique selection of k ($m > k > 1$) tasks out of m (the total number of tasks) tasks without replacement, then we have:*

$$\left[\frac{1}{m}\sum_{\tau=1}^{m} f(\tau)\right]^2 = C_1[\sum_{B\in\mathcal{B}}\left[\sum_{\tau\in B} f(\tau)\right]^2 - C_2\sum_{B\in\mathcal{B}}\sum_{\tau\in B}[f(\tau)]^2.$$

where $C_1 = \frac{k^2}{\binom{m-2}{k-2}m^2}$, *and* $C_2 = \left[\binom{m-1}{k-1} - \frac{1}{2}\binom{m-2}{k-2}\right] / \binom{m-1}{k-1}m^2\binom{m-2}{k-2}$, *g is an n-dimension vector.*

Proof. Note that

$$\begin{aligned}
&\left[\frac{1}{m}\sum_{\tau=1}^{m} f(\tau)\right]^2 \overset{(a)}{=} \sum_{\tau'}\sum_{\tau} 1(\tau,\tau')f(\tau)f(\tau') \\
&\overset{(b)}{=} \frac{1}{\binom{n-2}{B-2}}[\sum_{B\in\mathcal{B}}\sum_{\tau'\in B}\sum_{\tau\in B} 1(\tau,\tau')f(\tau)f(\tau') - \left[\binom{n-1}{B-1} - \frac{1}{2}\binom{n-2}{B-2}\right]\sum_{\tau\in\mathcal{T}} f(\tau)^2 \\
&= \frac{1}{\binom{n-2}{B-2}}[\sum_{B\in\mathcal{B}}\sum_{\tau'\in B}\sum_{\tau\in B} 1(\tau,\tau')f(\tau)f(\tau') - \frac{\left[\binom{n-1}{B-1} - \frac{1}{2}\binom{n-2}{B-2}\right]}{\binom{n-1}{B-1}}\sum_{B\in\mathcal{B}}\sum_{\tau\in B} f(\tau)^2] \\
&= \frac{1}{\binom{n-2}{B-2}m^2}[\sum_{B\in\mathcal{B}}[\sum_{\tau\in B} f(\tau)]^2 - \frac{\left[\binom{n-1}{B-1} - \frac{1}{2}\binom{n-2}{B-2}\right]}{\binom{n-1}{B-1}m^2}\sum_{B\in\mathcal{B}}\sum_{\tau\in B} f(\tau)^2] \\
&= C_1[\sum_{B\in\mathcal{B}}\left[\sum_{\tau\in B} f(\tau)\right]^2 - C_2\sum_{B\in\mathcal{B}}\sum_{\tau\in B}[f(\tau)]^2
\end{aligned}$$

(a) is due to the polynomial theorem and $\mathbb{1}(\tau,\tau') = \begin{cases} 2 & \tau \neq \tau' \\ 1 & \text{otherwise} \end{cases}$.

(b) is due to the fact that every task τ has occurred exactly $\binom{n-1}{B-1}$ times, and every tuple (τ,τ') has occurred exactly $\binom{n-2}{B-2}$ times. □

Lemma B.4. *If we construct a set $\mathcal{B}$ such that every element $B \in \mathcal{B}$ is a set consists of a unique selection of k ($m > k > 1$) tasks out of m (the total number of tasks) tasks without replacement, then we have:*

$$||\frac{1}{m}\sum_{\tau=1}^{m} g(\tau)||^2 = C_1\sum_{B\in\mathcal{B}}||\sum_{\tau\in B} g(\tau)||^2 - C_2\sum_{B\in\mathcal{B}}\sum_{\tau\in B}||g(\tau)||^2.$$

where $C_1 = \frac{k^2}{\binom{m-2}{k-2}m^2}$, *and* $C_2 = \left[\binom{m-1}{k-1} - \frac{1}{2}\binom{m-2}{k-2}\right] / \binom{m-1}{k-1}m^2\binom{m-2}{k-2}$.

Proof. This can be done by observing the fact that $||g(\tau)||^2 = \sum_i g_i(\tau)^2$. The remaining follows the Lemma B.3.

□

Proposition B.5. *Assume that $\tilde{K}_{\mathcal{T}}(\theta) = K(\theta)$. Set $\frac{\lambda'}{C_1|\mathcal{B}|} = \hat{\lambda}'$, $\lambda'' C_1|\mathcal{B}|k^2 = \hat{\lambda}''$, we have:*

$$(K(\theta) - \frac{1}{|\mathcal{B}|}\sum_{B}\tilde{K}_B(\theta))^2 \leq \frac{C_2 k|\mathcal{B}|G_1}{\lambda'} + C_2 k|\mathcal{B}|\lambda'' G_2,$$

where $C_1 := \frac{k^2}{\binom{m-2}{k-2}m^2}$, *and* $C_2 := \left[\binom{m-1}{k-1} - \frac{1}{2}\binom{m-2}{k-2}\right] / \binom{m-1}{k-1} m^2 \binom{m-2}{k-2}$,$G_1 := \frac{1}{k|\mathcal{B}|}\sum_{B\in\mathcal{B}}\sum_{\tau\in B}||r_\theta(\tau)\nabla_w \mathcal{L}_\tau(w^*_\tau(w))|_{w=w^s}||^2$, $G_2 := \frac{1}{k|\mathcal{B}|}\sum_{B\in\mathcal{B}}\sum_{\tau\in B}[r_\theta(\tau)-1]^2$.

When $k \ll m$, *we have* $(K(\theta) - \frac{1}{|\mathcal{B}|}\sum_B \tilde{K}_B(\theta))^2 \leq \frac{G_1}{k\lambda'} + \frac{G_2}{k}\lambda''$.

Proof. (of Proposition 3.3) Invoking lemmas B.3 and B.4, we have:

$$
\begin{aligned}
&\tilde{K}_{\mathcal{T}}(\theta) - \frac{1}{|\mathcal{B}|}\sum_B \tilde{K}_B(\theta) \\
&= \sum_{B\in\mathcal{B}}\left[-\frac{3C_1}{2\lambda'} + \frac{3}{\widetilde{2\lambda}'|\mathcal{B}|}\right]||\tilde{d}^*||^2 - \sum_{B\in\mathcal{B}}\left[\lambda'' C_1 - \frac{\hat{\lambda}''}{|\mathcal{B}|k^2}\right]\left(\sum_{\tau\in B}(r_\theta(\tau)-1)\right)^2 \\
&- C_2\left[\sum_{B\in\mathcal{B}}\sum_{\tau\in B}\left(-\frac{1}{\lambda'}||r_\theta(\tau)\nabla_w\mathcal{L}_\tau(w^*_\tau(w))|_{w=w^s}||^2 - \lambda''[r_\theta(\tau)-1]^2\right)\right] \\
&= \left[\sum_{B\in\mathcal{B}}\sum_{\tau\in B}\left(-C_2\frac{1}{\lambda'}||r_\theta(\tau)\nabla_w\mathcal{L}_\tau(w^*_\tau(w))|_{w=w^s}||^2 - C_2\lambda''[r_\theta(\tau)-1]^2\right)\right]
\end{aligned}
$$

where $\tilde{d}^* := \frac{1}{|B|}\sum_{B\in\mathcal{B}} r_\theta(\tau)\nabla_w\mathcal{L}_\tau(w^*_\tau(w))|_{w=w^s}$. Then,

$$
|\tilde{K}_{\mathcal{T}}(\theta) - \frac{1}{|\mathcal{B}|}\sum_B \tilde{K}_B(\theta)||^2 \leq \tag{20}
$$

$$
||\left[\left(-\frac{C_2 k|\mathcal{B}|}{\lambda'}\frac{1}{k|\mathcal{B}|}\sum_{B\in\mathcal{B}}\sum_{\tau\in B}||r_\theta(\tau)\nabla_w\mathcal{L}_\tau(w^*_\tau(w))|_{w=w^s}||^2 + C_2 k|\mathcal{B}|\lambda''\frac{1}{k|\mathcal{B}|}\sum_{B\in\mathcal{B}}\sum_{\tau\in B}[r_\theta(\tau)-1]^2\right)\right]||^2 \tag{21}
$$

$$
\leq \frac{C_2 k|\mathcal{B}|G_1}{\lambda'} + C_2 k|\mathcal{B}|\lambda'' G_2 \tag{22}
$$

By noting that $\tilde{K}_{\mathcal{T}}(\theta)$ with $K(\theta)$, we obtain our first claim.

Regarding the second claim, note that when m is large,

$$
\begin{aligned}
C_2 k|\mathcal{B}| &= \frac{k\left[\binom{m-1}{k-1} - \frac{1}{2}\binom{m-2}{k-2}\right]\binom{m}{k}}{\binom{m-1}{k-1}m^2\binom{m-2}{k-2}} \\
&\approx \frac{k\binom{m-1}{k-1}\binom{m}{k}}{\binom{m-1}{k-1}m^2\binom{m-2}{k-2}} = \frac{k\binom{m}{k}}{m^2\binom{m-2}{k-2}} \\
&= \frac{k\binom{m-1}{k-1}\frac{m}{k}}{m^2\binom{m-2}{k-2}} = \frac{k\binom{m-2}{k-2}\frac{m}{k}\frac{m-1}{k-1}}{m^2\binom{m-2}{k-2}} \\
&= \frac{m-1}{(k-1)m} \approx \frac{1}{k-1}.
\end{aligned}
$$

The first approximation is due to the fact that $\binom{m-1}{k-1} \gg \frac{1}{2}\binom{m-2}{k-2}$. The third and forth equations are due to the property of combinatorics ($\binom{m}{k} = \frac{m}{k}\binom{m-1}{k-1}$) Putting $\frac{1}{k-1}$ back to the Eq. (20), we obtain the results. □

B.2 PROOF OF LEMMA 4.1

Proof. (Lemma 4.1) By directly calculating, we have:

$$\begin{aligned}
&\nabla_\theta R(\theta, w) = \nabla_\theta \Delta(\theta) - \nabla_\theta \tfrac{\lambda''}{2}(\mathbb{E}_{\tau\sim U} r_\theta(\tau) - 1)^2 \\
&= -\tfrac{1}{\lambda'}\mathbb{E}_{\tau\sim U}\nabla_\theta r_\theta(\tau) \perp [(\mathcal{L}_\tau\left(w_\tau^*(w) + d^*\right) - \mathcal{L}_\tau\left(w_\tau^*(w)\right))] \perp [\mathbb{E}_{\tau\sim U} r_\theta(\tau)\left[\mathcal{L}_\tau\left(w_\tau^*(w) + d^*\right) - \mathcal{L}_\tau\left(w_\tau^*(w)\right)\right]] \\
&= \nabla_\theta ||\mathbb{E}_{\tau\sim U}\nabla_\theta r_\theta(\tau)\nabla_w \mathcal{L}_\tau(w_\tau^*(w)|_{w=w^s}||^2 - \nabla_\theta \tfrac{\lambda''}{2}(\mathbb{E}_{\tau\sim U} r_\theta(\tau) - 1)^2 \\
&= \nabla_\theta K(\theta)
\end{aligned}$$

Also notice that:

$$\begin{aligned}
\nabla_w R(\theta, w) = \mathbb{E}_{\tau\sim U}\left[\perp\left(r_\theta(\tau)\right)\left[\nabla_w \mathcal{L}_\tau(w_\tau^*(w)\right]\right] + \nabla_w \Delta(\theta) - \nabla_w \frac{\lambda''}{2}(\mathbb{E}_{\tau\sim U} r_\theta(\tau) - 1)^2], \\
= \mathbb{E}_{\tau\sim U} r_\theta(\tau)\left[\nabla_w L_\tau(w_\tau^*(w))\right],
\end{aligned}$$

Therefore,

$$\nabla_w R(\theta, w)|_{w\,=w_s} \;=\; -d^*$$

□

B.3 PROOF OF THEOREM 4.2

In (Montonen et al., 2018), the authors mention that when $s \to \infty$, x^s and H is differential w.r.t x^s, is the Pareto Stationary point without detailed proof. In the following, we provide a way to proof that.

We firstly define

Definition B.6. *(Weak Global Pareto Optimality) (Mäkelä et al., 2016) A solution x^* of problem (6) is weakly global Pareto optimal if there does not exist another x such that $f_\tau(x^*) \leq f_\tau(x)$ for $\tau \in \{1, \ldots, m\}$.*

Theorem B.7. *(Theorem 5 in (Miettinen & Mäkelä, 1995)) For a multi-objective learning $\min_w[f_1(w), f_2(w), \ldots, f_m(w)]$ and its corresponding improvement function $H\left(w, w^*\right)$. A necessary condition for $w^* \in \mathbb{R}^n$ to be a global weak Pareto optimum is that $w^* = \arg\min_{w\in\mathbb{R}^n} H\left(w, w^*\right)$. And if further $f_i(w),\ \forall i$ is convex, then it is a sufficient condition.*

Here we introduce two observations aid our following analysis:

1) By replacing $f_\tau = \mathcal{L}_\tau \circ w_\tau^*(w), \forall\tau$, the above theorem can be directly applied to MAML setting.

2) f in our setting might be neural network, and thus the convex condition is hard to preserve. Here, we give the local version of the above theorem:

Theorem B.8. $w^* := \arg\min_w H\left(w, w^*\right)$ *is also the Pareto stationary point.*

Proof. According to Theorem 2 in (Miettinen & Mäkelä, 1995), we have

$\partial_w H\left(w, w^*\right) \subset \operatorname{conv} \bigcup_i \partial_w\left(f_i(w) - f_i\left(w^*\right)\right)$, where ∂ is reloaded as sub-gradient, and conv is the convex set.

Note that $0 \in \partial_w H\left(w, w^*\right)$ due to the fact that the convex set of the unions of sub-gradients are still sub-gradients, and in extreme point, the sub-gradient is zero.

Then, we have $\sum_i w_i \partial_w\left(f_i(w) - f_i\left(w^*\right)\right) = 0,\ \sum w_i = 1$, based on the property of definition of convex set.

By reducing the above term, we have $\sum_i w_i \partial\left(f_i(w) - f_i\left(w^*\right)\right) = \sum_i w_i \partial_w f_i(w)\ = 0$. which is exactly the definition of Pareto stationary point. □

Theorems B.7 and B.8 discuss the global and local relationship between Pareto optimality and the improvement function respectively.

We firstly show the convergence properties of Alg. 1:

Lemma B.9. *By running Alg. 1, for each epoch s, when* $\lambda'' = \begin{cases} 0, & \mathbb{E}_p[r_\theta(\tau)] = 1 \\ \infty, & otherwise \end{cases}$ *, we have:*

$$H(w^s + d, w^s) \leq H(w^s, w^s),$$

If equality holds, it is a stationary point.

Proof. By Lemma B.2, the solutions of

$$\max_\theta \ \min_d \ \mathbb{E}_{U(\tau)}[r_\theta(\tau)\left[\mathcal{L}_\tau(w_\tau^*\left(w^s + d\right)) - \mathcal{L}_\tau(w_\tau^*\left(w^s\right))\right] + \frac{\lambda'}{2}||d||^2 - \frac{\lambda''}{2}(\mathbb{E}_p[r_\theta(\tau)] \ - 1)^2],$$

is equivalent to

$$\min_d \max_\theta \ \ \mathbb{E}_{U(\tau)}[r_\theta(\tau)\left[\mathcal{L}_\tau(w_\tau^*\left(w^s + d\right)) - \mathcal{L}_\tau(w_\tau^*\left(w^s\right))\right] + \frac{\lambda'}{2}||d||^2 - \frac{\lambda''}{2}(\mathbb{E}_p[r_\theta(\tau)] \ - 1)^2],$$

Solving θ, the above minimax problem degenerates to

$$\min_d \max_{\tau=1\ldots,m} \left[\mathcal{L}_\tau(w_\tau^*\left(w^s + d\right)) - \mathcal{L}_\tau(w_\tau^*\left(w^s\right))\right] + \frac{\lambda'}{2}||d||^2$$

Note the above equation is $\min_d H(w^s + d, w^s) + \frac{\lambda'}{2}||d||^2$, where d is the steepest descent direction. Then, due to the property of steepest descent direction, we always have $H(w^s + d, w^s) \leq H(w^s, w^s)$ and the equality holds iff the stationary point $||d||^2 = 0$. □

Lemma B.10. *Alg. 1 converges to Pareto stationary point. If further* $\mathcal{L}_\tau(w_\tau^*(w))$ *is convex, it is the global weak Pareto optilimity.*

Proof. According to Lemma B.9, the sequence $\{H(w^s + d, w^s, w_{1:m}^{s+1}, w_{1:m}^s)\}_s$ is deceasing.

Also obviously, $H(w^s + d, w^s, w_{1:m}^{s+1}, w_{1:m}^s)$ has lower bound.

Thus, leveraging the Monotone convergence theorem, Alg. 1 converges.

According to Theorem B.8, it is the Pareto stationary. Together with Theorem B.7, it is the global Pareto optimality for convex case. □

Lemma B.11. *i) If* $\mathcal{L}_\tau(w_\tau^*(w + d))$ *is L-smooth and* μ'*-convex w.r.t* w *and d, and* $r_\theta(\tau)$ *is* μ*-concave,* $r_\theta(\tau)$ *and* $\mathcal{L}_\tau(w_\tau^*(w^{s+1}))$ *are bounded (i.e.,* $r_\theta(\tau) \leq C_{\max}, \mathcal{L}_\tau(w_\tau^*(w^{s+1})) \leq C'_{\max}$*), then* $R(\theta, w)$ *is* $L + C_{\max}\mu'$ *convex w.r.t* w *and L smooth w.r.t* w, θ *and* $C'_{\max}\mu$ *concave w.r.t.* θ*.*

ii) And if $\mathcal{L}_\tau(w_\tau^*(w + d))$ *is L-smooth w.r.t* w *and d, and* $r_\theta(\tau)$ *is* μ*-concave, then* $\tilde{R}(\theta, w, B)$ *is* L *smooth w.r.t* w, θ *and* $C'_{\max}\mu$ *concave w.r.t.* θ*.*

Proof. For the i) claim.

We firstly prove that $\perp [f(x)] \perp [g(x)]$ is 0-smooth and convex w.r.t. x for differential function f, g.

Note that

$$||\nabla_x \perp [f(x)] \perp [g(x)] \ - \ \nabla_{x'} \perp [f(x')] \perp [g(x')] \ ||^2 \ = \ 0$$

due to the property of stop gradient operator.

$$\begin{aligned} &\perp [f(x)] \perp [g(x)] - \perp [f(x')] \perp [g(x')] \\ &\leq 0 \\ &= -\nabla_{x'}(\perp [f(x')] \perp [g(x')])(\perp [f(x)] \perp [g(x)] \end{aligned}$$

Regarding the convexity, by using the property of 0-smoothness,

implies

$$\perp [f(x')] \perp [g(x')] - \perp [f(x)] \perp [g(x)] \geq \nabla_{x'}(\perp [f(x')] \perp [g(x')])(\perp [f(x)] \perp [g(x)])$$

Thus, it is convex due to the definition of convexity.

Therefore, we conclude that $\perp [f(x)] \perp [g(x)]$ is also 0-smooth and convex w.r.t. x.

A direct application of the above conclusion is that $\perp [f(x)]$ is 0-smooth and convex w.r.t. x by regarding $g(x) \equiv 1$.

Then, using the above conclusion, we have $\perp [(L_\tau(w_\tau^*(w) + d) - \mathcal{L}_\tau(w_\tau^*(w))]$, $\perp [\mathbb{E}_{\tau\sim U} r_\theta(\tau) [\mathcal{L}_\tau (w_\tau^*(w) + d^*) - \mathcal{L}_\tau (w_\tau^*(w))]]$, $\perp [(\mathcal{L}_{D_T} (\boldsymbol{w}_T^*(\boldsymbol{w}) + d^*) - \mathcal{L}_{D_T} (\boldsymbol{w}_T^*(\boldsymbol{w}))]]$ $\perp \mathbb{E}_{T\sim U} r_\theta(T) [\mathcal{L}_{D_T} (\boldsymbol{w}_T^*(\boldsymbol{w}) + d^*) - \mathcal{L}_{D_T} (\boldsymbol{w}_T^*(\boldsymbol{w}))]$ are also 0-smooth and convex.

Therefore $r_\theta(T) \perp [(\mathcal{L}_{D_T} (\boldsymbol{w}_T^*(\boldsymbol{w}) + d^*) - \mathcal{L}_{D_T} (\boldsymbol{w}_T^*(\boldsymbol{w}))]] \perp [\mathbb{E}_{T\sim U} r_\theta(T) [\mathcal{L}_{D_T} (\boldsymbol{w}_T^*(\boldsymbol{w}) + d^*) - \mathcal{L}_{D_T} (\boldsymbol{w}_T^*(\boldsymbol{w}))]]$ is convex and L-smooth (as $L \geq 0$).

Also, we have $\perp (r_\theta(\tau))[\mathcal{L}_\tau(w_\tau^*(w)]]$ is $C_{\max}\mu'$ convex and L-smooth due to our assumption.

Then $\mathbb{E}_{\tau\sim U} [\perp (r_\theta(\tau))[\mathcal{L}_\tau(w_\tau^*(w)]] + \Delta(\theta) - \frac{\lambda''}{2}(\mathbb{E}_{\tau\sim U} r_\theta(\tau) - 1)^2$ is $C_{\max}\mu'$ convex and $2L$-smooth since the adding of convex functions implies $C_{\max}\mu'$ convex and the adding of smooth functions implies L-smooth, and the take the average does not affect the results.

Then, $R(\theta, w)$ is $C_{\max}\mu'$ convex and $2L$-smooth.

Regarding r_θ, since $-\frac{\lambda'}{2}\nabla_\theta(\mathbb{E}_p[r_{\theta^s}(\tau)] - 1)^2$ is concave, $\Delta(\theta)$ is $2\mu C'_{\max}$ concave. Then $\Delta(\theta) - \frac{\lambda''}{2}(\frac{1}{B}\sum_\tau [r_\theta(\tau)] - 1)^2$ is also $C'_{\max}\mu$-concave. Thus $R(\theta, w)$ is $C'_{\max}\mu$-concave.

The proof of our ii) claim can be similarly conducted by following the above proof. □

Theorem B.12. *Assume that 1) $\mathcal{L}_\tau(w_\tau^*(\cdot))$ is L-smooth and $r_\theta(\cdot)$ is μ-strongly concave. 2) The domain of θ is a convex and bounded set with diameter $D > 0$, 3) $\mathbb{E}_{B\sim U(\mathcal{B})}[\nabla_\theta \tilde{R}(\theta, w; B) - \nabla_\theta R(\theta, w)] = 0$, and $\mathbb{E}_{B\sim U(\mathcal{B})}\|\nabla_\theta \tilde{R}(\theta, w, B) - \nabla_\theta R(\theta, w)\|^2 \leq \sigma^2$. Assume the first-order approximation in Eq. (11), and take $\beta = \Theta\left(1/D^2\sigma^2(L^2 + \sigma^2)\right)$, $\beta' = \Theta(1/L\sigma^2)$. Algorithm 1 converges to an ϵ-Pareto stationary point of (6) with a rate of $O(1/\epsilon^8)$. If $\mathcal{L}_\tau(w_\tau^*(w))$ is also μ'-convex w.r.t. w and $\epsilon = 0$, the 0-Pareto stationary point is also global Pareto optimal.*

Proof. Recall Lemma B.11 on the properties of convex and smooth for $\tilde{R}$. Combining it with the assumptions mentioned above, and invoking Theorem 4.9 in (Lin et al., 2020), we can obtain the iteration bound of $O(\frac{1}{\epsilon^8})$ when $B = 1$.

When $B > 1$, noting that in Lemma A.2 in (Lin et al., 2020), we have:

$$\mathbb{E}\left[\left\|\frac{1}{B}\sum_{i=1}^{B}\tilde{R}(\theta, w, \tau, \tau')\right\|^2\right] \leq \|\nabla_w R(\theta, w)\|^2 + \frac{\sigma^2}{B}$$

Therefore, denoting $\sigma^{2'} = \frac{\sigma^2}{B}$ and reusing Theorem 4.9 in (Lin et al., 2020) shows iteration bound of $O(\frac{1}{\epsilon^8})$.

The remaining part that the fixed point of w is the global Pareto optilimity (resp. Pareto stationary point). It can be directly testified by invoking Lemma B.10 that reveals the fixed point of Alg. 1 is the globally Pareto optimality (resp. Pareto stationary point).

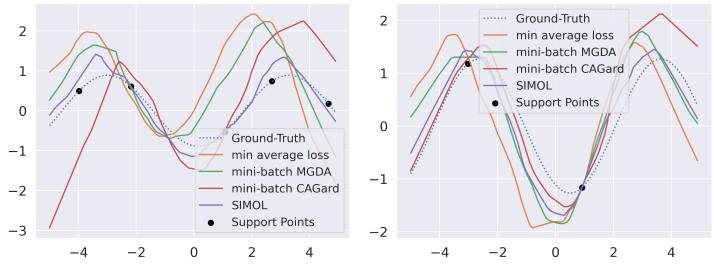
□

Finally, we show that using SIMOL to MAML can guarantee convergence based on the analysis above:

Corollary B.12.1. *Consider the MAML base learner update $w_\tau^*(w) = w - \alpha\nabla_w\mathcal{L}_\tau(w)$. assume that $\nabla_w\mathcal{L}_\tau(w)$ is Hessian-Lipschitz, bounded and Lipschitz-continuous, w is bounded. then, Algorithm 1 converges to an ϵ-Pareto stationary point of (6) with a rate of $O(1/\epsilon^8)$.*

Proof. We firstly show that if $\nabla_w\mathcal{L}_\tau(w)$ is Hessian-Lipschitz, bounded and Lipschitz continuous, and $||w||$ is bounded, then $w_\tau^*(w)$ is also Hessian-Lipschitz, bounded and Lipschitz continuous. Note that $||\nabla_w^2 w_\tau^*(w) - \nabla_{w'}^2 w_\tau^*(w')|| = \alpha||\nabla_w^2(\nabla_w\mathcal{L}_\tau(w)) - \nabla_{w'}^2(\nabla_{w'}\mathcal{L}_\tau(w'))||$,

$||\nabla_w w_\tau^*(w) - \nabla_{w'} w_\tau^*(w')|| = \alpha||\nabla_w(\nabla_w\mathcal{L}_\tau(w)) - \nabla_{w'}(\nabla_{w'}\mathcal{L}_\tau(w'))||$



(a) 5-shot. (b) 2-shot.

Figure 3: Learned functions on two example meta-testing tasks.

$$||w_\tau^*(w)|| = ||w - \alpha \nabla_w \mathcal{L}_\tau(w)|| = ||w|| + \alpha ||\nabla_w \mathcal{L}_\tau(w)||.$$

Then, it is easy to find that $w_\tau^*(w)$ is also Hessian-Lipschitz, bounded and Lipschitz continuous.

Applying Lemma 3 (Collins et al., 2020), we immediately get $\mathcal{L}_\tau(w_\tau^*(w))$ is C-smooth, where C is a positive number. Then setting $C = L$ and using theorem 4.2, we immediately get the desire result.

□

C Generalization between Training distribution to testing distributions.

Proposition C.1. *If the distribution over training tasks $P(\mathcal{T}_{train})$ and the distribution over testing tasks $P(\mathcal{T}_{test})$ are the same, when $E_{\tau \sim P(\mathcal{T}_{train})} \mathcal{L}_\tau(w_\tau^*(w^*)) \leq \epsilon$, where w^* is the parameter learnt from SIMOL, we have $E_{\tau \sim P(\mathcal{T}_{test})} \mathcal{L}_\tau(w_\tau^*(w^*)) \leq \epsilon$.*

Similarly, when $E_{\tau \sim P(\mathcal{T}_{train})} ||\nabla_w \mathcal{L}_\tau(w_\tau^*(w^*))||^2 \leq \epsilon$, we have $E_{\tau \sim P(\mathcal{T}_{test})} ||\nabla_w \mathcal{L}_\tau(w_\tau^*(w^*))||^2 \leq \epsilon$.

Proof. As the distribution over training tasks $P(\mathcal{T}_{train})$ and the distribution over testing tasks $P(\mathcal{T}_{test})$ are the same, $E_{\tau \sim P(\mathcal{T}_{train})} \mathcal{L}_\tau(w_\tau^*(w^*)) = E_{\tau \sim P(\mathcal{T}_{test})} \mathcal{L}_\tau(w_\tau^*(w^*))$. Therefore, it is obviously that $E_{\tau \sim P(\mathcal{T}_{test})} \mathcal{L}_\tau(w_\tau^*(w^*)) \leq \epsilon$.

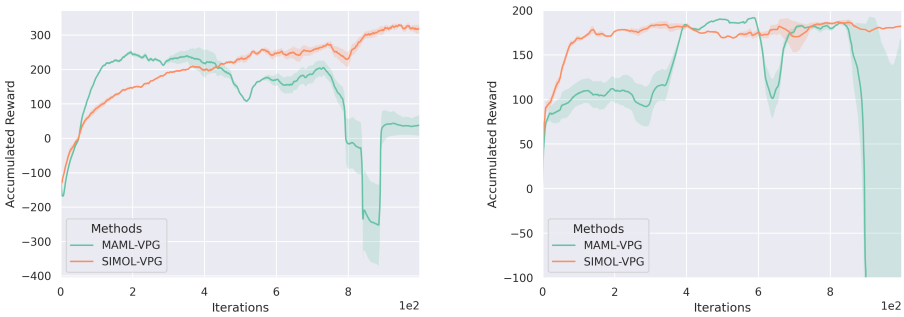
Similarly, $E_{\tau \sim P(\mathcal{T}_{train})} ||\nabla_w \mathcal{L}_\tau(w_\tau^*(w^*))||^2 = E_{\tau \sim P(\mathcal{T}_{test})} ||\nabla_w \mathcal{L}_\tau(w_\tau^*(w^*))||^2$, we also have $E_{\tau \sim P(\mathcal{T}_{test})} ||\nabla_w \mathcal{L}_\tau(w_\tau^*(w^*))||^2 \leq \epsilon$. □

D Extra Experiments

D.1 More detailed setup

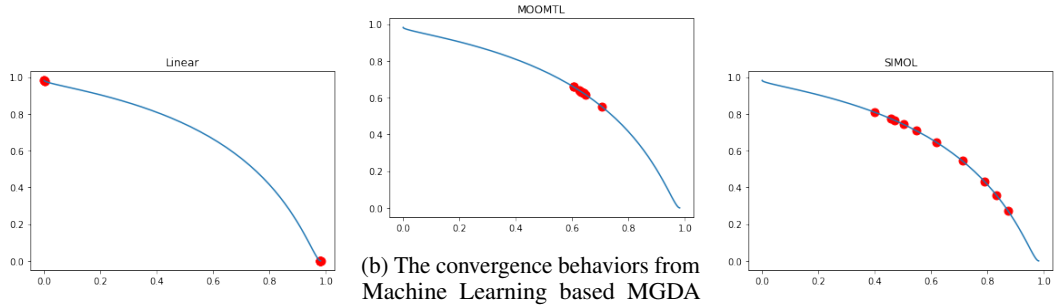
Regarding the classification scenarios. For the re-weighting network, we also use CNN4 as backbone. Differently, we feed every instance in a task to CNN4 and then two layer FC with size 128×48 and 48×6 to generate the task embedding, and then, they are stacked together and be feed into a layer with size $(6 * \#\text{way} * \#\text{shot}) * 1$ (e.g., for 1-way-1-shot task, the layer size is $(6 * 5) * 1$). The learning rate is 0.08 for SIMOL and 0.0008 for SIMOL+PN.

Regarding how to select the hyper-parameters, the weighting for regularizers $\hat{\lambda}'$, $\hat{\lambda}''$, and learning rate β' are selected by sweeping among $\{0.001, 0.01, 0.1, 1\}$, $\{0.001, 0.01, 0.1, 1\}$, and $\{0.01, 0.03, 0.1, 0.3, 1\}$ on 2-shot regression tasks, and choose the best combinations (i.e., $\hat{\lambda}' = 0.01$, $\hat{\lambda} = 1$, $\beta' = 0.03$). We directly use $\hat{\lambda}' = 0.01$, $\hat{\lambda} = 1$ to other environments. Regarding the learning rate β' in classification environments, we select the best among $\{0.0003, 0.0008, 0.01, 0.03, 0.08, 0.1\}$ in 1 shot miniImageNet (0.08 for SIMOL and 0.0008 for SIMOL+PN) and extend it other classification environments.



(a) HalfCheetah-Dir (b) Walker-Dir

Figure 4: Returns for SIMOL-VPG and MAML-VPG. Results are reported over 3 trials.



(a) The convergence behaviors from Random Linear Scalarization.

(b) The convergence behaviors from Machine Learning based MGDA (MOOMTL (Sener & Koltun, 2018)).

(c) The convergence behaviors from SIMOL.

Figure 5: convergence behaviors in popular task.

D.2 Toy Example

Our extra toy examples show the convergence behavior of ours when finding a Pareto front.

We construct the synthetic as:

$$\min_{w} \quad \frac{1}{2}\mathcal{L}_1\left(w_\tau^*(w)\right) + \frac{1}{2}\mathcal{L}_2\left(w_\tau^*(w)\right), \tag{23}$$

$$\text{s.t.} \quad \mathcal{L}_1\left(w_\tau^*(w)\right) := 1 - \exp\left(-\sum_{i=1}^{10}\left(w_i - \frac{1}{\sqrt{10}}\right)^2\right) \tag{24}$$

$$\mathcal{L}_2\left(w_\tau^*(w)\right) := 2.5 - 2.5\exp\left(-\sum_{i=1}^{10}\left(w_i + \frac{1}{\sqrt{10}}\right)^2\right) \tag{25}$$

The task 1 and task 2 has the same probability to be sampled for SIMOL. The initial point is uniformly sampled from $[-0.5, 0.5]$ following (Lin et al., 2019). GD is used as the Optimizer.

The baselines are Linear Scalarization (with equal weights for each objective) (Jahn, 1985), Machine Learning based MGDA (MOOMTL (Sener & Koltun, 2018)), as well as ParetoMTL (Lin et al., 2019).

Results: As can be seen, ours have a more disperse points than other baselines.

D.3 Regression Tasks.

Regarding the effectiveness of mini-batch size, as shown in Fig. 6, larger mini-batch size does improve the performance (1.43 for SIMOL-4 v.s. 1.24 for SIMOL-16). However, the gain is deceasing with

Table 5: Performance on different number of layers in 2-shot regression. SIMOL-X layers means the total number of layers is X.

	Overall MSE	Worst 10 % MSE
SIMOL-2 layers	1.24 ± 0.08	5.66 ± 0.33
SIMOL-3 layers	1.36 ± 0.09	6.01 ± 0.38
SIMOL-4 layers	1.42 ± 0.08	6.22 ± 0.34

Table 6: Performance on different minibatch sizes in 2-shot regression. SIMOL-X means the minibatch size is X.

	Overall MSE	Worst 10 % MSE
SIMOL-4	1.43 ± 0.08	6.09 ± 0.31
SIMOL-8	1.26 ± 0.08	5.67 ± 0.33
SIMOL-16	1.24 ± 0.08	5.66 ± 0.33
SIMOL-64	1.23 ± 0.09	5.66 ± 0.46

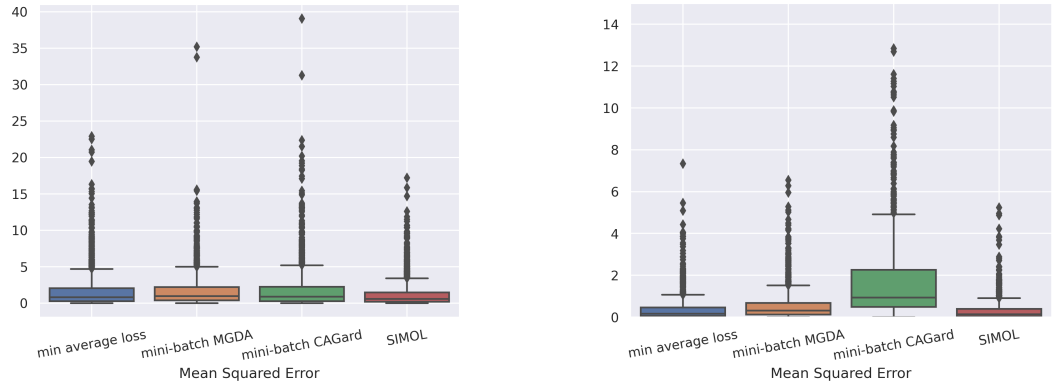
the increasing of the mini-batch size (1.23 for SIMOL-64 v.s. 1.24 for SIMOL-16). Therefore, it is not necessary to maintain a large batch size.

The ablations on different layers' structure is shown in 5. Results reveal that more layers with the same training time perform worse results.

The extra results on PCGrad is shown in Tab. 9, showing that PCGrad does not suit regression tasks with large-scale tasks.

D.4 Extra Classification Results

The best 10% performance is shown in Tabs. 7 and 8. Results show that ours have better performance over other baselines, but the improvement is smaller than that of the worse 10%.



(a) box plot shot-2.

(b) box plot shot-5.

Figure 6: MSE box-plots (Figures 6b & 6a)

Table 7: Performance of best 10 % MSE on few shot classification task on mini-ImageNet. The best results are in bold.

	1-shot	5-shot
MAML-min average loss	85.00 ± 0.99	81.99 ± 0.83
MAML-mini-batch MGDA	83.80 ± 0.89	79.76 ± 0.44
MAML-mini-batch CAGrad	82.00 ± 0.68	78.71 ± 0.73
PNL-min average loss	84.10 ± 1.13	83.39 ± 0.56
MAML-SIMOL	$\mathbf{85.80 \pm 1.38}$	$\mathbf{85.13 \pm 0.47}$
PN-SIMOL	85.50 ± 1.24	84.49 ± 0.52

Table 8: Performance of best 10 % MSE on few shot classification task on Tiered-ImageNet. The best results are in bold.

	1-shot	5-shot
MAML-min average loss	88.00 ± 1.95	88.99 ± 0.82
MAML-mini-batch MGDA	53.40 ± 2.11	74.15 ± 0.91
MAML-mini-batch CAGrad	89.00 ± 1.98	85.75 ± 0.73
PN-min average loss	84.40 ± 1.15	87.65 ± 0.48
MAML-SIMOL	$\mathbf{89.60 \pm 1.99}$	89.03 ± 0.71
PN-SIMOL	85.60 ± 1.31	$\mathbf{90.65 \pm 0.42}$

Table 9: MSE (with 95% confidence interval) for few-shot regression with PCGrad. The best results are in bold.

	overall		worst-10%	
MAML	5-shot	2-shot	5-shot	2-shot
min average loss	0.43 ± 0.11	1.70 ± 0.11	2.13 ± 0.21	7.75 ± 0.48
mini-batch MGDA	0.60 ± 0.02	1.73 ± 0.10	2.62 ± 0.10	7.04 ± 0.51
mini-batch CAGrad	1.90 ± 0.24	1.82 ± 0.44	8.18 ± 0.63	8.23 ± 2.33
mini-batch PCGrad	1.89 ± 0.12	1.80 ± 0.11	6.64 ± 0.47	7.73 ± 0.54
SIMOL	$\mathbf{0.34 \pm 0.04}$	$\mathbf{1.24 \pm 0.08}$	$\mathbf{1.69 \pm 0.18}$	$\mathbf{5.66 \pm 0.33}$