Distributed Meta-Learning with Networked Agents

Mert Kayaalp, Stefan Vlaski, and Ali H. Sayed
Adaptive Systems Laboratory, EPFL
{mert.kayaalp, stefan.vlaski, ali.sayed}@epfl.ch

***Abstract*—Meta-learning aims to improve efficiency of learning new tasks by exploiting the inductive biases obtained from related tasks. Previous works consider centralized or federated architectures that rely on central processors, whereas, in this paper, we propose a decentralized meta-learning scheme where the data and the computations are distributed across a network of agents. We provide convergence results for non-convex environments and illustrate the theoretical findings with experiments.**

***Index Terms*—meta-learning, learning to learn, multi-agent optimization, networked agents, distributed learning**

I. Introduction and Related Work

We consider a network of K agents. Each agent k has access to data arising from a collection of tasks $\mathcal{T}_k$. The probability distribution over $\mathcal{T}_k$ is denoted by π_k, i.e., the probability of drawing task t from $\mathcal{T}_k$ is $\pi_k(t)$. In standard machine learning, for any particular task $t \in \mathcal{T}_k$, agent k trains a separate model by solving:

$$\min_{w\in\mathbb{R}^M} J_k^{(t)}(w) \triangleq \min_{w\in\mathbb{R}^M} \mathbb{E}_{x_k^{(t)}} Q_k^{(t)}\Big(w; \boldsymbol{x}_k^{(t)}\Big) \tag{1}$$

where w denotes the model parameter, $\boldsymbol{x}_k^{(t)}$ denotes the random data corresponding to task t observed at agent k (all random variables are denoted in bold), the loss $Q_k^{(t)}(w;\boldsymbol{x}_k^{(t)})$ represents the penalty incurred by w under the random data $\boldsymbol{x}_k^{(t)}$, and $J_k^{(t)}(w)$ denotes the stochastic risk.

Learning models *de novo* might require vast amounts of data and computational resources. In order to address this problem, meta-learning, or learning to learn, utilizes prior knowledge induced from related tasks in order to facilitate and accelerate learning of new tasks.

One particular method is the model-agnostic meta-learning (MAML) algorithm [2], which explicitly optimizes for a *launch model* that can adapt to new tasks with one or small number of gradient steps. For one step, finding the launch model corresponds to minimizing the modified risk function:

$$\min_{w\in\mathbb{R}^M} \overline{J_k}(w) \triangleq \min_{w\in\mathbb{R}^M} \mathbb{E}_{t\sim\pi_k} J_k^{(\boldsymbol{t})}\Big(w - \alpha\nabla J_k^{(\boldsymbol{t})}(w)\Big) \tag{2}$$

where $\alpha > 0$ is the step size parameter. The gradient vector corresponding to this risk is given by :

$$\nabla\overline{J_k}(w) \triangleq \mathbb{E}_{t\sim\pi_k}\Big[\Big(I - \alpha\nabla^2 J_k^{(\boldsymbol{t})}(w)\Big)\nabla J_k^{(\boldsymbol{t})}\Big(w - \alpha\nabla J_k^{(\boldsymbol{t})}(w)\Big)\Big] \tag{3}$$

The proofs of the results presented in this paper are omitted due to space limitations. They can be found in [1].

In practice, π_k and the distribution of $\boldsymbol{x}_k^{(t)}$ are not known. Therefore, it is common to approximate (3) by a stochastic gradient vector:

$$\nabla\widehat{\overline{Q_k}}(w) \triangleq \frac{1}{|\mathcal{S}_k|}\sum_{\boldsymbol{t}\in\boldsymbol{\mathcal{S}}_k}\Big[\Big(I - \alpha\nabla^2 Q_k^{(\boldsymbol{t})}(w;\boldsymbol{\mathcal{X}}_{in}^{(t)})\Big)\times \nabla Q_k^{(\boldsymbol{t})}\Big(w - \alpha\nabla Q_k^{(\boldsymbol{t})}(w;\boldsymbol{\mathcal{X}}_{in}^{(t)})\ ;\ \boldsymbol{\mathcal{X}}_o^{(t)}\Big)\Big] \tag{4}$$

where $\boldsymbol{\mathcal{X}}_{in}^{(t)}$, $\boldsymbol{\mathcal{X}}_o^{(t)}$ are two random batches of data, $\boldsymbol{\mathcal{S}}_k \subset \mathcal{T}_k$ is a random batch of tasks, and $|\mathcal{S}_k|$ is the number of selected tasks. Note that we use the notation $Q(w;\mathcal{X})$ to abbreviate $\frac{1}{N}\sum_{n=1}^N Q(w;x_n)$ where $\{x_n\}_{n=1}^N$ are the elements of $\mathcal{X}$. We assume that all elements of $\boldsymbol{\mathcal{X}}_{in}^{(t)}$, $\boldsymbol{\mathcal{X}}_o^{(t)}$ are independently sampled from the distribution of $\boldsymbol{x}_k^{(t)}$ and all tasks $\boldsymbol{t} \in \boldsymbol{\mathcal{S}}_k$ are independently sampled from $\mathcal{T}_k$.

Several works building up on MAML in single-agent settings include the following: [3] proposes the Reptile algorithm, which does not require Hessian calculations, [4] analyzes the effectiveness of MAML and concludes feature reuse is a dominant factor, and [5] studies the convergence of MAML. In particular, [6] argues that minimizing the modified risk (2) requires more samples than doing joint training ($\alpha = 0$ in (2)). Thus, distributed systems in meta-learning are more feasible, considering the number of data and computational needs.

Another line of work examines meta-learning in federated settings [7]–[10]. These works depend upon a central server that gathers and averages the models of the agents. However, a failure at this center can break down the system. Moreover, increasing the number of agents adds to the communication burden and can severely limit the capacity. In order to have a system that is robust to failures and scalable, we propose a decentralized algorithm where the information transfer between networked agents occurs via peer-to-peer communications. This can be beneficial for privacy issues when the agents do not prefer to share model parameters or data with the server.

Our algorithm combines MAML and diffusion learning for decentralized optimization [11], which is particularly successful in adapting to drifts in the data with constant step-sizes [12]. References [13] and [14] analyze the convergence of the diffusion algorithm in non-convex environments under stochastic gradients, which are conditions that are also applicable to our work. However, in our analysis, we work with the MAML risk (2), which contains a gradient inside the arguments. Hence, we need to make proper adjustments to the analysis.

Contributions. In this paper, we propose a decentralized meta-learning algorithm in which agents collaborate with their

neighbors in order to solve an *aggregate* meta-objective. We show that the agents come to agreement at a linear rate, hence, the algorithm behaves like the centralized solution in terms of performance even though it does not possess a central processor. In addition, we demonstrate that the agreed models converge to a first-order stationary point of the aggregate objective in non-convex environments. Simulations with neural networks are provided to support the theoretical findings.

II. Diffusion-based MAML (Dif-MAML)

In a non-cooperative setting, each agent k would optimize its objective (2) separately. However, if the task collections $\{\mathcal{T}_k\}_{k=1}^K$ are related, we would expect a better performance when the agents cooperate since the effective number of the data each agent processes increases. The cooperation can be achieved by seeking a launch model that minimizes the following network objective:

$$\min_{w\in\mathbb{R}^M} \overline{J}(w) \triangleq \frac{1}{K}\sum_{k=1}^{K} \overline{J_k}(w) \tag{5}$$

Diffusion is a distributed algorithm that minimizes (5). We assume a graph with combination matrix $A = [a_{\ell k}]$, where the coefficients $\{a_{\ell k}\}$ are non-negative and add up to one:

$$\sum_{\ell=1}^{K} a_{\ell k} = 1,\ a_{\ell k} > 0 \ \text{ if agents } \ell \text{ and } k \text{ are connected}$$

In the Adapt-then-Combine variant of diffusion [11], at every iteration i, each agent k simultaneously performs the following updates:

$$\boldsymbol{\phi}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu\nabla\overline{Q_k}(\boldsymbol{w}_{k,i-1}) \tag{6a}$$

$$\boldsymbol{w}_{k,i} = \sum_{\ell=1}^{K} a_{\ell k}\boldsymbol{\phi}_{\ell,i} \tag{6b}$$

Expression (6a) is an *adaptation* step where all agents simultaneously obtain intermediate states $\boldsymbol{\phi}_{k,i}$ by a stochastic gradient update with constant outer-loop step size μ. Recall that $\nabla\overline{Q_k}(\boldsymbol{w}_{k,i-1})$ from (4) is the stochastic approximation of the exact gradient $\nabla\overline{J_k}(\boldsymbol{w}_{k,i-1})$ from (3) . Expression (6b) is a *combination* step where the agents combine their neighbors' intermediate steps to obtain updated iterates $\boldsymbol{w}_{k,i}$.

The proposed method is presented in Algorithm 1. Each agent starts with its own launch model. At every iteration, the agents sample a batch of tasks and compute the task-specific models by applying the task-specific stochastic gradients to their current launch models. Consecutively, intermediate states are found by applying the meta-stochastic gradients (4) to the launch models. A non-cooperative algorithm would use intermediate states as updated launch models and stop here without exchanging any information between the agents. In contrast, Dif-MAML has an extra step in which agents communicate with their neighbors and update their launch models based on what they receive. This diffuses information across the network and increases the number of tasks and data each agent trains its model on.

Algorithm 1 Dif-MAML

Initialize the *launch* models $\{w_{k,0}\}_{k=1}^K$
while not done **do**
  for all agents **do**
    Agent k samples a batch of $i.i.d.$ tasks $\boldsymbol{\mathcal{S}}_{k,i}$ from $\mathcal{T}_k$
    for all tasks $\boldsymbol{t} \in \boldsymbol{\mathcal{S}}_{k,i}$ **do**
      Evaluate $\nabla Q_k^{(\boldsymbol{t})}\Big(\boldsymbol{w}_{k,i-1}; \boldsymbol{\mathcal{X}}_{in,i}^{(t)}\Big)$ using a batch of $i.i.d.$ data $\boldsymbol{\mathcal{X}}_{in,i}^{(t)}$
      Set *task-specific* models $\boldsymbol{w}_{k,i}^{(t)} = \boldsymbol{w}_{k,i-1} - \alpha\nabla Q_k^{(\boldsymbol{t})}\Big(\boldsymbol{w}_{k,i-1}; \boldsymbol{\mathcal{X}}_{in,i}^{(t)}\Big)$
    end for
    Compute *intermediate states* $\boldsymbol{\phi}_{k,i} = \boldsymbol{w}_{k,i-1} - (\mu/|\mathcal{S}_{k,i}|)\sum_{\boldsymbol{t}\in\boldsymbol{\mathcal{S}}_{k,i}} \nabla Q_k^{(\boldsymbol{t})}\Big(\boldsymbol{w}_{k,i}^{(t)}; \boldsymbol{\mathcal{X}}_{o,i}^{(t)}\Big)$ using a batch of $i.i.d.$ data $\boldsymbol{\mathcal{X}}_{o,i}^{(t)}$ for each task (Check (4) to see gradient expression explicitly)
  end for
  for all agents **do**
    Update the launch models by combining the intermediate states $\boldsymbol{w}_{k,i} = \sum_{\ell=1}^K a_{\ell k}\boldsymbol{\phi}_{\ell,i}$
  end for
  $i \leftarrow i+1$
end while

III. Performance Results

In this section, we analyze the convergence of Dif-MAML in non-convex environments; proofs are omitted due to space limitations but can be found in [1].

A. Assumptions

Assumption 1 (Lipschitz gradients). *For each agent k and task $t \in \mathcal{T}_k$, the gradient $\nabla Q_k^{(t)}(\cdot;\cdot)$ is Lipschitz, namely, for any $w, u \in \mathbb{R}^M$ and $\boldsymbol{x}_k^{(t)}$ denoting a data point:*

$$\Big\|\nabla Q_k^{(t)}\Big(w; \boldsymbol{x}_k^{(t)}\Big) - \nabla Q_k^{(t)}\Big(u; \boldsymbol{x}_k^{(t)}\Big)\Big\| \le L^{\boldsymbol{x}_k^{(t)}}\|w-u\| \tag{7}$$

We assume the second-order moment of the Lipschitz constant is bounded by a data-independent constant:

$$\mathbb{E}_{x_k^{(t)}}\Big(L^{\boldsymbol{x}_k^{(t)}}\Big)^2 \le \Big(L_k^{(t)}\Big)^2 \tag{8}$$

In this paper, for simplicity, we will work with $L \triangleq \max_k \max_t L_k^{(t)}$.

Assumption 2 (Lipschitz Hessians). *For each agent k and task $t \in \mathcal{T}_k$, the Hessian $\nabla^2 Q_k^{(t)}(\cdot;\cdot)$ is Lipschitz in expectation, namely, for any $w, u \in \mathbb{R}^M$ and $\boldsymbol{x}_k^{(t)}$ denoting a data point:*

$$\mathbb{E}_{x_k^{(t)}}\Big\|\nabla^2 Q_k^{(t)}\Big(w; \boldsymbol{x}_k^{(t)}\Big) - \nabla^2 Q_k^{(t)}\Big(u; \boldsymbol{x}_k^{(t)}\Big)\Big\| \le \rho_k^{(t)}\|w-u\| \tag{9}$$

In this paper, for simplicity, we will work with $\rho \triangleq \max_k \max_t \rho_k^{(t)}$.

Assumption 3 (Bounded gradients). *For each agent k and task $t \in \mathcal{T}_k$, the gradient $\nabla Q_k^{(t)}(\cdot;\cdot)$ is bounded in expectation, namely, for any $w \in \mathbb{R}^M$ and $\boldsymbol{x}_k^{(t)}$ denoting a data point:*

$$\mathbb{E}_{x_k^{(t)}} \left\| \nabla Q_k^{(t)}\left(w; \boldsymbol{x}_k^{(t)}\right) \right\| \leq B_k^{(t)} \tag{10}$$

In this paper, for simplicity, we will work with $B \triangleq \max_k \max_t B_k^{(t)}$.

Assumption 4 (Bounded noise moments). *For each agent k and task $t \in \mathcal{T}_k$, the gradient $\nabla Q_k^{(t)}(\cdot;\cdot)$ and the Hessian $\nabla^2 Q_k^{(t)}(\cdot;\cdot)$ have bounded fourth-order central moments, namely, for any $w \in \mathbb{R}^M$:*

$$\mathbb{E}_{x_k^{(t)}} \left\| \nabla Q_k^{(t)}\left(w; \boldsymbol{x}_k^{(t)}\right) - \nabla J_k^{(t)}(w) \right\|^4 \leq \sigma_G^4 \tag{11}$$

$$\mathbb{E}_{x_k^{(t)}} \left\| \nabla^2 Q_k^{(t)}\left(w; \boldsymbol{x}_k^{(t)}\right) - \nabla^2 J_k^{(t)}(w) \right\|^4 \leq \sigma_H^4 \tag{12}$$

Defining the mean of the risk functions of the tasks in $\mathcal{T}_k$ as $J_k(w) \triangleq \mathbb{E}_{t\sim\pi_k} J_k^{(t)}(w)$, we have the following assumption on the relations between the tasks of a particular agent.

Assumption 5 (Bounded task variability). *For each agent k , the gradient $\nabla J_k^{(t)}(\cdot)$ and the Hessian $\nabla^2 J_k^{(t)}(\cdot)$ have bounded fourth-order central moments, namely, for any $w \in \mathbb{R}^M$:*

$$\mathbb{E}_{t\sim\pi_k} \left\| \nabla J_k^{(t)}(w) - \nabla J_k(w) \right\|^4 \leq \gamma_G^4 \tag{13}$$

$$\mathbb{E}_{t\sim\pi_k} \left\| \nabla^2 J_k^{(t)}(w) - \nabla^2 J_k(w) \right\|^4 \leq \gamma_H^4 \tag{14}$$

Note that we do not need to assume a relation between the tasks of different agents for our convergence results to hold. However, transfer learning between agents via diffusion would be beneficial if the agents' tasks are related. We keep this assumption implicit.

Assumption 6 (Doubly-stochastic combination matrix). *The weighted combination matrix $A = [a_{\ell k}]$ representing the graph is doubly-stochastic. This means that the matrix has non-negative elements and satisfies:*

$$A\mathbb{1} = \mathbb{1}, \quad A^{\mathsf{T}}\mathbb{1} = \mathbb{1} \tag{15}$$

The matrix A is also primitive which means that a path with positive weights can be found between any arbitrary nodes (k, ℓ), and moreover at least one $a_{kk} > 0$ for some k.

B. Adjusted MAML Objective

Because of the gradient within a gradient, the stochastic MAML gradient (4) is a *biased* estimator of (3). Alternatively, we can consider the following adjusted objective in lieu of (2):

$$\widehat{J_k}(w) \triangleq \mathbb{E}_{t\sim\pi_k} \mathbb{E}_{\mathcal{X}_{in}^{(t)}} J_k^{(t)}\left(w - \alpha \nabla Q_k^{(t)}(w; \boldsymbol{\mathcal{X}}_{in}^{(t)})\right) \tag{16}$$

The stochastic MAML gradient (4) is an *unbiased* estimator of the gradient corresponding to this adjusted objective:

$$\nabla\widehat{J_k}(w) = \mathbb{E}\nabla\overline{Q_k}(w) \tag{17}$$

	Single-task	Meta-objective	Meta-gradient
Risk function	$J_k^{(t)}$	$\overline{J_k}$	$\nabla\overline{J_k}$
Adjusted Risk	X	$\widehat{J_k}$	$\nabla\widehat{J_k}$
Stochastic Approximation	$Q_k^{(t)}$	X	$\nabla\overline{Q_k}$

TABLE I
SUMMARY OF SOME NOTATION USED IN THE PAPER.

Refer to Table 1 for a summary of some notation used in the paper. Next, we introduce a result that analyses the perturbation between the gradients of the MAML objective (2) and the adjusted objective (16). By using this, we will present our convergence results for both objectives.

Lemma 1 (Gradient perturbation bound). *Under assumptions 1,3,4, for each agent k, the disagreement between $\nabla\overline{J_k}(\cdot)$ and $\nabla\widehat{J_k}(\cdot)$ is bounded, namely, for any $w \in \mathbb{R}^M$:*

$$\left\| \nabla\overline{J_k}(w) - \nabla\widehat{J_k}(w) \right\| \leq (1+\alpha L)\frac{\alpha L \sigma_G}{\sqrt{|\mathcal{X}_{in}|}} + \frac{B\alpha\sigma_H}{\sqrt{|\mathcal{X}_{in}|}} \tag{18}$$

Lemma 1 suggests that the disagreement between the gradients of the standard MAML objective and the adjusted objective gets smaller with decreasing inner learning rate α and increasing inner batch size $|\mathcal{X}_{in}|$.

Now, we show that the gradient of the adjusted objective has bounded norm and is Lipschitz in terms of the assumptions made on the standard loss functions.

Lemma 2 (Bounded gradient of adjusted objective). *Under assumptions 1,3, for each agent k, the gradient $\nabla\widehat{J_k}(\cdot)$ of the adjusted objective is bounded, namely, for any $w \in \mathbb{R}^M$:*

$$\left\| \nabla\widehat{J_k}(w) \right\| \leq \widehat{B} \tag{19}$$

where $\widehat{B} \triangleq (1+\alpha L)B$ is a non-negative constant.

Lemma 3 (Lipschitz gradient of adjusted objective). *Under assumptions 1-3, for each agent k, the gradient $\nabla\widehat{J_k}(\cdot)$ of adjusted objective is Lipschitz, namely, for any $w, u \in \mathbb{R}^M$:*

$$\left\| \nabla\widehat{J_k}(w) - \nabla\widehat{J_k}(u) \right\| \leq \widehat{L}\|w - u\| \tag{20}$$

where $\widehat{L} \triangleq (L(1+\alpha L)^2 + \alpha\rho B)$ is a non-negative constant.

In the following lemma, we derive that the stochastic MAML gradient has bounded variance.

Lemma 4 (Gradient noise for adjusted objective). *Under assumptions 1-5, the gradient noise defined as $\nabla\overline{Q_k}(w) - \nabla\widehat{J_k}(w)$ is bounded for any $w \in \mathbb{R}^M$, namely:*

$$\mathbb{E}\left\| \nabla\overline{Q_k}(w) - \nabla\widehat{J_k}(w) \right\|^2 \leq C^2 \tag{21}$$

where $C^2 = \frac{3}{|\mathcal{S}_k|}\left(\gamma_G^2(20\gamma_G^2 + 4\gamma_G + 9) + \frac{6\sigma_G^2}{|\mathcal{X}_o|}\right) + O(\alpha)$ is a non-negative constant. See [1] for the full expression.

Lemma 4 states that the size of the MAML gradient noise decreases with task-relatedness and task batch-size, and increases with data variance and inner step-size α.

C. Convergence of Dif-MAML

We define the network centroid as $\boldsymbol{w}_{c,i} \triangleq \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{w}_{k,i}$. It is an average of the agents' parameters. First, we prove that agents will cluster around this network centroid in sufficient number of iterations.

Theorem 1 (Network disagreement). *Under assumptions 1-6, network disagreement between the centroid launch model and the launch models of each agent k is bounded after $O(\log\mu) = o(1/\mu)$ iterations, namely:*

$$\frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\|\boldsymbol{w}_{k,i}-\boldsymbol{w}_{c,i}\|^2 \leq \mu^2\frac{\lambda_2^2}{(1-\lambda_2)^2}\left(\widehat{B}^2+C^2\right) + O(\mu^3) \tag{22}$$

for

$$i \geq \frac{3\log\mu}{\log\lambda_2} + O(1) = o(1/\mu) \tag{23}$$

where λ_2 is the mixing rate of the combination matrix A, i.e., it is the spectral radius of $A^{\mathsf{T}} - \frac{1}{K}\mathbb{1}_K\mathbb{1}_K^{\mathsf{T}}$.

Theorem 1 shows that the difference between the centroid launch model and agent-specific launch models is bounded after $O(\log\mu) = o(1/\mu)$ iterations. Therefore, we use the network centroid as a deputy for all models and study the convergence to the stationary points on that in the next theorem.

Theorem 2 (Stationary points of adjusted objective). *In addition to assumptions 1-6, assume that $\widehat{J}(w)$ is bounded from below, i.e., $\widehat{J}(w) \geq \widehat{J}^o$. Then, the centroid launch model $\boldsymbol{w}_{c,i}$ will reach an $O(\mu)$-mean-square-stationary point in at most $O\left(1/\mu^2\right)$ iterations. In particular, there exists a time instant $i^\star$ such that:*

$$\mathbb{E}\left\|\nabla\widehat{J}(\boldsymbol{w}_{c,i^\star})\right\|^2 \leq 2\mu\widehat{L}C^2 + O(\mu^2) \tag{24}$$

and

$$i^\star \leq \left(\frac{2(\widehat{J}(w_0)-\widehat{J}^o)}{\widehat{L}C^2}\right)1/\mu^2 + O(1/\mu) \tag{25}$$

A similar result can be obtained for the MAML objective (2) by using Lemma 1.

Corollary 1 (Stationary points of MAML objective). *Assume that the same conditions of Theorem 2 hold. Then, the centroid launch model $\boldsymbol{w}_{c,i}$ will reach an $O(\mu)$-mean-square-stationary point, up to a constant, in at most $O\left(1/\mu^2\right)$ iterations. Namely, for time instant $i^\star$ defined in* (25)*:*

$$\mathbb{E}\left\|\nabla\overline{J}(\boldsymbol{w}_{c,i^\star})\right\|^2 \leq 4\mu\widehat{L}C^2 + O(\mu^2) + 2\left((1+\alpha L)\frac{\alpha L\sigma_G}{\sqrt{|\mathcal{X}_{in}|}} + \frac{B\alpha\sigma_H}{\sqrt{|\mathcal{X}_{in}|}}\right)^2 \tag{26}$$

Corollary 1 shows that the network centroid reaches an $O(\mu)$-mean-square-stationary point for sufficiently small inner learning rate α and for sufficiently large inner batch size $|\mathcal{X}_{in}|$, in at most $O\left(1/\mu^2\right)$ iterations. Note that as $\mu \to 0$, when opposed to the number of iterations required for convergence ($O(1/\mu^2)$), the necessary number of iterations required for network agreement ($O(\log\mu) = o(1/\mu)$) becomes negligible.

IV. Experiments

In this section, we compare Dif-MAML with the non-cooperative strategy in which agents learn separate launch models and the centralized strategy in which all data and computational power are gathered on a server. In terms of accuracy/loss performance, the centralized strategy is in fact equivalent to having a network that is fully-connected because cooperating agents would reach a consensus at first iteration. Indeed, we illustrate by simulations in regression and classification tasks that even with sparsely-connected graphs, Dif-MAML is able to match the performance of the centralized strategy and outperform the non-cooperative strategy. Our experiments and hyperparameters are based on the experiments and implementation of [2]. They are conducted with neural networks.

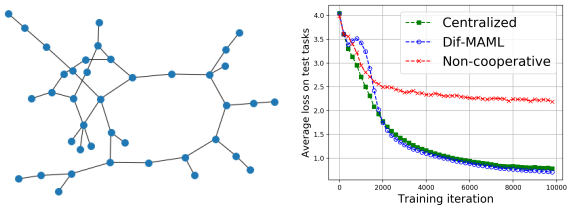
A. Regression

We consider tasks that require regressing the output of a sine wave. Agents have access to different task distributions. Namely, the amplitude interval $[0.1, 5.0]$ is evenly partitioned into $K = 40$ agents connected with the sparse graph Fig. 1a. The phases are changing between $[0, \pi]$ for each agent. The combination weights are found by the averaging rule [11], which is simply assigning the same weights to all neighbors. We use Adam [15] optimizer ($\mu = 0.001$) for the outer-loop.

During training, every 200th iteration, all agents are tested with the same validation tasks stemming from the amplitude interval $[0.1, 5.0]$ and the phase interval $[0, \pi]$. It can be seen in Fig. 1b that Dif-MAML matches the centralized strategy and outperforms the non-cooperative strategy even with a relatively sparse network topology. This results show that agents can benefit cooperation even when they have different task distributions. Furthermore, despite the fact that we consider stochastic gradient descent (SGD) in this paper, Fig. 1b supports that our analysis can be extended to other optimization methods such as Adam.

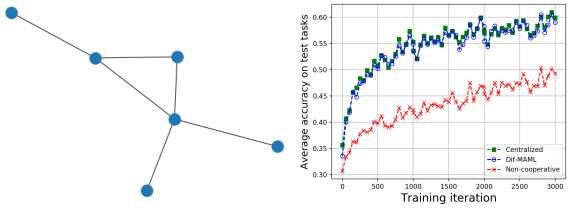
B. Classification

We use the popular few-shot learning benchmark MiniImagenet 5-way 5-shot task [16]. As opposed to the regression experiment, in classification: (i) agents sample their data from the same task distribution, (ii) they are allowed to use multiple updates for adaptation both in training and testing, (iii) outer-loop optimization is based on SGD ($\mu = 0.01$), (iv) combination weights are set with Metropolis rule [17], and (v) convolutional neural networks are used. These differences are preferred in order to show that Dif-MAML can also be successful in the listed settings.



(a) (b)

Fig. 1. Regression : (a) 40-agent sparsely-connected network (b) Validation losses during training



(a) (b)

Fig. 2. Classification: (a) 6-agent network (b) Validation accuracies during training

The graph underlying the network and the accuracies in every 50th iteration can be seen in Fig. 2. The agents are tested on the same tasks. Fig. 2b is averaged over the tasks and the agents. In accordance with the regression experiment, diffusion-based decentralized solution can match the centralized solution. Moreover, it is performing significantly better than the non-cooperative solution since effective number of data each agent processes increases due to cooperation. In fact, the match between the centralized and decentralized algorithms is an evidence for the fact that each agent is processing effectively *all data* across the network in Dif-MAML. Even though our theoretical results are based on one step adaptation as in Eq. 2, the classification results advocate that they might hold true for multiple step adaptations.

In addition, we note that in order to use Dif-MAML, or diffusion in general, with neural networks batch normalization [18] is necessary because the combination step (6b) reduces the variance of the weights due to averaging.

V. Conclusion and Future Work

In this paper, we proposed a cooperative and decentralized meta-learning algorithm with networked agents. Our theoretical analysis of the algorithm established that agents come to agreement at a linear rate and they proceed to a first-order stationary-point of an aggregate meta-learning objective. Experiments with simulated and real data confirm that the proposed *fully distributed* strategy is able to match the centralized strategy. Future work might be combining meta-learning with distributed algorithms that consider asynchronous agents [19] or agents having multiple local updates before communicating with their neighbors [20].

Acknowledgment

The authors would like to thank Y. Efe Erginbas for helpful discussions on the experiments.

References

[1] M. Kayaalp, S. Vlaski, and A. H. Sayed, "Dif-MAML: Decentralized multi-agent meta-learning," *available as arXiv:2010.02870*, 2020.

[2] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proc. International Conference on Machine Learning*, Sydney, Australia, Aug. 2017, pp. 1126–1135.

[3] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *available as arXiv: 1803.02999*, March 2018.

[4] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? Towards understanding the effectiveness of MAML," in *Proc. International Conference on Learning Representations*, 2020.

[5] A. Fallah, A. Mokhtari, and A. Ozdaglar, "On the convergence theory of gradient-based model-agnostic meta-learning algorithms," in *Proc. International Conference on Artificial Intelligence and Statistics*, Aug. 2020, vol. 108, pp. 1082–1092.

[6] K. Gao and O. Sener, "Modeling and optimization trade-off in meta-learning," in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 11154–11165.

[7] M. Khodak, M.-F. Balcan, and A. Talwalkar, "Adaptive gradient-based meta-learning methods," in *Advances in Neural Information Processing Systems 32*, pp. 5917–5928. Vancouver, Canada, Dec. 2019.

[8] Y. Jiang, J. Konecný, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *available as arXiv:1909.12488*, Sep. 2019.

[9] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," in *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 3557–3568.

[10] F. Chen, M. Luo, Z. Dong, Z. Li, and X. He, "Federated meta-learning with fast convergence and efficient communication," *available as arXiv: 1802.07876*, Feb. 2018.

[11] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. 4-5, pp. 311–801, July 2014.

[12] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks - Part I: Transient analysis," *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3487–3517, June 2015.

[13] S. Vlaski and A. H. Sayed, "Diffusion learning in non-convex environments," in *Proc. of IEEE ICASSP*, Brighton, UK, May 2019, pp. 5262–5266.

[14] S. Vlaski and A. H. Sayed, "Distributed learning in non-convex environments—Part I: Agreement at a linear rate," *IEEE Transactions on Signal Processing*, vol. 69, pp. 1242–1256, 2021.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations*, San Diego, CA, USA, May 2015.

[16] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *Proc. International Conference on Learning Representations*, Toulon, France, April 2017.

[17] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, June 1953.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. International Conference on Machine Learning*, 2015, vol. 37, p. 448–456.

[19] X. Zhao and A. H. Sayed, "Asynchronous adaptation and learning over networks – Part II: Performance analysis," *IEEE Transactions on Signal Processing*, vol. 63, no. 4, pp. 827–842, Feb 2015.

[20] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd," *Proc. Machine Learning and Systems 1*, pp. 212–229, 2019.