How Important is the Train-Validation Split in Meta-Learning?

Yu Bai [1] **Minshuo Chen** [2] **Pan Zhou** [1] **Tuo Zhao** [2] **Jason D. Lee** [3] **Sham Kakade** [4]
Huan Wang [1] **Caiming Xiong** [1]

Abstract

Meta-learning aims to perform fast adaptation on a new task through learning a "prior" from multiple existing tasks. A common practice in meta-learning is to perform a train-validation split (*train-val method*) where the prior adapts to the task on one split of the data, and the resulting predictor is evaluated on another split. Despite its prevalence, the importance of the train-validation split is not well understood either in theory or in practice, particularly in comparison to the more direct *train-train method*, which uses all the per-task data for both training and evaluation.

We provide a detailed theoretical study on whether and when the train-validation split is helpful in the linear centroid meta-learning problem. In the agnostic case, we show that the expected loss of the train-val method is minimized at the optimal prior for meta testing, and this is not the case for the train-train method in general without structural assumptions on the data. In contrast, in the realizable case where the data are generated from linear models, we show that both the train-val and train-train losses are minimized at the optimal prior in expectation. Further, perhaps surprisingly, our main result shows that the train-train method achieves a *strictly better* excess loss in this realizable case, even when the regularization parameter and split ratio are optimally tuned for both methods. Our results highlight that sample splitting may not always be preferable, especially when the data is realizable by the model. We validate our theories by experimentally showing that the train-train method can indeed outperform the train-val method, on both simulations and real meta-learning tasks.

[1]Salesforce Research [2]Georgia Tech [3]Princeton University [4]University of Washington. Correspondence to: Yu Bai <yu.bai@salesforce.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1. Introduction

Meta-learning, also known as "learning to learn", has recently emerged as a powerful paradigm for learning to adapt to unseen tasks (Schmidhuber, 1987). The high-level methodology in meta-learning is akin to how human beings learn new skills, which is typically done by relating to certain prior experience that makes the learning process easier. More concretely, meta-learning does not train one model for each individual task, but rather learns a "prior" model from multiple existing tasks so that it is able to quickly adapt to unseen new tasks. Meta-learning has been successfully applied to many real problems, including few-shot image classification (Finn et al., 2017; Snell et al., 2017), hyper-parameter optimization (Franceschi et al., 2018), low-resource machine translation (Gu et al., 2018) and short event sequence modeling (Xie et al., 2019).

A common practice in meta-learning algorithms is to perform a *sample splitting*, where the data within each task is divided into a *training split* which the prior uses to adapt to a task-specific predictor, and a *validation split* on which we evaluate the performance of the task-specific predictor (Vinyals et al., 2016; Nichol et al., 2018; Rajeswaran et al., 2019; Fallah et al., 2020; Wang et al., 2020a). For example, in a 5-way k-shot image classification task, standard meta-learning algorithms such as MAML (Finn et al., 2017) use $5k$ examples within each task as training data, and use additional examples (e.g. k images, one for each class) as validation data. This sample splitting is believed to be crucial as it matches the evaluation criterion at meta-test time, where we perform adaptation on training data from a new task but evaluate its performance on unseen data from the same task.

Despite the aforementioned importance, performing the train-validation split has a potential drawback from the data efficiency perspective — Because of the split, neither the training nor the evaluation stage is able to use all the available per-task data. In the few-shot image classification example, each task has a total of $6k$ examples available, but the train-validation split forces us to use these data separately in the two stages. Meanwhile, performing the train-validation split is also not the only option in practice: there exist algorithms such as Reptile (Nichol & Schulman, 2018) and

Meta-MinibatchProx (Zhou et al., 2019) that can instead use all the per-task data for training the task-specific predictor and also perform well empirically on benchmark tasks. These algorithms modify the loss function in the outer loop so that the training loss no longer matches the meta-test loss, but may have the advantage in terms of data efficiency for the overall problem of learning the best prior. So far it is theoretically unclear how these two approaches (with/without train-validation split) compare with each other, which motivates us to ask the following

> **Question**: Is the train-validation split *necessary* and *optimal* in meta-learning?

In this paper, we perform a detailed theoretical study on the importance of the train-validation split. We consider the linear centroid meta-learning problem (Denevi et al., 2018b), where for each task we learn a linear predictor that is close to a common centroid in the inner loop, and find the best centroid in the outer loop (see Section 2 for the detailed problem setup). We compare two meta-learning algorithms: the **train-val method** which performs the standard train-validation split, and the **train-train method** which uses all the per-task data for both training and evaluation.

We summarize our contributions as follows:

- We show that the train-validation split is necessary in the general agnostic setting (Section 3): The expected loss of the train-val method equals the meta test-time loss. In contrast, the train-train method has a different expected loss and is not minimized at the best test-time centroid in general, for which we construct a concrete counter-example.

- In the perhaps more interesting realizable setting, we show the train-validation split is not necessary: When the tasks are generated from noiseless linear models, the expected loss of both the train-val and train-train methods are minimized at the best test-time centroid (Section 4.1).

- Our main theoretical contribution shows that **the train-validation split is non-optimal** in the realizable setting: The MSE (and test loss) of the two methods concentrates sharply around $C^{\{\mathsf{tr\text{-}val},\mathsf{tr\text{-}tr}\}}/T$ when T (the number of tasks) is large, where the constants depend on the {dimension, per-task sample size, regularization parameter}. A precise comparison of constants further shows that $C^{\mathsf{tr\text{-}tr}} < C^{\mathsf{tr\text{-}val}}$ when we optimally tune the regularization parameter in both methods (Section 4.2). Thus, in the realizable setting, the train-train method performs strictly better than the train-val method, which is in stark contrast with the agnostic case. This result provides a novel insight into the effect of the train-validation split on the sample complexity of meta-learning.

- We perform meta-learning experiments on simulations and benchmark few-shot image classification tasks, showing that the train-train method consistently outperforms the train-val method (Section 5 & Appendix F). This validates our theories and presents empirical evidence that sample-splitting may not be crucial; methods that utilize the per-task data more efficiently may be preferred.

- On the technical end, our main results in Section 4 build on concentration analyses on a group of ridge-covariance matrices, as well as tools from random matrix theory in the proportional regime, which may be of broader interest. (See Section 4.3 for an overview of techniques.)

1.1. Related work

Meta-learning and representation learning theory Baxter (2000) provided the first theoretical analysis of meta-learning via covering numbers, and Maurer et al. (2016) improved the analysis via Gaussian complexity techniques. Another recent line of theoretical work analyzed gradient-based meta-learning methods (Denevi et al., 2018a; Finn et al., 2019; Khodak et al., 2019; Ji et al., 2020) and showed guarantees for convex losses by using tools from online convex optimization. Saunshi et al. (2020) proved the success of Reptile in a one-dimensional subspace setting. Wang et al. (2020c) compared the performance of train-train and train-val methods for learning the learning rate. Denevi et al. (2018b) proposed the linear centroid model studied in this paper, and provided generalization error bounds for train-val method; the bounds proved also hold for train-train method, so are not sharp enough to compare the two algorithms. Wang et al. (2020b;a) studied the convergence of gradient-based meta-learning by relating to the kernelized approximation. Arnold et al. (2019) observe that MAML adapts better with a deep model architecture both empirically and theoretically.

On the representation learning end, Du et al. (2020); Tripuraneni et al. (2020a;b) showed that ERM can successfully pool data across tasks to learn the representation. Yet the focus is on the accurate estimation of the common representation, not on the fast adaptation of the learned prior. Several recent work compares MAML versus ERM style approches (Gao & Sener, 2020; Collins et al., 2020); these comparisons couple the effect of sample splitting with other factors such as whether the algorithm uses per-task adaptation. Lastly, we remark that there are analyses for other representation learning schemes (McNamara & Balcan, 2017; Galanti et al., 2016; Alquier et al., 2016).

Empirical understandings of meta-learning Raghu et al. (2020) showed that MAML with a full finetuning inner loop mostly learns the top-layer linear classifier and does not change the representation layers much. This re-

sult partly justifies the validity of our linear centroid meta-learning problem in which the features (representations) are fixed and only a linear classifier is learned. Goldblum et al. (2020) investigated the difference of the neural representations learned by classical training (supervised learning) and meta-learning, and showed that the meta-learned representation is better for downstream adaptation and makes classes more separated. Additionally, Setlur et al. (2020); Yao et al. (2020) investigated alternative ways of choosing the support set (training split) in meta-learning.

Multi-task learning Multi-task learning also exploits structures and similarities across multiple tasks. The earliest idea dates back to Caruana (1997); Thrun & Pratt (1998); Baxter (2000), initially in connections to neural network models. They further motivated other approaches using kernel methods (Evgeniou et al., 2005; Argyriou et al., 2007) and multivariate linear regression models with structured sparsity (Liu et al., 2009; 2015). More recent advances on deep multi-task learning focus on learning shared intermediate representations across tasks (Ruder, 2017). These multi-task learning approaches usually minimize the joint empirical risk over all tasks, and the models for different tasks are enforced to share a large amount of parameters. In contrast, meta-learning only requires the models to share the same "prior", and is more flexible than multi-task learning.

2. Preliminaries

In this paper, we consider the standard meta-learning setting, in which we observe data from $T \geq 1$ supervised learning tasks, and the goal is to find a prior (or "initialization") using the combined data, such that the $(T+1)$-th new task may be solved sample-efficiently using the prior.

Linear centroid meta-learning We instantiate our study on the *linear centroid meta-learning problem* (also known as learning to learn around a common mean, Denevi et al. (2018b)), where we wish to learn a task-specific linear predictor $\mathbf{w}_t \in \mathbb{R}^d$ in the inner loop for each task t, and learn a "centroid" $\mathbf{w}_0$ in the outer loop that enables fast adaptation to $\mathbf{w}_t$ within each task:

> Find the best centroid $\mathbf{w}_0 \in \mathbb{R}^d$ for adapting to a linear predictor $\mathbf{w}_t$ on each task t.

Formally, we assume that we observe training data from $T \geq 1$ tasks, where for each task index t, we sample a task p_t (a distribution over $\mathbb{R}^d \times \mathbb{R}$) from some distribution of tasks Π, and observe n examples $(\mathbf{X}_t, \mathbf{y}_t) \in \mathbb{R}^{n\times d} \times \mathbb{R}^n$ that are drawn i.i.d. from p_t:

$$p_t \sim \Pi,\ (\mathbf{X}_t, \mathbf{y}_t) = \{(\mathbf{x}_{t,i}, y_{t,i})\}_{i=1}^n \text{ where } (\mathbf{x}_{t,i}, y_{t,i}) \overset{\text{iid}}{\sim} p_t. \tag{1}$$

We do not make further assumptions on (n, d); in particular, we allow the underdetermined setting $n \leq d$, in which there exists (one or many) interpolators $\widetilde{\mathbf{w}}_t$ that perfectly fit the data: $\mathbf{X}_t \widetilde{\mathbf{w}}_t = \mathbf{y}_t$.

Inner loop: Ridge solver with biased regularization towards the centroid Our goal in the inner loop is to find a linear predictor $\mathbf{w}_t$ that fits the data in task t while being close to the given "centroid" $\mathbf{w}_0 \in \mathbb{R}^d$. We instantiate this through ridge regression (i.e. linear regression with L_2 regularization) where the regularization biases $\mathbf{w}_t$ towards the centroid. Formally, for any $\mathbf{w}_0 \in \mathbb{R}^d$ and any dataset $(\mathbf{X}, \mathbf{y})$, we consider the algorithm

$$\begin{aligned}
&\mathcal{A}_\lambda(\mathbf{w}_0; \mathbf{X}, \mathbf{y}) \\
&:= \arg\min_{\mathbf{w}} \frac{1}{n} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w} - \mathbf{w}_0\|_2^2 \\
&= \mathbf{w}_0 + \left(\mathbf{X}^\top \mathbf{X} + n\lambda \mathbf{I}_d\right)^{-1} \mathbf{X}^\top (\mathbf{y} - \mathbf{X}\mathbf{w}_0),
\end{aligned}$$

where $\lambda > 0$ is the regularization strength (typically a tunable hyper-parameter). As we regularize by $\|\mathbf{w} - \mathbf{w}_0\|_2^2$, this inner solver encourages the solution to be close to $\mathbf{w}_0$, as we desire. Such a regularizer is widely used in practical meta-learning algorithms such as MetaOptNet (Lee et al., 2019) and Meta-MinibatchProx (Zhou et al., 2019). In addition, as $\lambda \to 0$, this solver recovers gradient descent fine-tuning: we have

$$\begin{aligned}
&\mathcal{A}_0(\mathbf{w}_0; \mathbf{X}, \mathbf{y}) := \lim_{\lambda \to 0} \mathcal{A}_\lambda(\mathbf{w}_0; \mathbf{X}, \mathbf{y}) \\
&= \mathbf{w}_0 + \mathbf{X}^\dagger (\mathbf{y} - \mathbf{X}\mathbf{w}_0) = \arg\min_{\mathbf{X}\mathbf{w} = \mathbf{y}} \|\mathbf{w} - \mathbf{w}_0\|_2^2,
\end{aligned}$$

where $\mathbf{X}^\dagger \in \mathbb{R}^{d\times n}$ denotes the pseudo-inverse of $\mathbf{X}$. This is the *minimum-distance* interpolator of $(\mathbf{X}, \mathbf{y})$ and also the solution found by gradient descent [1] on $\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$ initialized at $\mathbf{w}_0$. Therefore our ridge solver with $\lambda > 0$ can be seen as a generalized version of the gradient descent solver used in MAML (Finn et al., 2017).

Outer loop: Learning the best centroid In the outer loop, our goal is to find the best centroid $\mathbf{w}_0$. The standard approach in meta-learning is to perform a *train-validation split*, that is, (1) execute the inner solver on a first split of the task-specific data, and (2) evaluate the loss on a second split, yielding a function of $\mathbf{w}_0$ that we can optimize. This two-stage procedure can be written as

$$\begin{aligned}
&\text{Compute } \mathbf{w}_t(\mathbf{w}_0) = \mathcal{A}_\lambda(\mathbf{w}_0; \mathbf{X}_t^{\mathsf{train}}, \mathbf{y}_t^{\mathsf{train}}), \quad \text{and} \\
&\text{Evaluate } \left\|\mathbf{y}_t^{\mathsf{val}} - \mathbf{X}_t^{\mathsf{val}} \mathbf{w}_t(\mathbf{w}_0)\right\|_2^2.
\end{aligned}$$

where $(\mathbf{X}_t^{\mathsf{train}}, \mathbf{y}_t^{\mathsf{train}}) = \{(\mathbf{x}_{t,i}, y_{t,i})\}_{i=1}^{n_1}$ and $(\mathbf{X}_t^{\mathsf{val}}, \mathbf{y}_t^{\mathsf{val}}) = \{(\mathbf{x}_{t,i}, y_{t,i})\}_{i=n_1+1}^{n}$ are two disjoint splits of the per-task

[1] with a small step-size, or gradient flow.

data $(\mathbf{X}_t, \mathbf{y}_t)$ of size (n_1, n_2), with $n_1 + n_2 = n$. This amounts to the

Train-val method: Output $\widehat{\mathbf{w}}_{0,T}^{\mathsf{tr\text{-}val}}$ that minimizes

$$\widehat{L}_T^{\mathsf{tr\text{-}val}}(\mathbf{w}_0) = \frac{1}{T}\sum_{t=1}^{T} \ell_t^{\mathsf{tr\text{-}val}}(\mathbf{w}_0)$$
$$:= \frac{1}{T}\sum_{t=1}^{T} \frac{1}{2n_2} \left\| \mathbf{y}_t^{\mathsf{val}} - \mathbf{X}_t^{\mathsf{val}} \mathcal{A}_\lambda(\mathbf{w}_0; \mathbf{X}_t^{\mathsf{train}}, \mathbf{y}_t^{\mathsf{train}}) \right\|_2^2. \tag{2}$$

We compare the train-val method to an alternative version, where we do not perform the train-validation split, but instead use *all the per-task data for both training and evaluation*. Formally, this is to consider the

Train-train method: Output $\widehat{\mathbf{w}}_{0,T}^{\mathsf{tr\text{-}tr}}$ that minimizes

$$\widehat{L}_T^{\mathsf{tr\text{-}tr}}(\mathbf{w}_0) = \frac{1}{T}\sum_{t=1}^{T} \ell_t^{\mathsf{tr\text{-}tr}}(\mathbf{w}_0)$$
$$:= \frac{1}{T}\sum_{t=1}^{T} \frac{1}{2n} \left\| \mathbf{y}_t - \mathbf{X}_t \mathcal{A}_\lambda(\mathbf{w}_0; \mathbf{X}_t, \mathbf{y}_t) \right\|_2^2. \tag{3}$$

Let $L^{\{\mathsf{tr\text{-}val},\mathsf{tr\text{-}tr}\}}(\mathbf{w}_0) = \mathbb{E}\left[\ell_t^{\{\mathsf{tr\text{-}val},\mathsf{tr\text{-}tr}\}}(\mathbf{w}_0)\right]$ denote the corresponding expected losses. We remark that this expectation is equivalent to observing an infinite amount of tasks, but still with a finite (n, d) within each task.

(Meta-)Test time The meta-test time performance of any meta-learning algorithm is a joint function of the (learned) centroid $\mathbf{w}_0$ and the inner algorithm Alg. Upon receiving a new task $p_{T+1} \sim \Pi$ and training data $(\mathbf{X}_{T+1}, \mathbf{y}_{T+1}) \in \mathbb{R}^{n\times d} \times \mathbb{R}^n$, we run the inner loop Alg with prior $\mathbf{w}_0$ on the training data, and evaluate it on an (unseen) test example $(\mathbf{x}', y') \sim p_{T+1}$:

$$L^{\mathsf{test}}(\mathbf{w}_0; \mathsf{Alg}) := \mathbb{E}\left[\frac{1}{2}\left(\mathbf{x}'^\top \mathsf{Alg}(\mathbf{w}_0; \mathbf{X}_{T+1}, \mathbf{y}_{T+1}) - y'\right)^2\right].$$

Additionally, for both train-val and train-train methods, we need to ensure that the inner loop used for meta-test is exactly the same as that used in meta-training. Therefore, the meta-test performance for the train-val and train-train methods above should be evaluated as

$$L_{\lambda,n_1}^{\mathsf{test}}(\widehat{\mathbf{w}}_{0,T}^{\mathsf{tr\text{-}val}}) := L^{\mathsf{test}}(\widehat{\mathbf{w}}_{0,T}^{\mathsf{tr\text{-}val}}; \mathcal{A}_{\lambda,n_1}),$$
$$L_{\lambda,n}^{\mathsf{test}}(\widehat{\mathbf{w}}_{0,T}^{\mathsf{tr\text{-}tr}}) := L^{\mathsf{test}}(\widehat{\mathbf{w}}_{0,T}^{\mathsf{tr\text{-}tr}}; \mathcal{A}_{\lambda,n}),$$

where $\mathcal{A}_{\lambda,m}$ denotes the ridge solver with regularization strength $\lambda > 0$ on $m \le n$ data points. Finally, we let

$$\mathbf{w}_{0,\star}(\lambda; n) = \arg\min_{\mathbf{w}_0} L_{\lambda,n}^{\mathsf{test}}(\mathbf{w}_0) \tag{4}$$

denote the best centroid if the inner loop uses $\mathcal{A}_{\lambda,n}$. The performance of the train-val algorithm $\widehat{\mathbf{w}}_{0,T}^{\mathsf{tr\text{-}val}}$ should be compared against $\mathbf{w}_{0,\star}(\lambda, n_1)$, whereas the train-train algorithm $\widehat{\mathbf{w}}_{0,T}^{\mathsf{tr\text{-}tr}}$ should be compared against $\mathbf{w}_{0,\star}(\lambda, n)$.

3. The importance of sample splitting

We begin by analyzing the train-train and train-val methods defined in (2) and (3), in the agnostic setting where we do not make structural assumptions on the data distribution p_t.

In this case, we show that the importance of the sample splitting is clear even at the population level: the expected loss of the train-val method matches the test-time loss, whereas the expected loss of the train-train method does not match the test-time in general and have a different minimizer.

Theorem 1 (Properties of expected losses in the agnostic case)**.** *Suppose the task distributions satisfy $\mathbb{E}_{\mathbf{x}\sim p_t}[\mathbf{x}\mathbf{x}^\top] \succ \mathbf{0}$, $\mathbb{E}_{\mathbf{x}\sim p_t}[\|\mathbf{x}\|_2^4] < \infty$ and $\mathbb{E}_{(\mathbf{x},y)\sim p_t}[\|\mathbf{x}y\|_2] < \infty$ for almost surely all $p_t \sim \Pi$, but can be otherwise arbitrary. Then, we have the following:*

(a) (Unbiased loss for train-val method) For any $\lambda > 0$ and any (n_1, n_2) such that $n_1 + n_2 = n$, the expected loss of the train-val method is equal to the meta test-time loss, and thus minimized at the best test-time centroid:

$$L_{\lambda,n_1,n_2}^{\mathsf{tr\text{-}val}}(\mathbf{w}_0) = L_{\lambda,n_1}^{\mathsf{test}}(\mathbf{w}_0).$$

(b) (Biased loss for train-train method) There exists a distribution of tasks Π on $d = 1$ satisfying the above conditions, on which for any $n \ge 1$ and $\lambda > 0$, the expected loss of the train-train method is not equal to the test-time loss, and the minimizers are not equal:

$$L_{\lambda,n}^{\mathsf{tr\text{-}tr}}(\cdot) \neq L_{\lambda,n}^{\mathsf{test}}(\cdot), \quad \text{and}$$
$$\mathbf{w}_{0,\star}^{\mathsf{tr\text{-}tr}} := \arg\min_{\mathbf{w}_0} L^{\mathsf{tr\text{-}tr}}(\mathbf{w}_0) \neq \arg\min_{\mathbf{w}_0} L_{\lambda,n}^{\mathsf{test}}(\mathbf{w}_0).$$

Further, the excess test loss of $\mathbf{w}_{0,\star}^{\mathsf{tr\text{-}tr}}$ is bounded away from zero: $L_{\lambda,n}^{\mathsf{test}}(\mathbf{w}_{0,\star}^{\mathsf{tr\text{-}tr}}) - \min_{\mathbf{w}_0} L_{\lambda,n}^{\mathsf{test}}(\mathbf{w}_0) > 0$.

Theorem 1 makes clear the advantage of the train-val method when there is no structural assumption on the data distributions: The expected version of the train-val loss matches the meta test-time, whereas the expected version of the train-train loss has a bias in general. By standard consistency results (Van der Vaart, 2000), this advantage carries on to the sampled versions as well for large T. In other words, the train-val method is a "valid ERM" (empirical risk minimization) procedure for the test-time loss, whereas the train-train method is not a valid ERM.

Proof intuitions The proof of part (a) follows from direct calculations, whereas the proof of part (b) is trickier as we

need to construct a counter-example in which the expected loss of the train-train method is not equal the test-time loss for *any* λ, n. We provide such a construction in $d = 1$, where the distribution p_t has a certain asymmetry that results in a bias the train-train loss function for any λ and n. However, we expect such a bias to be present in general for any dimensions. The proof of Theorem 1 can be found in Appendix A.

4. Is sample splitting always optimal?

Theorem 1 states a negative result for the train-train method, showing that its expected loss and the meta test-time loss does not have the same values and minimizers. However, such a result does not preclude the possibility that there exists a data distribution on which the minimizers coincide (even though the loss values can still be different).

In this section, we construct a simple data distribution on which train-train method is indeed unbiased in terms of the minimizer of the expected loss, and compare its performance against the train-val method more explicitly.

Realizable linear model We consider the following instantiation of the (generic) meta-learning data distribution assumption in (1): We assume that each task p_t is specified by a $\mathbf{w}_t \in \mathbb{R}^d$ sampled from some distribution Π (overloading notation), and the observed data follows the noiseless linear model with ground truth parameter $\mathbf{w}_t$:

$$\mathbf{y}_t = \mathbf{X}_t \mathbf{w}_t. \tag{5}$$

Note that when $n \geq d$ and inputs are in general position, we are able to perfectly recover $\mathbf{w}_t$ (by solving linear equations), therefore the problem in the inner loop is easy. However, even in this case the outer loop problem is still nontrivial as we wish to learn the best centroid $\mathbf{w}_0$.

In this section, we make the following assumption on the distributions of $\mathbf{X}_t$ and $\mathbf{w}_t$:

Assumption A (Data distributions for realizable linear model)**.** *The inputs are standard Gaussian:* $\mathbf{x}_{t,i} \overset{\text{iid}}{\sim} \mathsf{N}(\mathbf{0}, \mathbf{I}_d)$. *The true coefficient* $\mathbf{w}_t$ *is independent of* $\mathbf{X}_t$ *and satisfies*

$$\mathrm{Cov}(\mathbf{w}_t) = \mathbb{E}_{\mathbf{w}_t}\left[(\mathbf{w}_t - \mathbf{w}_{0,\star})(\mathbf{w}_t - \mathbf{w}_{0,\star})^\top\right] = \frac{R^2}{d}\mathbf{I}_d, \tag{6}$$

for some fixed $R^2 > 0$*, and that the individual entries* $\{w_{t,i} - w_{0,\star,i}\}_{i\in[d],t\in[T]}$ *are i.i.d. mean-zero and* KR^2/d*-sub-Gaussian for some absolute constant* $K = O(1)$.

The sub-Gaussian assumption on $\mathbf{w}_t$ allows for a sharp concentration of the MSE to its expectation (over $\mathbf{w}_t$). The Gaussian input assumption allows for a precise characterization of certain ridge covariance type random matrices.

4.1. Population minimizers

We first show that on the realizable linear model (5), the test-time best centroids $\mathbf{w}_{0,\star}(\lambda, n) = \arg\min_{\mathbf{w}_0} L^{\mathsf{test}}_{\lambda,n}(\mathbf{w}_0)$ is the same for any (λ, n), and both the train-train and train-val methods are unbiased: Both expected losses are minimized at $\mathbf{w}_{0,\star}$.

Theorem 2 (Population minimizers on the realizable model)**.** *On the realizable linear model (5), suppose Assumption A holds. Then the test-time meta loss for all* $\lambda > 0$ *and all* n *is minimized at the same point, that is, the mean of the ground truth parameters:*

$$\begin{aligned}\mathbf{w}_{0,\star}(\lambda, n) &= \arg\min_{\mathbf{w}_0} L^{\mathsf{test}}_{\lambda,n}(\mathbf{w}_0) \\ &= \mathbf{w}_{0,\star} := \mathbb{E}_{\mathbf{w}_t\sim\Pi}[\mathbf{w}_t], \quad \textit{for all } \lambda > 0,\ n.\end{aligned}$$

Furthermore, for both the train-val method and the train-train method, the expected loss is minimized at $\mathbf{w}_{0,\star}$ *for any* $\lambda > 0$*,* n*, and* (n_1, n_2)*:*

$$\arg\min_{\mathbf{w}_0} L^{\mathsf{tr\text{-}val}}_{\lambda,n_1,n_2}(\mathbf{w}_0) = \arg\min_{\mathbf{w}_0} L^{\mathsf{tr\text{-}tr}}_{\lambda,n}(\mathbf{w}_0) = \mathbf{w}_{0,\star}.$$

Theorem 2 shows that both the train-val and train-train methods are in expectation minimized at the same optimal parameter $\mathbf{w}_{0,\star}$ which is the mean of $\mathbf{w}_t$. This is a consequence of the good structure in our realizable linear model (5): at a high level, $\mathbf{w}_{0,\star}$ is indeed the best centroid since it has (on average) the closest distance to a randomly sampled $\mathbf{w}_t$. The proof of Theorem 2 be found in Appendix B.

4.2. Precise comparison of rates

Theorem 2 suggests that we are now able to compare performance of the two methods based on their parameter estimation error (for estimating $\mathbf{w}_{0,\star}$).

We are now ready to state our two main theorems, which provide a precise comparison of the MSEs of the train-train and train-val methods under the realizable linear model.

Theorem 3 (Concentration of MSEs in the realizable linear model)**.** *In the realizable linear model (5), suppose Assumption A holds,* $T = \widetilde{\Omega}(d)$*,* $d/n = \Theta(1)$*,* $n_2/n = \Theta(1)$*, and* $\lambda = \Theta(1) > 0$*. Then with probability at least* $1 - Td^{-10}$*, the MSE of the train-train and train-val methods has the following concentrations, respectively:*

$$\left\|\widehat{\mathbf{w}}_0^{\mathsf{tr\text{-}tr}} - \mathbf{w}_{0,\star}\right\|_2^2 = \frac{R^2}{T}\left(C^{\mathsf{tr\text{-}tr}}_{d,n,\lambda} + \widetilde{O}\left(\sqrt{\frac{d}{T}} + \frac{1}{\sqrt{d}}\right)\right),$$

$$\left\|\widehat{\mathbf{w}}_0^{\mathsf{tr\text{-}val}} - \mathbf{w}_{0,\star}\right\|_2^2 = \frac{R^2}{T}\left(C^{\mathsf{tr\text{-}val}}_{d,n_1,n_2,\lambda} + \widetilde{O}\left(\sqrt{\frac{d}{T}} + \frac{1}{\sqrt{d}}\right)\right),$$

where $\widetilde{O}(\cdot)$ *hides* $\log(ndT)$ *factor. Further, the constants*

$C^{\text{tr-tr}}, C^{\text{tr-val}} = \Theta(1)$ *and have explicit expressions:*

$$C^{\text{tr-tr}}_{d,n,\lambda} = \frac{\frac{1}{d}\mathbb{E}\Big[\mathrm{tr}\Big((\widehat{\mathbf{\Sigma}}_n + \lambda \mathbf{I}_d)^{-4}\widehat{\mathbf{\Sigma}}_n^2\Big)\Big]}{\Big(\frac{1}{d}\mathbb{E}\Big[\mathrm{tr}\Big((\widehat{\mathbf{\Sigma}}_n + \lambda \mathbf{I}_d)^{-2}\widehat{\mathbf{\Sigma}}_n\Big)\Big]\Big)^2},$$

$$C^{\text{tr-val}}_{d,n_1,n_2,\lambda} = \frac{\frac{1}{dn_2}\mathbb{E}\Big[\mathrm{tr}\Big((\widehat{\mathbf{\Sigma}}_{n_1} + \lambda \mathbf{I}_d)^{-2}\Big)^2 + (n_2+1)\mathrm{tr}\Big((\widehat{\mathbf{\Sigma}}_{n_1} + \lambda \mathbf{I}_d)^{-4}\Big)\Big]}{\Big(\frac{1}{d}\mathbb{E}\Big[\mathrm{tr}\Big((\widehat{\mathbf{\Sigma}}_{n_1} + \lambda \mathbf{I}_d)^{-2}\Big)\Big]\Big)^2}$$

where $\widehat{\mathbf{\Sigma}}_n := \mathbf{X}_t^\top \mathbf{X}_t/n$ *denotes the empirical covariance of a standard Gaussian random matrix* $\mathbf{X}_t \in \mathbb{R}^{n\times d}$.

Theorem 3 asserts that the MSEs of both methods concentrate around R^2/T times a $\Theta(1)$ constant, when both T, d are large and $T = \widetilde{\Omega}(d)$ (so that the error terms vanish). This allows us to compare the performances of the train-train and train-val methods based on the constants. For a fair comparison, we look at the constants with optimal choices of λ and the split ratio, which we state in the following

Theorem 4 (Comparison of constants $C^{\text{tr-tr}}$ and $C^{\text{tr-val}}$)**.** *In the high-dimensional limiting regime* $d, n \to \infty$, $d/n \to \gamma \in (0,\infty)$*, the optimal constant of the train-train method obtained by tuning the regularization* $\lambda \in (0,\infty)$ *satisfies*

$$\inf_{\lambda>0} \lim_{d,n\to\infty, d/n\to\gamma} C^{\text{tr-tr}}_{d,n,\lambda} = \inf_{\lambda>0} \rho_{\lambda,\gamma} \overset{(\star)}{\le} \max\left\{1 + \frac{5}{27}\gamma, \frac{5}{27} + \gamma\right\},$$

where $\rho_{\lambda,\gamma} := 4\gamma^2\big[(\gamma-1)^2 + (\gamma+1)\lambda\big]/(\lambda+\gamma+1-\sqrt{(\lambda+\gamma+1)^2-4\gamma})^2/\big((\lambda+\gamma+1)^2-4\gamma\big)^{3/2}$*, and the inequality becomes equality at* $\gamma = 1$*. In contrast, the optimal rate of the train-val method by tuning the regularization* $\lambda \in (0,\infty)$ *and split ratio* $s \in (0,1)$ *is*

$$\inf_{\lambda>0, s\in(0,1)} \lim_{d,n\to\infty, d/n\to\gamma} C^{\text{tr-val}}_{d,ns,n(1-s),\lambda} = (1+\gamma).$$

As $\max\{1 + 5\gamma/27, 5/27 + \gamma\} < 1 + \gamma$ *for any* $\gamma > 0$*, the train-train method has a strictly better constant than the train-val method when* λ *and* s *are optimally tuned in both methods.*

Implications Theorem 4 shows that, perhaps surprisingly, the train-train method achieves a strictly better MSE (in terms of the constant) than the train-val method in the realizable linear model[2]. (See Figure 1(a) for a visualization of the exact optimal rates and the upper bound $(\star)$.) This suggests that the train-validation split may not be crucial when the data has structural assumptions such as realizability by the model. To the best of our knowledge, this is the first theoretical result that offers a disentangled comparison of meta-learning algorithms with and without sample splitting. Note that our result features an *optimal tuning of hyperparameters*: we compare the rates at the (theoretically) optimal λ for the train-train method and the optimal λ, n_1 for the train-val method. A closely related existing result is (Denevi et al., 2018b, Proposition 3) which proved a sample complexity upper bound for the linear centroid meta-learning problem; however, they only consider the train-val method, and their upper bounds do not tell the exact leading constants as in our Theorem 3.

We also remark that, while our theory considers the linear centroid meta-learning problem, our real data experiments in Section 5.2 suggests that the superiority of the train-train method may also hold on real meta-learning tasks with neural networks.

[2]The same conclusion also holds for the excess test loss, as the Hessian of the test loss is a rescaled identity, see Appendix C.2.

4.3. Overview of techniques

Here we provide an overview of the techniques in proving Theorem 3 and Theorem 4. We defer the full proofs to Appendix C and Appendix D respectively.

Closed-form expressions for $\widehat{\mathbf{w}}^{\text{tr-tr}}_{0,T}$ and $\widehat{\mathbf{w}}^{\text{tr-val}}_{0,T}$ Our first step is to obtain the following closed-form expressions for the estimation errors of both methods in the realizable linear model (see Lemma C.1):

$$\widehat{\mathbf{w}}^{\text{tr-tr}}_{0,T} - \mathbf{w}_{0,\star} = \left(\sum_{t=1}^T \mathbf{A}_t\right)^{-1} \sum_{t=1}^T \mathbf{A}_t(\mathbf{w}_t - \mathbf{w}_{0,\star}),$$

$$\widehat{\mathbf{w}}^{\text{tr-val}}_{0,T} - \mathbf{w}_{0,\star} = \left(\sum_{t=1}^T \mathbf{B}_t\right)^{-1} \sum_{t=1}^T \mathbf{B}_t(\mathbf{w}_t - \mathbf{w}_{0,\star}),$$

where

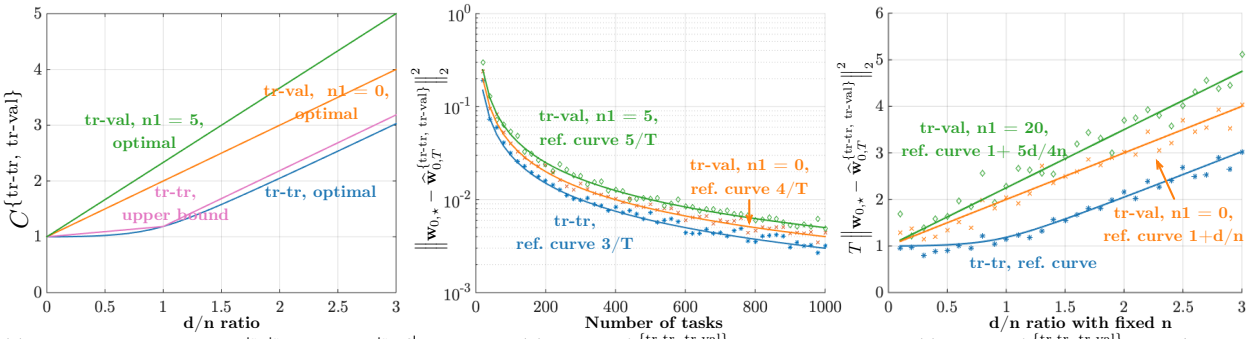
$$\mathbf{A}_t := \lambda^2\big(\mathbf{X}_t^\top\mathbf{X}_t/n + \lambda\mathbf{I}_d\big)^{-2}(\mathbf{X}_t^\top\mathbf{X}_t/n),$$

$$\mathbf{B}_t := \lambda^2\big(\mathbf{X}_t^{\text{train}\top}\mathbf{X}_t^{\text{train}}/n_1 + \lambda\mathbf{I}_d\big)^{-1}\big(\mathbf{X}_t^{\text{val}\top}\mathbf{X}_t^{\text{val}}/n_2\big) \cdot \big(\mathbf{X}_t^{\text{train}\top}\mathbf{X}_t^{\text{train}}/n_1 + \lambda\mathbf{I}_d\big)^{-1}.$$

These expressions simplify the estimation errors as the "weighted averages" of the $\{\mathbf{w}_t - \mathbf{w}_0\}$ with weighting matrices $\mathbf{A}_t$ and $\mathbf{B}_t$.

Sharp concentration to exact constants Our next step is to establish the concentration

$$\big\|\widehat{\mathbf{w}}^{\text{tr-tr}}_{0,T} - \mathbf{w}_{0,\star}\big\|_2^2 \overset{(i)}{\approx} \frac{R^2}{d} \cdot \mathrm{tr}\left(\left(\sum_{t=1}^T \mathbf{A}_t\right)^{-2}\left(\sum_{t=1}^T \mathbf{A}_t^2\right)\right)$$



(a) Optimal constants $\inf_\lambda C^{\text{tr-tr}}_{d,n,\lambda}$ and $\inf_\lambda C^{\text{tr-val}}_{d,n_1,n_2,\lambda}$ (b) MSE of $\widehat{\mathbf{w}}^{\{\text{tr-tr, tr-val}\}}_{0,T}$ v.s. T (c) MSE of $\widehat{\mathbf{w}}^{\{\text{tr-tr, tr-val}\}}_{0,T}$ v.s. d/n ratio

Figure 1. Panel (a) plots the exact constants in Theorem 4: The optimal train-train constant $\inf_\lambda C^{\text{tr-tr}}_{d,n,\lambda}$ (blue) and its upper bound ($\star$) (magenta), as well as the optimal train-val constant $\inf_\lambda C^{\text{tr-val}}_{d,n_1,n_2,\lambda}$ with $n_1 = 0$ (orange, optimal choice) and $n_1 = 5$ (green). (Optimal $\inf_\lambda C^{\text{tr-val}}_{d,n_1,n_2,\lambda}$ at each n_1 can be found in Lemma D.1.) Curves in panel (a) are used as reference curves in plots (b) and (c). Panel (b) plots the MSE of $\widehat{\mathbf{w}}^{\{\text{tr-tr,tr-val}\}}_{0,T}$ as the total number of tasks increases from 20 to 1000 with an increment of 20. We fix data dimension $d = 60$ and per-task sample size $n = 20$. For the train-val method, we experiment on $n_1 = 0$ and $n_1 = 5$. Panel (c) shows the rescaled MSE of $\widehat{\mathbf{w}}^{\{\text{tr-tr,tr-val}\}}_{0,T}$ as the ratio d/n varies from 0 to 3 (with $n = 100$ and $T = 300$).

$$\overset{(ii)}{\approx} \frac{R^2}{T}(\mathrm{tr}(\mathbb{E}[\mathbf{A}_t])/d)^{-2}\big(\mathrm{tr}\big(\mathbb{E}[\mathbf{A}_t^2]\big)/d\big) = \frac{R^2}{T}C^{\text{tr-tr}}_{d,n,\lambda}.$$

(and a similar result for $\widehat{\mathbf{w}}^{\text{tr-val}}_{0,T}$ using $\mathbf{B}_t$.) Above, (i) relies on the concentration of a certain quadratic form involving the $(\mathbf{w}_t - \mathbf{w}_0)$'s, following from the Hanson-Wright inequality (cf. Lemma C.5), and (ii) relies on the concentration of the matrices $\sum_{t=1}^T \mathbf{A}_t/T$ and $\sum_{t=1}^T \mathbf{A}_t^2/T$, using standard sub-Gaussian matrix concentration and a truncation argument (cf. Lemma C.4). Further calculating the expectations $\mathbb{E}[\mathbf{A}_t]$ and $\mathbb{E}[\mathbf{A}_t^2]$ gives the exact formula of $C^{\text{tr-tr}}_{d,n,\lambda}$ (cf. Lemma C.2) and finishes the proof of Theorem 3.

Optimizing and comparing $C^{\text{tr-tr}}_{d,n,\lambda}$ and $C^{\text{tr-val}}_{d,n_1,n_2,\lambda}$ The constants $C^{\text{tr-tr}}_{d,n,\lambda}$ and $C^{\text{tr-val}}_{d,n_1,n_2,\lambda}$ involve tunable hyperparameters λ (for both methods) and n_1 (for the train-val method). We use the following strategies to optimize the hyperparameters in each method, which combine to yield Theorem 4.

- For the train-val method, we show that the optimal tunable parameters for any (n, d) is taken at a special case $\lambda = \infty$ and $(n_1, n_2) = (0, n)$, at which the rate only depends on $\frac{1}{n_1}\mathbf{X}_t^{\text{train}\top}\mathbf{X}_t^{\text{train}}$ through its rank (and thus has a simple closed-form). We state this result in Lemma D.1. The proof builds on algebraic manipulations of the quantity $C^{\text{tr-val}}_{d,n,\lambda_1,\lambda_2}$, and can be found in Appendix D.1.
- For the train-train method, we apply random matrix theory to simplify the spectrum of $\frac{1}{n}\mathbf{X}_t^\top\mathbf{X}_t$ in the *proportional limit* where $d, n \to \infty$ and d/n stays as a constant (Bai & Silverstein, 2010; Anderson et al., 2010), and obtain a closed-form expression of the asymptotic MSE for any $\lambda > 0$, which we can analytically optimize over λ. We state this result in Theorem D.1. The proof builds on the Stieltjes transform and its "derivative trick" (Dobriban et al., 2018), and is deferred to Appendix D.2.

5. Experiments

5.1. Simulations

We experiment on the realizable linear model studied in Section 4. Recall that the observed data of the t-th task are generated as

$$\mathbf{y}_t = \mathbf{X}_t\mathbf{w}_t, \quad \text{with} \quad \mathbf{x}_{t,i} \overset{\text{iid}}{\sim} \mathsf{N}(0, \mathbf{I}_d).$$

We independently generate $\mathbf{w}_t \overset{\text{iid}}{\sim} \mathsf{N}(\mathbf{w}_{0,\star}, \mathbf{I}_d/\sqrt{d})$, where $\mathbf{w}_{0,\star}$ is the linear centroid and the corresponding $R^2 = 1$ here. The goal is to learn the linear centroid $\mathbf{w}_{0,\star}$ using the train-train method and train-val method, i.e., minimizing $\widehat{L}^{\text{tr-tr}}_T$ and $\widehat{L}^{\text{tr-val}}_T$, respectively. Recall that the optimal closed-form solutions $\widehat{\mathbf{w}}^{\{\text{tr-tr,tr-val}\}}_{0,T}$ are given in Section 4.3. We measure the performance of the train-train and train-val methods using the ℓ_2-error $\|\mathbf{w}_{0,\star} - \widehat{\mathbf{w}}^{\{\text{tr-tr,tr-val}\}}_{0,T}\|_2^2$.

Result Figure 1 shows the performance of the train-train and train-val methods on simulated linear centroid meta-learning problems. Across all simulations, we optimally tune the regularization coefficient λ in the train-train method, and use a sufficiently large $\lambda = 2000$ in the train-val method (according to Lemma D.1). Observe that the MSEs of the two methods decay at rate $O(1/T)$ (Figure 1(b)). Further, the performance of the two methods

Table 1. Comparison of train-train and train-val on few-shot image classification (accuracy in %).

	method	1-shot 5-way	5-shot 5-way	1-shot 20-way	5-shot 20-way
miniImage	train-val	48.76 ± 0.87	63.56 ± 0.95	17.52 ± 0.49	21.32 ± 0.54
	train-train	$\mathbf{50.77 \pm 0.90}$	$\mathbf{67.43 \pm 0.89}$	$\mathbf{21.17 \pm 0.38}$	$\mathbf{34.30 \pm 0.41}$
	method	1-shot 5-way	5-shot 5-way	1-shot 10-way	5-shot 10-way
tieredImage	train-val	50.61 ± 1.12	67.30 ± 0.98	29.18 ± 0.57	43.15 ± 0.72
	train-train	$\mathbf{54.37 \pm 0.93}$	$\mathbf{71.45 \pm 0.94}$	$\mathbf{35.56 \pm 0.60}$	$\mathbf{54.50 \pm 0.71}$

in our simulation closely matches the theoretical result in Theorem 4, and the train-train method reliably outperforms the train-val method at all d/n with a moderately large T (Figure 1(c)). In Appendix G, we additionally investigate the effect of averaging the loss over multiple splits in the train-val method (a "cross-validation" type loss).

5.2. Few-shot image classification

We further compare train-train and train-val type methods on the benchmark few-shot image classification tasks miniImageNet (Ravi & Larochelle, 2017) and tieredImageNet (Ren et al., 2018).

Methods We instantiate the train-train and train-val method in the centroid meta-learning setting with a ridge solver. The methods are almost exactly the same as in our theoretical setting in (2) and (3), with the only differences being that the parameters $\mathbf{w}_t$ (and hence $\mathbf{w}_0$) parametrize a deep neural network instead of a linear classifier, and the loss function is the cross-entropy instead of squared loss. Mathematically, we minimize the following two loss functions:

$$L^{\text{tr-val}}_{\lambda,n_1}(\mathbf{w}_0) := \frac{1}{T}\sum_{t=1} \ell^{\text{tr-val}}_t(\mathbf{w}_0)$$

$$= \frac{1}{T}\sum_{t=1}^{T} \ell\Big(\arg\min_{\mathbf{w}_t} \ell(\mathbf{w}_t; \mathbf{X}^{\text{train}}_t, \mathbf{y}^{\text{train}}_t) + \lambda \|\mathbf{w}_t - \mathbf{w}_0\|_2^2 ; \mathbf{X}^{\text{val}}_t, \mathbf{y}^{\text{val}}_t \Big),$$

$$L^{\text{tr-tr}}_{\lambda}(\mathbf{w}_0) := \frac{1}{T}\sum_{t=1}^{T} \ell^{\text{tr-tr}}_t(\mathbf{w}_0)$$

$$= \frac{1}{T}\sum_{t=1}^{T} \ell\Big(\arg\min_{\mathbf{w}_t} \ell(\mathbf{w}_t; \mathbf{X}_t, \mathbf{y}_t) + \lambda \|\mathbf{w}_t - \mathbf{w}_0\|_2^2 ; \mathbf{X}_t, \mathbf{y}_t \Big),$$

where $(\mathbf{X}_t, \mathbf{y}_t)$ is the data for task t of size n, and $(\mathbf{X}^{\text{train}}_t, \mathbf{y}^{\text{train}}_t)$ and $(\mathbf{X}^{\text{val}}_t, \mathbf{y}^{\text{val}}_t)$ is a split of the data of size (n_1, n_2). We note that both loss functions above have been considered in prior work ($L^{\text{tr-val}}$ in iMAML (Rajeswaran et al., 2019), and $L^{\text{tr-tr}}$ in Meta-MinibatchProx (Zhou et al., 2019)), though we use slightly different implementation details from these prior work to make sure that the two methods here are exactly the same except for whether the split is used. Additional details about the implementation can be found in Appendix F.

We experiment on miniImageNet (Ravi & Larochelle, 2017) and tieredImageNet (Ren et al., 2018) datasets. MiniImageNet consists of 100 classes of images from ImageNet (Krizhevsky et al., 2012) and each class has 600 images of resolution $84 \times 84 \times 3$. We use 64 classes for training, 16 classes for validation, and the remaining 20 classes for testing (Ravi & Larochelle, 2017). TieredImageNet consists of 608 classes from the ILSVRC-12 data set (Russakovsky et al., 2015) and each image is also of resolution $84 \times 84 \times 3$.

We adopt the episodic training procedure (Vinyals et al., 2016; Finn et al., 2017; Zhou et al., 2019; Rajeswaran et al., 2019) In each "N-way K-shot setting" (in Table 1), in meta-test, each task provides an N-way K-shot dataset for the model adaptation. In meta-training, for each task we sample an N-way $(K+1)$-shot dataset (and does not allow the algorithm to tune the size of this dataset), so that each task *only* has $n = N(K+1)$ examples, and we allow the algorithm to tune $n_1 \in [0, n]$[3]. Table 1 uses the default choice of an even split $n_1 = n_2 = n/2$ following (Zhou et al., 2019; Rajeswaran et al., 2019). For example, for a 5-way 5-shot classification setting, each task contains $5 \times (5+1) = 30$ total images, and we set $n_1 = n_2 = 15$. (We additionally investigate the optimality of this split ratio in Appendix F.1.) We report the average accuracy over $2,000$ random test episodes with 95% confidence interval.

Results We find that the train-train method consistently outperforms the train-val method (Table 1). Specifically, on miniImageNet, train-train method outperforms train-val by 2.01% and 3.87% on the 1-shot 5-way and 5-shot 5-way tasks respectively; On tieredImageNet, train-train on average improves by about 6.40% on the four testing cases. These results show the advantages of train-train method over train-val and support our theoretical findings in Theorem 4.

[3]This setting deviates slightly from our setting (in Section 2 & 3) that meta-train and meta-test needs to have exactly the same (n_1, n_2), but allows a fair comparison of algorithms under the realistic scenario of limited per-task data (fixed n) and fixed data size at meta test time.

6. Conclusion

We study the importance of train-validation split on the linear-centroid meta-learning problem, and show that the necessity and optimality of train-validation split depends greatly on whether the tasks are structured: the sample splitting is necessary in general situations, and not necessary and non-optimal when the tasks are nicely structured. It would be of interest to study whether similar conclusions hold on other meta-learning problems such as learning representations, or how our insights can guide the design of meta-learning algorithms with better empirical performance.

Acknowledgment

We thank Song Mei, Wei Hu, Nikunj Saunshi, Huaxiu Yao, Weihao Kong for the many insightful discussions. We thank the anonymous reviewers for the helpful feedback on our paper.

References

Alquier, P., Mai, T. T., and Pontil, M. Regret bounds for lifelong learning. *arXiv preprint arXiv:1610.08628*, 2016.

Anderson, G. W., Guionnet, A., and Zeitouni, O. *An introduction to random matrices*, volume 118. Cambridge university press, 2010.

Argyriou, A., Evgeniou, T., and Pontil, M. Multi-task feature learning. In *Advances in neural information processing systems*, pp. 41–48, 2007.

Arnold, S. M., Iqbal, S., and Sha, F. When maml can adapt fast and how to assist when it cannot. *arXiv preprint arXiv:1910.13603*, 2019.

Bai, Y. and Lee, J. D. Beyond linearization: On quadratic and higher-order approximation of wide neural networks. *arXiv preprint arXiv:1910.01619*, 2019.

Bai, Z. and Silverstein, J. W. *Spectral analysis of large dimensional random matrices*, volume 20. Springer, 2010.

Baxter, J. A model of inductive bias learning. *J. Artif. Int. Res.*, 2000.

Caruana, R. Multitask learning. *Machine Learning*, 28 (1):41–75, Jul 1997. ISSN 1573-0565. doi: 10.1023/A:1007379606734. URL https://doi.org/10.1023/A:1007379606734.

Collins, L., Mokhtari, A., and Shakkottai, S. Why does maml outperform erm? an optimization perspective. *arXiv preprint arXiv:2010.14672*, 2020.

Denevi, G., Ciliberto, C., Stamos, D., and Pontil, M. Incremental learning-to-learn with statistical guarantees. *arXiv preprint arXiv:1803.08089*, 2018a.

Denevi, G., Ciliberto, C., Stamos, D., and Pontil, M. Learning to learn around a common mean. In *Advances in Neural Information Processing Systems*, pp. 10169–10179, 2018b.

Dobriban, E., Wager, S., et al. High-dimensional asymptotics of prediction: Ridge regression and classification. *The Annals of Statistics*, 46(1):247–279, 2018.

Du, S. S., Hu, W., Kakade, S. M., Lee, J. D., and Lei, Q. Few-shot learning via learning the representation, provably. *arXiv preprint arXiv:2002.09434*, 2020.

Evgeniou, T., Micchelli, C. A., and Pontil, M. Learning multiple tasks with kernel methods. *Journal of machine learning research*, 6(Apr):615–637, 2005.

Fallah, A., Mokhtari, A., and Ozdaglar, A. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, pp. 1082–1092, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135, 2017.

Finn, C., Rajeswaran, A., Kakade, S., and Levine, S. Online meta-learning. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. *arXiv preprint arXiv:1806.04910*, 2018.

Galanti, T., Wolf, L., and Hazan, T. A theoretical framework for deep transfer learning. *Information and Inference: A Journal of the IMA*, 5(2):159–209, 2016.

Gao, K. and Sener, O. Modeling and optimization trade-off in meta-learning. *arXiv preprint arXiv:2010.12916*, 2020.

Goldblum, M., Reich, S., Fowl, L., Ni, R., Cherepanova, V., and Goldstein, T. Unraveling meta-learning: Understanding feature representations for few-shot tasks. *arXiv preprint arXiv:2002.06753*, 2020.

Gu, J., Wang, Y., Chen, Y., Cho, K., and Li, V. O. Meta-learning for low-resource neural machine translation. *arXiv preprint arXiv:1808.08437*, 2018.

Ji, K., Lee, J. D., Liang, Y., and Poor, H. V. Convergence of meta-learning with task-specific adaptation over partial parameters. *arXiv preprint arXiv:2006.09486*, 2020.

Khodak, M., Balcan, M.-F., and Talwalkar, A. Adaptive gradient-based meta-learning methods. *arXiv preprint arXiv:1906.02717*, 2019.

Krizhevsky, A., Sutskever, I., and Hinton, G. Imagenet classification with deep convolutional neural networks. pp. 1097–1105, 2012.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

Liu, H., Palatucci, M., and Zhang, J. Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 649–656, 2009.

Liu, H., Wang, L., and Zhao, T. Calibrated multivariate regression with application to neural semantic basis discovery. *Journal of machine learning research: JMLR*, 16: 1579, 2015.

Maurer, A., Pontil, M., and Romera-Paredes, B. The benefit of multitask representation learning. *The Journal of Machine Learning Research*, 17(1):2853–2884, 2016.

McNamara, D. and Balcan, M.-F. Risk bounds for transferring representations with and without fine-tuning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2373–2381. JMLR. org, 2017.

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

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgMkCEtPB.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pp. 113–124, 2019.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. 2017.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J., Larochelle, H., and Zemel, R. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., and Bernstein, M. Imagenet large scale visual recognition challenge. 115(3):211–252, 2015.

Saunshi, N., Zhang, Y., Khodak, M., and Arora, S. A sample complexity separation between non-convex and convex meta-learning. *arXiv preprint arXiv:2002.11172*, 2020.

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

Setlur, A., Li, O., and Smith, V. Is support set diversity necessary for meta-learning?, 2020.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. pp. 4077–4087, 2017.

Thrun, S. and Pratt, L. *Learning to Learn: Introduction and Overview*, pp. 3–17. Springer US, Boston, MA, 1998. ISBN 978-1-4615-5529-2. doi: 10.1007/978-1-4615-5529-2_1. URL https://doi.org/10.1007/978-1-4615-5529-2_1.

Tripuraneni, N., Jin, C., and Jordan, M. I. Provable meta-learning of linear representations. *arXiv preprint arXiv:2002.11684*, 2020a.

Tripuraneni, N., Jordan, M. I., and Jin, C. On the theory of transfer learning: The importance of task diversity. *arXiv preprint arXiv:2006.11650*, 2020b.

Van der Vaart, A. W. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

Vershynin, R. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

Wang, H., Sun, R., and Li, B. Global convergence and induced kernels of gradient-based meta-learning with neural nets. *arXiv preprint arXiv:2006.14606*, 2020a.

Wang, L., Cai, Q., Yang, Z., and Wang, Z. On the global optimality of model-agnostic meta-learning. In *International Conference on Machine Learning*, pp. 9837–9846. PMLR, 2020b.

Wang, X., Yuan, S., Wu, C., and Ge, R. Guarantees for tuning the step size using a learning-to-learn approach. *arXiv preprint arXiv:2006.16495*, 2020c.

Xie, Y., Jiang, H., Liu, F., Zhao, T., and Zha, H. Meta learning with relational information for short sequences. In *Advances in Neural Information Processing Systems*, pp. 9904–9915, 2019.

Yao, H., Huang, L., Wei, Y., Tian, L., Huang, J., and Li, Z. Don't overlook the support set: Towards improving generalization in meta-learning. *arXiv preprint arXiv:2007.13040*, 2020.

Zhou, P., Yuan, X., Xu, H., Yan, S., and Feng, J. Efficient meta learning via minibatch proximal update. In *Advances in Neural Information Processing Systems*, pp. 1534–1544, 2019.