Modular Meta-Learning with Shrinkage

Yutian Chen, Abram L. Friesen, Feryal Behbahani,
David Budden, Matthew W. Hoffman, Arnaud Doucet, Nando de Freitas

DeepMind
London, UK

{yutianc,abef}@google.com

Abstract

Most gradient-based approaches to meta-learning do not explicitly account for the fact that different parts of the underlying model adapt by different amounts when applied to a new task. For example, the input layers of an image classification convnet typically adapt very little, while the output layers can change significantly. This can cause parts of the model to begin to overfit while others underfit. To address this, we introduce a hierarchical Bayesian model with per-module shrinkage parameters, which we propose to learn by maximizing an approximation of the predictive likelihood using implicit differentiation. Our algorithm subsumes Reptile and outperforms variants of MAML on two synthetic few-shot meta-learning problems.

1 Introduction

The goal of meta-learning is to extract common knowledge from a set of training tasks in order to solve held-out tasks more efficiently and accurately. One avenue for learning and re-using this knowledge is to learn a set of modules that can be re-used or re-purposed at test time as needed. Modularity is intrinsic to deep learning, and examples range from receptive fields or layers to larger components, such as perception or policy networks. This modularity enables pre-trained convolutional neural networks to be rapidly fine-tuned on other image classification datasets, for example. However, most meta-learning algorithms that use test-time adaptation of a learned model, such as MAML [1] and Reptile [2], do not explicitly account for the modularity present in their models.

We propose here a hierarchical Bayesian modelling approach to modular meta-learning. The parameters within a module are assumed conditionally independent across tasks and their mean follows a normal distribution parameterized by a per-module "central" parameter and variance term, which acts as a local shrinkage parameter; see, e.g., Gelman et al. [3]. As the marginal likelihood is typically intractable in the scenarios we are interested in, we estimate the shrinkage parameters by maximizing an approximation of a predictive likelihood criterion using implicit differentiation. Empirically, we find that this approach is numerically stable, and we provide a theoretical analysis on a toy example suggesting it exhibits good properties. Our approach is complementary to that of Alet et al. [4], which proposed a modular method to learn model structures. More related work is discussed in Appendix A.

We evaluate our shrinkage-based approach on two synthetic few-shot meta-learning tasks as a proof of concept. We show that properly accounting for modularity is crucial for achieving good performance in these tasks. Our method outperforms Reptile [2], MAML [1], and a modular variant of MAML.

2 Hierarchical Bayes formulation of modular meta-learning

We consider a multi-task learning scenario with a number of tasks bearing some similarity to each other. For each task $\mathcal{T}_t$ indexed by t, a finite set of N_t observations $\mathcal{D}_t = \{\mathbf{x}_n\}_{n=1}^{N_t}$ is assumed

3rd Workshop on Meta-Learning at NeurIPS 2019, Vancouver, Canada.

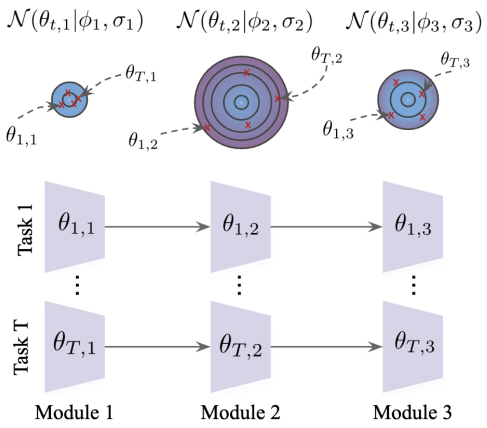


Figure 1: Modular meta-learning: a shrinkage parameter σ_m is associated with each module to control the deviation of task-dependent parameters $\boldsymbol{\theta}_{m,t}$ from the central value $\boldsymbol{\phi}_m$.

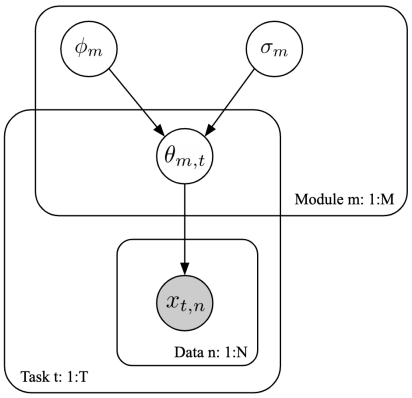


Figure 2: Graphical representation of the Bayesian hierarchical model described in Eq. (1).

available and $\mathcal{D}_t$ is modelled using a probabilistic model with parameters $\boldsymbol{\theta}_t$; these parameters being partitioned into M modules; i.e. $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{1:M,t} = (\boldsymbol{\theta}_{1,t}, \ldots, \boldsymbol{\theta}_{m,t}, \ldots, \boldsymbol{\theta}_{M,t})$, where $\boldsymbol{\theta}_{m,t} \in \boldsymbol{\Phi}_m \subseteq \mathbb{R}^{D_m}$. For example, $\boldsymbol{\theta}_{m,t}$ could be the weights of the m-th layer of a neural network for task t.

To model the relationship between tasks, we adopt a hierarchical Bayesian approach. We assume that the parameters for a task $\boldsymbol{\theta}_t$ are conditionally independent of those of all other tasks given some "central" parameters $\boldsymbol{\phi} = \boldsymbol{\phi}_{1:M} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_m, \ldots, \boldsymbol{\phi}_M)$, with $\boldsymbol{\theta}_{m,t} \sim \mathcal{N}(\boldsymbol{\theta}_{m,t}|\boldsymbol{\phi}_m, \sigma_m^2\mathbf{I})$ for all m, where $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the normal of mean $\boldsymbol{\mu}$ with covariance matrix $\boldsymbol{\Sigma}$, and $\mathbf{I}$ is the identity matrix with appropriate dimensionality. The parameter σ_m is a shrinkage parameter quantifying how $\boldsymbol{\theta}_{m,t}$ can deviate from $\boldsymbol{\phi}_m$. If $\sigma_m \approx 0$, then $\boldsymbol{\theta}_{m,t} \approx \boldsymbol{\phi}_m$, i.e., when σ_m shrinks to zero the parameters of module m become task independent; see Fig. 1 for an illustration. We assign a non-informative prior to $\boldsymbol{\phi}$ and $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_M)$, and follow an empirical Bayes approach to learn their values from the data. This allows the model to automatically decide which modules to re-use and which to adapt.

If we denote by $\mathcal{D} = (\mathcal{D}_1, ..., \mathcal{D}_T)$ the collection of observations corresponding to T tasks and $\boldsymbol{\sigma} = (\sigma_1, ..., \sigma_M)$, then the Bayesian hierarchical model considered here can be summarized by the following probability density (with graphical model shown in Fig. 2)

$$p(\boldsymbol{\phi}, \boldsymbol{\theta}, \mathcal{D}|\boldsymbol{\sigma}) = p(\boldsymbol{\phi}) \prod_{m=1}^{M} \prod_{t=1}^{T} \mathcal{N}(\boldsymbol{\theta}_{m,t}|\boldsymbol{\phi}_m, \sigma_m^2\mathbf{I}) \prod_{t=1}^{T} \prod_{n=1}^{N_t} p(\mathbf{x}_{t,n}|\boldsymbol{\theta}_t). \tag{1}$$

A standard learning strategy is to maximize the marginal likelihood $p(\mathcal{D}|\boldsymbol{\sigma})$ to obtain a point estimate of $\boldsymbol{\sigma}$, and then compute the resulting posterior $p(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathcal{D}, \boldsymbol{\sigma})$. However, in standard applications of meta-learning, this approach does not scale due to the intractability of marginalization. In the next section, we propose an approximate, scalable, Bayesian approach to parameter estimation in this model.

3 Learning strategy

To deal with the fact that a large, possibly infinite, number of tasks is available, we propose an iterative algorithm that at each iteration first samples a batch of tasks $\mathcal{T}_t \sim p(\mathcal{T})$ for $t = 1, ..., T$ and collects the corresponding datasets $\mathcal{D}_1, ..., \mathcal{D}_T$. The resulting probabilistic model for these datasets is thus of the form (1). As the corresponding marginal likelihood $p(\mathcal{D}|\boldsymbol{\sigma})$ and posterior $p(\boldsymbol{\phi}, \boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\sigma})$ are typically intractable, it might be tempting to maximize the joint distribution (1) w.r.t. to $\boldsymbol{\phi}, \boldsymbol{\sigma}, \boldsymbol{\theta}$ to estimate those parameters. Unfortunately, this approach fails as explained on a toy example in Appendix B. In short, even if the model is correctly specified, the optimal value of $\boldsymbol{\sigma}$ when maximizing the joint distribution is $\mathbf{0}$, and modules do not adapt when $\boldsymbol{\sigma} = \mathbf{0}$.

We instead take an approach similar to that of many recent meta-learning algorithms and split each dataset $\mathcal{D}_t$ into train and validation subsets, $\mathcal{D}_t^{\text{train}}$ and $\mathcal{D}_t^{\text{val}}$, respectively. We estimate the task parameters $\boldsymbol{\theta}_{1:T}$ and shared meta parameters $\boldsymbol{\phi}$ with MAP (maximum a posteriori) on the train

subsets $\mathcal{D}_{1:T}^{\text{train}}$ given $\boldsymbol{\sigma}$. This is equivalent to maximizing the log-joint density in Eq. (1), giving

$$\hat{\boldsymbol{\theta}}_{1:T}(\boldsymbol{\sigma}), \hat{\boldsymbol{\phi}}(\boldsymbol{\sigma}) = \underset{\boldsymbol{\theta}_{1:T}, \boldsymbol{\phi}}{\operatorname{argmax}} \ell_{\text{MAP}}, \quad \text{where} \quad \ell_{\text{MAP}} := \log p\left(\boldsymbol{\theta}_{1:T}, \boldsymbol{\phi} | \mathcal{D}_{1:T}^{\text{train}}, \boldsymbol{\sigma}\right) . \tag{2}$$

Given these, we estimate the shrinkage parameters $\boldsymbol{\sigma}$ by maximizing the predictive log-likelihood on the validation subsets $\mathcal{D}_{1:T}^{\text{val}}$:

$$\hat{\boldsymbol{\sigma}} = \underset{\boldsymbol{\sigma}}{\operatorname{argmax}} \log p\left(\mathcal{D}_{1:T}^{\text{val}} | \mathcal{D}_{1:T}^{\text{train}}, \boldsymbol{\sigma}\right) \approx \underset{\boldsymbol{\sigma}}{\operatorname{argmax}} \sum_{t=1}^{T} \log p(\mathcal{D}_t^{\text{val}} | \hat{\boldsymbol{\theta}}_t(\boldsymbol{\sigma})) := \underset{\boldsymbol{\sigma}}{\operatorname{argmax}} \ell_{\text{PLL}} , \tag{3}$$

where the marginalization over the posterior of $\boldsymbol{\theta}_t$ is approximated by the MAP point estimate. Using the MAP approximation of $\boldsymbol{\theta}_t$ within the predictive log-likelihood ensures that our meta-train and meta-test time procedures match, and that the meta-train metric directly optimizes the metric being evaluated at meta-test. Additionally, we show in Appendix B.2 that this provides a consistent estimate of $\boldsymbol{\sigma}$ on a toy example under regularity conditions. This type of end-to-end meta-learning objective is similar to various recent works such as Ravi and Larochelle [5] and Finn et al. [1]. However, in contrast to their emphasis on fast adaptation with a small number of adaptation steps, we are interested in sample efficiency, and thus allow sufficient time for the task adaptation to converge. Solving Eq. (3) requires solving Eq. (2), which can require expensive second-order derivatives; however, we show in Appendix C using implicit differentiation that we can approximate the derivative as

$$\frac{\partial \ell_{\text{PLL}}(\boldsymbol{\sigma})}{\partial \log \sigma_m^2} \approx \sum_{t=1}^{T} (\boldsymbol{\theta}_{m,t} - \boldsymbol{\phi}_m)^\top \frac{\partial}{\partial \hat{\boldsymbol{\theta}}_{m,t}} \log p\left(\mathcal{D}_t^{\text{val}} | \hat{\boldsymbol{\theta}}_t\right) . \tag{4}$$

Our resulting meta-learning algorithm is shown in Algorithm 1, where $\text{SGD}_k(\ell)$ corresponds to taking k steps with stochastic gradient descent (or an adaptive optimizer such as Adam) on loss ℓ. Notice that Reptile [2] is a special case of our method when $\boldsymbol{\sigma}$ tends to ∞ and an appropriate learning rate is used. It is also possible to estimate $\boldsymbol{\phi}$ with the predictive log-likelihood objective. We omit the derivation but the approximate gradient for $\boldsymbol{\phi}$ is then equivalent to the first-order MAML update [1].

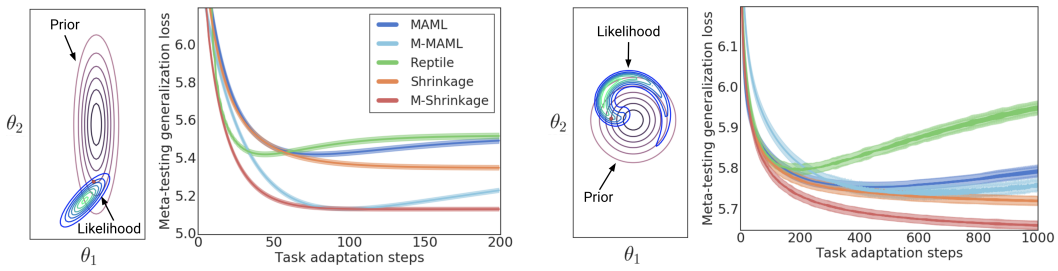
4 Experimental evaluation

We evaluate our proposed method Shrinkage, along with variants of MAML [1] and Reptile [2] on two synthetic few-shot meta-learning domains constructed from hierarchical normal distributions. For each task t, we first sample latent variables $\theta_{m,t} \sim \mathcal{N}(\theta_{m,t} | \phi_m, \sigma_m^2)$ for each dimension m, and then observations $\mathbf{x}_{t,n} \sim \mathcal{N}(\mathbf{x}_{t,n} | \boldsymbol{\mu}_t(\boldsymbol{\theta}_t), \Xi)$. Parameters $\boldsymbol{\phi}$ and $\boldsymbol{\sigma}$ are fixed but unknown, and different dimensions of $\boldsymbol{\sigma}$ have different values. To assess the efficacy of the learning strategy, we use a data generating process that matches our modeling assumptions. The data distribution's mean $\boldsymbol{\mu}_t$ is a function of $\boldsymbol{\theta}_t$ and is the main aspect that changes between experiments. The observation noise variance Ξ is a fixed and known diagonal matrix. The problem in each domain is to learn the parameters $\boldsymbol{\theta}_{\tilde{t}}$ of a new task $\mathcal{T}_{\tilde{t}}$ given a few observations $\{\mathbf{x}_{\tilde{t},n}\}$. The main difference between the two evaluation domains is that $\boldsymbol{\mu}_t$ is a linear function of $\boldsymbol{\theta}_t$ in the first and is non-linear in the second. Fig. 3 illustrates the experiments in 2-D, and Appendix D contains a precise specification of both.

Algorithm 1: Shrinkage-based meta-learning.

Input: Task distribution $p(\mathcal{T})$ and inner steps K.
initialize $\boldsymbol{\phi}$ and $\boldsymbol{\sigma}$
while *not done* **do**
  $\{\mathcal{T}_t\} \leftarrow$ sample batch of tasks from $p(\mathcal{T})$
  initialize $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\phi}$ for each task in batch $\{\mathcal{T}_t\}$
  for *each task* $\mathcal{T}_t$ **do**
    $\boldsymbol{\theta}_t \leftarrow \text{SGD}_K(\ell_{\text{MAP}})$ // *Eq.* (2).
  end
  $\boldsymbol{\phi} \leftarrow \text{SGD}_1(\ell_{\text{MAP}})$ // *Eq.* (2).
  $\log \boldsymbol{\sigma}^2 \leftarrow \text{SGD}_1(\ell_{\text{PLL}})$ // *Eq.* (4).
end

To compare the algorithms, we use the negative log-likelihood up to a constant as the loss, and compare the generalization loss of each algorithm after it adapts to the new task with multiple steps of gradient descent. For MAML and Shrinkage, we evaluate both the standard (non-modular) versions and modular versions that learn module-specific parameters, denoted by the "M-" prefix. To ensure a fair comparison, we increase the flexibility of MAML to match Shrinkage by learning the learning rate of the inner-loop gradient update for each module, similar to Antoniou et al. [6], who do this to stabilize MAML as opposed to enabling modularity. Similarly, M-Shrinkage learns a separate σ_m for each



(a) Experiment 1: Linear transform.

(b) Experiment 2: Nonlinear spiral transform.

Figure 3: Illustration of the hierarchical normal distributions for both experiments and corresponding performance results. The loss figures each show the mean generalization loss with 68% credible interval for 1000 test tasks for all algorithms versus test-time adaptation steps.

module and Shrinkage learns a single σ for all modules. In both experiments, the modular algorithms treat each dimension of $\boldsymbol{\theta}_t$ as a separate module, whereas the underlying distributions have only two modules, to make the task more challenging for the modular algorithms. We performed extensive random search for the hyperparameters of each algorithm (i.e., learning rates, number of adaptation steps, and initial values of $\boldsymbol{\sigma}$), choosing the values that minimized the meta-test validation loss.

Linear transform. We begin with a simple model – a two-module joint normal distribution over $\boldsymbol{\theta}_t$. To make each task nontrivial, we restrict the posterior to a narrow subspace near the $\sum_m \boldsymbol{\theta}_{m,t}$ hyperplane, causing gradient descent to converge slowly regardless of the number of observations.

Fig. 3a shows a 2-D example of this model as well as the performance of all algorithms on it. With the small number of observations for each task, the main challenge in this task is overfitting. While all algorithms are able to learn each module accurately, each module's loss must reach its minimum at the same time in order to achieve the global minimum of the total loss, otherwise the some modules will begin to overfit while others still underfit. The modular algorithms, M-Shrinkage and M-MAML, learn to adapt different modules at different rates and thus outperform their single-module counterparts. Among non-modular algorithms, the generalization loss of Shrinkage is better than Reptile, and matches MAML at its best adaptation step. Without proper regularization, all versions of MAML and Reptile eventually overfit, whereas Shrinkage and M-Shrinkage do not. The different behaviors of these algorithms are further illustrated by comparing the trajectories of parameter estimation error during task adaptation in Fig. 5 (Appendix D).

Nonlinear transform. In the second experiment, we explore a more realistic scenario where the data is generated by a nonlinear transformation, again with two modules. The transformation $\boldsymbol{\mu}_t(\boldsymbol{\theta}_t)$ is a spiral that rotates consecutive non-overlapping pairs of parameters by an angle proportional to their L_2 distance from the origin. Fig. 3b shows an example of this transform applied to a 2-D Gaussian.

Fig. 3b shows the performance of all algorithms on this second experiment. Due to the narrow valley in the optimization landscape caused by the spiral transform, all algorithms require hundreds of adaptation steps to minimize the loss. Again, the modular algorithms do well relative to their non-modular counterparts and to Reptile, which overfits badly. In our experiments, MAML was unstable when training with more than 100 inner loop steps. As a result, the best-performing hyperparameter value had a shorter adaptation horizon than required for this task, causing both MAML variants to perform worse. We also trained first-order MAML models to try to avoid this instability but these also underperformed (see Fig. 7, Appendix D). Overall, learning a modular prior for M-Shrinkage allows it to avoid overfitting and outperform the other methods in both of our experiments.

5 Conclusions

We showed that explicitly accounting for modularity is important for good performance in few-shot meta-learning. Our resulting algorithm has ties to MAML and contains Reptile as a special case, providing a new justification for its meta parameter update rule. Our analysis in the supplement highlights the importance of cross validation for meta-learning. In future work, we plan to extend our analysis to include more general models, and to apply our Shrinkage algorithm to more challenging domains, such as few-shot classification and reinforcement learning.

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135, 2017.

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

[3] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2013.

[4] Ferran Alet, Tomás Lozano-Pérez, and Leslie P Kaelbling. Modular meta-learning. In *Conference on Robot Learning*, 2018.

[5] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[6] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your MAML. In *International Conference on Learning Representations*, 2019.

[7] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Conference on Neural Information Processing Systems*, 2018.

[8] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE Computer Vision and Pattern Recognition*, 2019.

[9] Tianhe Yu, Chelsea Finn, Sudeep Dasari, Annie Xie, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. In *Robotics: Science and Systems*, 2018.

[10] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *International Conference on Learning Representations*, 2019.

[11] Yutian Chen, Yannis Assael, Brendan Shillingford, David Budden, Scott Reed, Heiga Zen, Quan Wang, Luis C. Cobo, Andrew Trask, Ben Laurie, Caglar Gulcehre, Aaron van den Oord, Oriol Vinyals, and Nando de Freitas. Sample efficient adaptive text-to-speech. In *International Conference on Learning Representations*, 2019.

[12] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Conference on Neural Information Processing Systems*, pages 4077–4087, 2017.

[13] Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Conference on Neural Information Processing Systems*, pages 3630–3638, 2016.

[14] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563, 2017.

[15] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, pages 1842–1850, 2016.

[16] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

[17] Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Learning to learn around a common mean. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10169–10179. 2018.

[18] Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Learning-to-learn stochastic gradient descent with biased regularization. In *International Conference on Machine Learning*, pages 1566–1575, 2019.

[19] Mikhail Khodak, Maria Florina-Balcan, and Ameet Talwalkar. Adaptive gradient-based meta-learning methods. *arXiv preprint arXiv:1906.02717*, 2019.

[20] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Conference on Neural Information Processing Systems*, pages 3981–3989, 2016.

[21] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[22] Yutian Chen, Matthew W Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P Lillicrap, Matt Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. In *International Conference on Machine Learning*, pages 748–756, 2017.

[23] Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. *International Conference on Learning Representations*, 2018.

[24] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical Bayes. In *International Conference on Learning Representations*, 2018.

[25] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Conference on Neural Information Processing Systems*, pages 7332–7342, 2018.

[26] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Conference on Neural Information Processing Systems*, pages 9516–9527, 2018.

[27] Sachin Ravi and Alex Beatson. Amortized Bayesian meta-learning. In *International Conference on Learning Representations*, 2019.

[28] Harrison Edwards and Amos Storkey. Towards a neural statistician. In *International Conference on Learning Representations*, 2017.

[29] Marta Garnelo, Dan Rosenbaum, Chris J Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J Rezende, and S.M. Ali Eslami. Conditional neural processes. *arXiv preprint arXiv:1807.01613*, 2018.

[30] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. *International Conference on Learning Representations*, 2019.

[31] Luisa M. Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, 2019.

[32] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019.

[33] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, 2018.

A Related work

Meta-learning refers to a class of algorithms where models are quickly adapted to new tasks with few training examples by leveraging knowledge acquired from a set of training tasks. Meta-learning has proven successful across a wide range of problems, including image classification [7, 8], robotics [9, 10], and speech synthesis [11], and many different approaches have been investigated.

Metric based methods learn a kernel for an embedding space, which is then used at test time to relate unseen examples to already seen data [12, 13]. Memory based methods [14–16] use external or internal memory architectures to store and leverage key training examples or history dependent information at test time. Optimization based methods instead modify the learning procedure, by learning good initialization [1, 2], regularization [17–19], or an optimizer [5, 20–22] to output the parameters of the learned model, allowing the learner to quickly and effectively adapt to new tasks.

Our work builds on this third class of approaches. A popular example of these, MAML, directly learns an initialization of the network parameters, from which it can adapt to a new task from the task distribution in only a very small number of gradient steps [1]. MAML requires expensive computation of second order derivatives and is hard to scale with the number of adaptation steps, subject to the short horizon bias [23]. Compared to works that emphasize fast adaptation and back-propagation of gradients through a limited number of consecutive gradient descent steps, our method optimizes task parameters toward the regularized optimum and back-propagates through the stationary point. We also show in this paper that Reptile [2], which avoids computing the second order derivative by updating the meta parameters towards the average of the optimized task parameters, can be derived as a special case of our framework.

Several hierarchical Bayesian approaches and a variety of inference methods have been proposed for meta learning. Grant et al. [24] introduced the first Bayesian variant of MAML using a Laplace approximation. Yoon et al. [25] and Finn et al. [26] introduced different probabilistic extensions to MAML, using approximate posteriors (either through an ensemble of particles or variational inference with a Gaussian posterior). Other methods have also proposed gradient based hierarchical Bayesian methods (e.g. Ravi and Beatson [27]) where posterior uncertainty is captured via a learned variational distribution that allows efficient test-time variational inference after a few gradient update steps. Other works (e.g. [28–30]) employ inference networks directly mapping from query set to variational latent variables without any gradient based optimization required at test time. These methods are more space and compute efficient although the capacity is limited by the size of the inference network. We also take a hierarchical Bayesian modeling approach and use the MAP estimate of task parameters for scalability. More sophisticated inference methods can be considered under our framework but we leave this for future work.

Related to our focus on modularity are a few meta-learning works that do not update all the parameters of the network at test time. Instead, they split them into task-specific and shared parameters [31], learn to produce network weights from task-specific embeddings [32], or learn a layer-wise subspace in which to do task specific gradient-based adaptation [33]. Our hierarchical Bayesian approach with shrinkage prior provides a flexible framework for incorporating prior knowledge about model structures and task similarities. Recent work also proposed a compositional approach to modular meta-learning [4], where at test time a search over module structure is performed with optional adaptation. Our method is complementary with theirs in that we learn how to optimally adapt or re-use modules at test time, which can be used within their adaptation phase.

Our method is also closely related to works on meta-regularization [17–19]. While meta-initialization aims at learning a good initialization for parameters of the network, meta-regularization aims at learning a regularization parameter to avoid over-fitting and improve stability. Our work differs from most works in this thread on the way of estimating the regularization strength σ_m^{-2} in a modular model.

B A simple Gaussian example

To illustrate the asymptotic behavior of different learning strategies as $T \to \infty$, we consider a simple model with $M = 1$ module, $D = D_1 = 1$ observation dimensionality, and normally-distributed observations,

Example (1D Normal observation)**.**

$$M = 1, D = 1, N_t \equiv N, \forall t = 1, \dots, T,$$
$$x_{t,n} \sim \mathcal{N}(x_{t,n}|\theta_t, 1), \forall t = 1, \dots, T, n = 1, \dots, N.$$

The task likelihood function is

$$p(\theta_{1:T}, \mathbf{x}_{1:T}|\phi, \sigma) = \prod_{t=1}^{T} \left(\mathcal{N}(\theta_t|\phi, \sigma^2) \prod_{n=1}^{N} \mathcal{N}(x_{t,n}|\theta_t, 1) \right). \tag{5}$$

We denote by ϕ_{True} and σ_{True} the true value of variable ϕ and σ in the following analysis.

B.1 Estimating all variables with MAP

We propose here to estimate the parameters $\theta_{1:T}, \phi, \sigma$ by maximizing $\log p(\theta_{1:T}, \phi|\mathcal{D}, \sigma)$. Since we are interested in the case when $T \to \infty$, we assign a flat prior to the parameter ϕ without affecting the conclusion at the asymptotic case. Maximizing this quantity is equivalent (up to an additive constant) to maximizing the function

$$\ell_{\text{MAP}}(\theta_{1:T}, \phi, \sigma) = -\frac{T}{2} \log \sigma^2 - \frac{1}{2} \sum_{t=1}^{T} \frac{(\theta_t - \phi)^2}{\sigma^2} - \frac{1}{2} \sum_{t=1}^{T} \sum_{i=1}^{N} (x_{t,n} - \theta_t)^2 . \tag{6}$$

Proposition 1 (Non-existence of the joint MAP estimate of (θ_t, ϕ, σ))**.** *ℓ_{MAP} does not have a global maximum and diverges to $+\infty$ as $\sigma \to 0^+$, $\forall \phi \in \mathbb{R}$ so that $\phi = \theta_t, \forall t = 1 \dots T$.*

The proof follows directly from the definition of the joint.

Proposition 2 (Consistency of MAP estimate of ϕ)**.** *For any $\sigma > 0$, estimate $\hat{\phi}$ at the joint MAP value of $(\phi, \theta_{1:T})$ is the sample mean across all tasks*

$$\hat{\phi} = \bar{x} \sim \mathcal{N}\left(\bar{x}|\phi_{\text{True}}, \frac{1}{T} \left(\sigma^2_{\text{True}} + \frac{1}{N} \right) \right) \to \delta[\hat{\phi} = \phi_{\text{True}}], \quad \text{as } T \to \infty$$

where $\bar{x} := \frac{1}{T} \sum_t \bar{x}_t := \frac{1}{NT} \sum_{n,t} x_{n,t}$, and δ denotes a delta distribution.

Proof of Proposition 2. We set all the partial derivatives of Eq. (6) with respect to ϕ and θ_t to zero. The equation $\partial_\phi \ell_{\text{MAP}} = 0$ gives

$$\hat{\phi} = \frac{1}{T} \sum_{t=1}^{T} \theta_t. \tag{7}$$

The equation $\partial_{\theta_t} \ell_{\text{MAP}} = 0$ gives

$$\hat{\theta}_t = \frac{\sum_{n=1}^{N} x_{t,n} + \phi/\sigma^2}{N + 1/\sigma^2}. \tag{8}$$

Now by summing Eq. (8) over $t = 1, ..., T$ and using Eq. (7) then for any $\sigma > 0$ we obtain

$$\hat{\phi} = \frac{1}{NT} \sum_{t=1}^{T} \sum_{n=1}^{N} x_{t,n} := \bar{x} \sim \mathcal{N}\left(\phi_{\text{True}}, \frac{1}{\sqrt{T}} (\sigma^2_{\text{True}} + \frac{1}{N}) \right). \tag{9}$$

Therefore, the MAP estimate of ϕ is an unbiased and consistent estimate of the ground truth value as $T \to \infty$ for any $\sigma > 0$. □

Proposition 3 (Estimation of σ by gradient ascent)**.** *Let ϕ and $\theta_{1:T}$ be at the MAP as a function of σ, and denote the sample variance of $\bar{x}_t$ across tasks with $S = \frac{\sum_{t=1}^{T} (\bar{x}_t - \bar{x})^2}{T}$. Maximizing the objective $\ell_{\text{MAP}}(\theta_{1:T}(\sigma), \phi(\sigma), \sigma)$ with respect to σ by gradient ascent will diverge at $\sigma \to 0^+$ if either of the following two conditions is satisfied*

1. $S < \frac{4}{N}$,

2. σ^2 *is initialized within* $\left(0, \frac{1}{2}\left(S - \frac{2}{N} - \sqrt{S(S - \frac{4}{N})}\right)\right)$.

Otherwise, it converges to a local maximum $\hat{\sigma}^2 = \frac{1}{2}\left(S - \frac{2}{N} + \sqrt{S(S - \frac{4}{N})}\right)$.

Corollary 1. *As the number of training tasks* $T \to \infty$*, condition 1 is equivalent to*

$$\sigma^2_{\text{True}} < \frac{3}{N}$$

and the initialization boundary of σ^2 *in condition 2 converges to*

$$\frac{1}{2}\left(\sigma^2_{\text{True}} - \frac{1}{N} - \sqrt{(\sigma^2_{\text{True}} + \frac{1}{N})(\sigma^2_{\text{True}} - \frac{3}{N})}\right),$$

beyond which the σ^2 *estimate converges at*

$$\hat{\sigma}^2 = \frac{1}{2}\left(\sigma^2_{\text{True}} - \frac{1}{N} + \sqrt{(\sigma^2_{\text{True}} + \frac{1}{N})(\sigma^2_{\text{True}} - \frac{3}{N})}\right). \tag{10}$$

Proof of Proposition 3 and Corollary 1. Plugging Eq. (9) into Eq. (8), we have

$$\hat{\theta}_t - \hat{\phi} = \frac{\bar{x}_t - \bar{x}}{1 + \frac{1}{N\sigma^2}}. \tag{11}$$

Setting the partial derivatives of Eq. (6) $\partial_{\sigma^2}\ell_{\text{MAP}} = 0$ gives

$$\hat{\sigma}^2 = \frac{\sum_{t=1}^{T}(\theta_t - \phi)^2}{T}. \tag{12}$$

Now by plugging Eq. (11) into this expression, we have

$$\hat{\sigma}^2 = \frac{\sum_{t=1}^{T}(\bar{x}_t - \bar{x})^2}{T(1 + \frac{1}{N\hat{\sigma}^2})^2}. \tag{13}$$

Hence we have a quadratic equation in σ^2 which is given by

$$\hat{\sigma}^4 + (2/N - S)\hat{\sigma}^2 + 1/N^2 = 0, \tag{14}$$

where $S = \frac{\sum_{t=1}^{T}(\bar{x}_t - \bar{x})^2}{T} \sim \frac{\sigma^2_{\text{True}} + \frac{1}{N}}{T}\chi^2_{T-1}$ with χ^2_{T-1} being a standard Chi-squared random variable with a degree of freedom $T - 1$.

Positive roots of Eq. (14) exist if and only if

$$S \geq \frac{4}{N}. \tag{15}$$

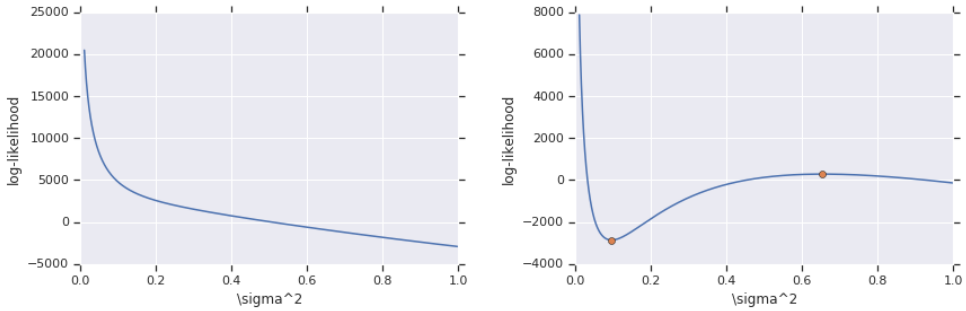
When $T \to \infty$, $S \to \sigma^2_{\text{True}} + \frac{1}{N}$, the condition above approaches

$$\sigma^2_{\text{True}} \geq \frac{3}{N}. \tag{16}$$

When the condition (15) or (16) does not hold, no stationary point exists and gradient ascent from any initialization will diverges toward $\hat{\sigma}^2 \to 0^+$. Figs. 4a and 4c illustrate the log-posterior and its gradient in that case.

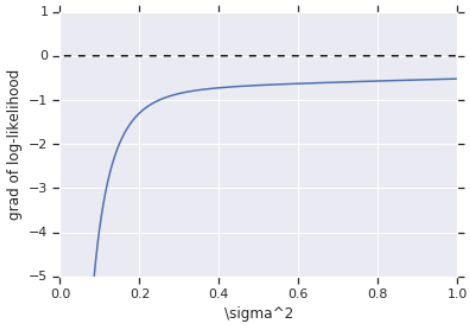
When the condition above is satisfied, there exist two (or one when the equality holds) roots at:

$$\sigma^2_{\text{root}} = \frac{1}{2}\left(S - \frac{2}{N} \pm \sqrt{S(S - \frac{4}{N})}\right), \tag{17}$$

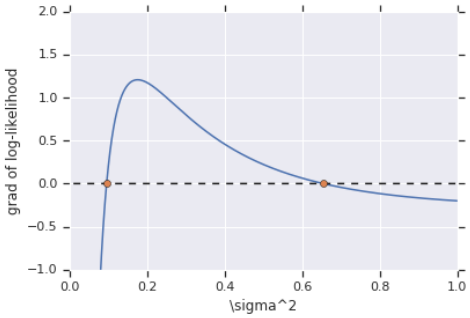


(a) $L(\sigma^2)$ when stationary points do not exist.

(b) $L(\sigma^2)$ when stationary points exist.



(c) $\partial_{\sigma^2} L$ when stationary points do not exist.



(d) $\partial_{\sigma^2} L$ when stationary points exist.

Figure 4: Example of the joint log-likelihood (up to a constant) and its gradient wrt σ^2

that is asymptotically

$$\sigma^2_{\text{root}} = \frac{1}{2}\left(\sigma^2_{\text{True}} - \frac{1}{N} \pm \sqrt{(\sigma^2_{\text{True}} + \frac{1}{N})(\sigma^2_{\text{True}} - \frac{3}{N})}\right), \quad T \to \infty. \tag{18}$$

By checking the sign of the gradient $\partial_{\sigma^2}\ell_{\text{MAP}}$ and plugging in Eq. (11), we can find that the left root is a local minimum and the right root is a local maximum. So if one follows gradient ascent to estimate σ^2, the optimization will diverge toward 0 when σ^2 is initialized below the left root, and converge to the second root otherwise. Figs. 4b and 4d illustrate the log-posterior ℓ_{MAP} as a function of σ^2 and its gradient when ϕ and θ_t are at their stationary point and condition 1 is satisfied. □

B.2 Estimating σ with predictive log-likelihood

Here we consider updating ϕ and $\theta_{1:T}$ following the same MAP estimate as above, but estimate σ with the approximate gradient of the log-posterior (Eq. (4)) on a set of independently sampled validation data $\mathbf{y}_{1:T}$ where $y_{t,n} \sim \mathcal{N}(x; \theta_t, 1), \forall t = 1, \ldots, T, n = 1, \ldots, K$.

Proposition 4. *Following the updating rules Eqs.* (2) *and* (4)*, as $T \to \infty$ the stationary point of ϕ and σ exists and satisfies*

$$\hat{\phi} = \phi_{\text{True}}, \quad \hat{\sigma}^2 = \sigma^2_{\text{True}}.$$

Proof. Following Eq. (4), the approximate gradient of the validation log-likelihood wrt $\log \sigma^2$ is

$$\begin{aligned}\frac{\partial \ell_{\text{PLL}}(\sigma^2)}{\partial \log \sigma^2} &\approx \sum_t \frac{\partial}{\partial \theta_t} \log p\left(\mathbf{y}_t | \theta_t\right)(\theta_t - \phi) \\ &= -K \sum_t (\theta_t - \bar{y}_t)(\theta_t - \phi) := g.\end{aligned} \tag{19}$$

By plugging the conditions at a stationary point Eqs. (8) and (9) we have

$$g = -\frac{KT}{(1+\frac{1}{N\sigma^2})^2}\frac{1}{T}\sum_t \left(\bar{x}_t - \bar{y}_t + \frac{1}{N\sigma^2}(\bar{x}-\bar{y}_t)\right)(\bar{x}_t - \bar{x}) \tag{20}$$

Following the generating process of $\mathbf{x}_t$ and $\mathbf{y}_t$, conditioned on ϕ and σ, the joint distribution of $\bar{x}_t$ and $\bar{y}_t$ with θ_t marginalized out is jointly normal and satisfies

$$[\bar{x}_t, \bar{y}_t]^T \sim \mathcal{N}\left(\phi \mathbf{1}_2, \begin{bmatrix} \sigma^2_{\text{True}} + \frac{1}{N} & \sigma^2_{\text{True}} \\ \sigma^2_{\text{True}} & \sigma^2_{\text{True}} + \frac{1}{K} \end{bmatrix}\right), \tag{21}$$

and $\bar{x}$ is the sample average of T conditionally independent random variables $\bar{\mathbf{x}}_{1:T}$.

At the limit of $T \to \infty$, by expanding the product in Eq. (20) and taking the average of various terms with their expectations, we have

$$g \xrightarrow{a.s.} \frac{m}{N(1+\frac{1}{N\sigma^2})^2}\left(\frac{\sigma^2_{\text{True}}}{\sigma^2} - 1\right). \tag{22}$$

We can find that at this limit, the update of $\log \sigma^2$ will converge to the true value $\hat{\sigma} = \sigma_{\text{True}}$. □

C Derivation of the approximate gradient of predictive log-likelihood

Lemma 1. *(Implicit differentiation) Let $\hat{\mathbf{y}}(\mathbf{x})$ be the stationary point of function $f(\mathbf{x}, \mathbf{y})$, that is, $\frac{\partial f(\mathbf{x},\mathbf{y})}{\partial \mathbf{y}}|_{\mathbf{y}=\hat{\mathbf{y}}(\mathbf{x})} = 0, \forall \mathbf{x}$ then the gradient of $\hat{\mathbf{y}}$ can be computed as*

$$\frac{\partial}{\partial \mathbf{x}}\hat{\mathbf{y}}(\mathbf{x}) = -\left(\frac{\partial^2 f}{\partial \mathbf{y}^2}\right)^{-1}\frac{\partial^2 f}{\partial \mathbf{y}\partial \mathbf{x}}. \tag{23}$$

By applying the chain rule, the derivative of the predictive log-likelihood Eq. (3) is given by

$$\frac{\partial \ell_{\text{PLL}}(\boldsymbol{\sigma})}{\partial \log \sigma_m^2} = \left(\sum_{t=1}^T \frac{\partial}{\partial \hat{\boldsymbol{\theta}}_t}\log p\left(\mathcal{D}_t^{\text{val}}|\hat{\boldsymbol{\theta}}_t\right)\frac{\partial \hat{\boldsymbol{\theta}}_t(\boldsymbol{\sigma}^{-2})}{\partial \sigma_m^{-2}}\right)\frac{\partial \sigma_m^{-2}}{\partial \log \sigma_m^2}. \tag{24}$$

To compute the derivative of $\hat{\boldsymbol{\theta}}_t(\boldsymbol{\sigma})$, denote the set of all the model parameters we take MAP estimate of as $\boldsymbol{\Theta} = [\boldsymbol{\phi}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_T]$, and apply Lemma 1 to the log-posterior on the training subset in Eq. (2), $\ell_{\text{MAP}}(\boldsymbol{\Theta}|\boldsymbol{\sigma}^{-2})$. We have

$$\frac{\partial \hat{\boldsymbol{\theta}}_t(\boldsymbol{\sigma}^{-2})}{\partial \boldsymbol{\sigma}^{-2}} = -\left(\frac{\partial^2 \ell_{\text{MAP}}}{\partial \boldsymbol{\Theta}^2}\right)^{-1}_{t,\cdot}\frac{\partial^2 \ell_{\text{MAP}}}{\partial \boldsymbol{\Theta}\partial \boldsymbol{\sigma}^{-2}}, \tag{25}$$

where $\mathbf{A}_{t,\cdot}$ denotes the t's block row of the matrix $\mathbf{A}$. We assume a normal distribution for the prior of $\boldsymbol{\phi}$, $p(\boldsymbol{\phi}) = \mathcal{N}(\boldsymbol{\phi}|\mathbf{0}, \delta^2\mathbf{I})$ and then the second order derivatives are given as

$$\frac{\partial^2 \ell_{\text{MAP}}}{\partial \boldsymbol{\Theta}^2} = \begin{bmatrix} -\boldsymbol{\Delta}^{-1} - T\boldsymbol{\Sigma}^{-1} & \boldsymbol{\Sigma}^{-1} & \ldots & \boldsymbol{\Sigma}^{-1} \\ \boldsymbol{\Sigma}^{-1} & -\mathbf{H}_1 - \boldsymbol{\Sigma}^{-1} & \ldots & \mathbf{0} \\ \vdots & \vdots & \ddots & \ldots \\ \boldsymbol{\Sigma}^{-1} & \mathbf{0} & \ldots & -\mathbf{H}_T - \boldsymbol{\Sigma}^{-1} \end{bmatrix}, \tag{26}$$

$$\frac{\partial^2 \ell_{\text{MAP}}}{\partial \boldsymbol{\Theta}\partial \boldsymbol{\sigma}^{-2}} = \left[\text{Diag}_m\left(\sum_t(\boldsymbol{\theta}_{m,t} - \boldsymbol{\phi}_m)\right), \text{Diag}_m\left(\boldsymbol{\phi}_m - \boldsymbol{\theta}_{m,1}\right), \ldots, \text{Diag}_m\left(\boldsymbol{\phi}_m - \boldsymbol{\theta}_{m,T}\right)\right]^T, \tag{27}$$

where $\boldsymbol{\Delta} = \text{Diag}_m(\delta_m^2\mathbf{I}_{D_m})$, $\boldsymbol{\Sigma} = \text{Diag}_m(\sigma_m^2\mathbf{I}_{D_m})$, $\mathbf{H}_t = -\frac{\partial^2}{\partial \boldsymbol{\theta}_t^2}\log p(\mathcal{D}_t^{\text{train}}|\boldsymbol{\theta}_t)$ is the negative Hessian matrix of task t's training log-likelihood, $\text{Diag}_m(\mathbf{A}_m)$ denotes an operator to convert a list of matrices $(\mathbf{A}_1, \ldots, \mathbf{A}_M)$ with index m to a block diagonal matrix with m's diagonal element being $\mathbf{A}_m$, $\mathbf{I}_d$ denotes an identity matrix of dimension d and D_m is the dimension of m's module.

It is impractical to invert the Hessian matrix except for a simple model with a few tasks. We first make a block diagonal approximation

$$\frac{\partial^2 \ell_{\text{MAP}}}{\partial \boldsymbol{\Theta}^2} \approx -\text{Diag}\left(\boldsymbol{\Delta}^{-1} + T\boldsymbol{\Sigma}^{-1}, \mathbf{H}_1 + \boldsymbol{\Sigma}^{-1}, \ldots, \mathbf{H}_T + \boldsymbol{\Sigma}^{-1}\right) \tag{28}$$

and plug that into Eq. (25), and further into Eq. (4)

$$\frac{\partial \ell_{\text{PLL}}(\boldsymbol{\sigma})}{\partial \log \sigma_m^2} \approx \sigma_m^{-2} \sum_t \frac{\partial}{\partial \boldsymbol{\theta}_t} \log p\left(\mathcal{D}_t^{\text{val}} | \hat{\boldsymbol{\theta}}_t(\boldsymbol{\sigma}^{-2})\right) \left(\mathbf{H}_t + \boldsymbol{\Sigma}^{-1}\right)^{-1}_{\cdot,m} (\boldsymbol{\theta}_{m,t} - \boldsymbol{\phi}_m),$$

where $(\mathbf{A})^{-1}_{\cdot,m}$ denotes the m-th column of the inverse matrix of $\mathbf{A}$. Unfortunately, the Hessian matrix $\mathbf{H}_t$ of $\log p(\mathcal{D}_t^{\text{train}} | \boldsymbol{\theta}_t)$ is still expensive to compute. We can either take a diagonal approximation

$$\frac{\partial \ell_{\text{PLL}}(\boldsymbol{\sigma})}{\partial \log \sigma_m^2} \approx \sigma_m^{-2} \sum_t \frac{\partial}{\partial \boldsymbol{\theta}_{m,t}} \log p\left(\mathcal{D}_{m,t}^{\text{val}} | \hat{\boldsymbol{\theta}}_t(\boldsymbol{\sigma}^{-2})\right) \left(\text{diag}\left((\mathbf{H}_t)_{m,m} + \sigma_m^{-2}\mathbf{I}\right)^{-1} (\boldsymbol{\theta}_{m,t} - \boldsymbol{\phi}_m)\right), \tag{29}$$

where $\text{diag}(\mathbf{A})$ is the diagonal matrix of $\mathbf{A}$, or if the prior influence from $\boldsymbol{\sigma}$ is a lot stronger than the task observation, i.e., $\sigma_m^{-2} \gg -\frac{\partial^2}{\partial \left(\theta_{m,t}^{(d)}\right)^2} \log p(\mathcal{D}_t^{\text{train}} | \boldsymbol{\theta}_t), \forall d, m, t$, we can ignore the second order derivative and further simplify as

$$\frac{\partial \ell_{\text{PLL}}(\boldsymbol{\sigma})}{\partial \log \sigma_m^2} \approx \sum_t (\boldsymbol{\theta}_{m,t} - \boldsymbol{\phi}_m)^T \frac{\partial}{\partial \boldsymbol{\theta}_{m,t}} \log p\left(\mathcal{D}_{m,t}^{\text{val}} | \hat{\boldsymbol{\theta}}_t(\boldsymbol{\sigma}^{-2})\right). \tag{30}$$

D Additional experiment details

We evaluate our proposed method Shrinkage, along with variants of MAML [1] and Reptile [2] on two synthetic few-shot meta-learning domains.

Recall from the main paper that the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\sigma}$ are fixed but unknown and different modules have different shrinkage parameters, σ_m. We use $\mathbf{I}_k$, $\mathbf{1}_k$ and $\mathbf{0}_k$ to denote the $k \times k$ identity matrix and the length-k vector of 1s and 0, respectively. We sample

$$\boldsymbol{\theta}_{m,t} \sim \mathcal{N}(\boldsymbol{\theta}_{m,t} | \boldsymbol{\phi}_m, \sigma_m^2 \mathbf{I}_{D_m})$$

and

$$\mathbf{x}_{t,n} \sim \mathcal{N}(\mathbf{x}_{t,n} | \boldsymbol{\mu}_t(\boldsymbol{\theta}_t), \Xi)$$

for each module m, task t, and n. The data distribution's mean is a function of $\boldsymbol{\theta}$ and varies across experiments. The observation noise variance $\Xi = \text{diag}(\boldsymbol{\xi}^2) = \text{diag}(\xi_1^2, \ldots, \xi_D^2)$ is a fixed and known diagonal matrix with dimensions $d \in \{1, \ldots, D\}$. The problem in each domain is to learn the parameters $\boldsymbol{\theta}_{\tilde{t}}$ of a new task $\mathcal{T}_{\tilde{t}}$ given a few observations $\{\mathbf{x}_{\tilde{t},n}\}$, where $N_t^{\text{train}} = N_t^{\text{val}}$ for all tasks. The main difference between the two evaluation domains is that $\boldsymbol{\mu}_t$ is a linear function of $\boldsymbol{\theta}_t$ in the first and is non-linear in the second.

Experiment 1: Linear transformation

Experiment 1 defines a simple, modular model in which the final observation dimension is the sum of the θ_m terms, in order to create a valley in the optimization landscape. We use the following parameters:

$$\begin{aligned} M =& 8, \\ D =& 9, \\ \boldsymbol{\phi} =& \mathbf{1}_M, \\ \boldsymbol{\sigma} =& [8, 8, 8, 8, 2, 2, 2, 2], \\ \boldsymbol{\xi} =& [8, 8, 8, 8, 5, 5, 5, 5, 1], \end{aligned}$$

with transformation

$$\boldsymbol{\mu}_t(\boldsymbol{\theta}_t) = [\mathbf{I}_M, \mathbf{1}_M/\sqrt{M}]^\top \boldsymbol{\theta}_t.$$

The two true modules are thus $\boldsymbol{\theta}_{1:4}$ and $\boldsymbol{\theta}_{5:8}$ but, in this experiment, we do not give the module structure to the algorithms, which treat each dimension as a separate module. This allows us to evaluate how well the algorithms can identify the module structure from data. The mean of the observations is thus $\boldsymbol{\theta}$ in the first M dimensions and $\sum_m \theta_m/\sqrt{M}$ in the final dimension. However, because $\boldsymbol{\xi}$ is small in the final dimension (i.e., ξ_M is small), the posterior of $\boldsymbol{\theta}$ is restricted to a small subspace near the $\sum_m \theta_m$ hyperplane. Gradient descent thus converges slowly regardless of the number of observations.

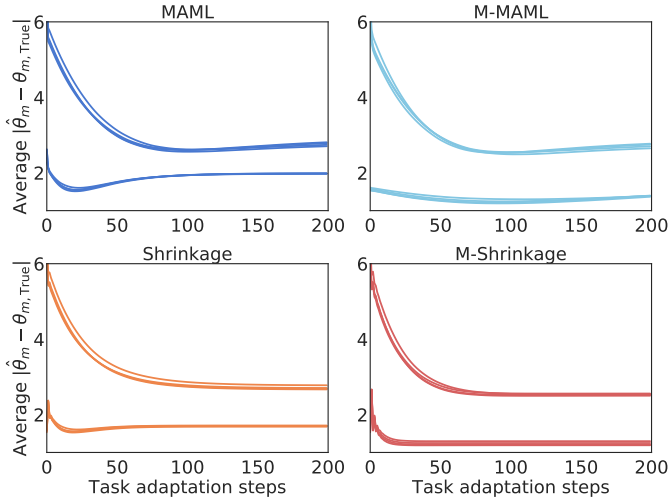


Figure 5: The absolute error for the linear transformation experiment between the estimated value of each parameter $\hat{\theta}_m$ and its true value versus the number of adaptation steps at meta-test time. Curves show the average behavior over 1000 test tasks. Each curve corresponds to one parameter dimension.

Fig. 5 shows the task adaptation behavior of each θ_m during meta-test for MAML, M-MAML, Shrinkage, and M-Shrinkage. The two-module structure is apparent in each subfigure, as dimensions with higher variance have higher average error. In MAML, the dimensions in each module reach their respective minima at different times, indicating that some dimensions are overfit while others remain underfit. Similarly, Shrinkage with a single σ is unable to properly fit each dimension, as some still overfit while others underfit. M-MAML solves MAML's adaptation-timing issues by learning step-sizes for each module so that each dimension begins to overfit at the same time; however, overfitting still occurs after this point. Finally, M-Shrinkage learns a separate σ_m for each module, allowing each dimension to be fit without overfitting.

Experiment 2: Nonlinear transformation

Experiment 2 defines a more realistic optimization scenario for meta-learning, where the data is generated by a nonlinear transformation of the task parameters. Specifically,

$$\begin{aligned} M =& 10, \\ D =& 10, \\ \boldsymbol{\sigma} =& [4, 4, 4, 4, 4, 4, 4, 4, 8, 8], \\ \boldsymbol{\phi} =& 2 \cdot \mathbf{1}_M, \\ \boldsymbol{\xi} =& 10 \cdot \mathbf{1}_D. \end{aligned}$$

The true modules are $\boldsymbol{\theta}_{1:8}$ and $\boldsymbol{\theta}_{9:10}$. The transformation $\boldsymbol{\mu}_t(\boldsymbol{\theta}_t)$ is a "swirl" effect that rotates non-overlapping pairs of consecutive parameters with an angle proportional to their L_2 distance from the origin. Specifically, each consecutive non-overlapping pair $(\mu_{t,d}, \mu_{t,d+1})$ is defined as

$$\begin{bmatrix} \mu_{t,d} \\ \mu_{t,d+1} \end{bmatrix} = \text{Rot}\left(\omega\sqrt{\theta_{t,d}^2 + \theta_{t,d+1}^2}\right) \cdot \begin{bmatrix} \theta_{t,d} \\ \theta_{t,d+1} \end{bmatrix}, \quad \text{for } d = 1, 3, ..., M-1,$$

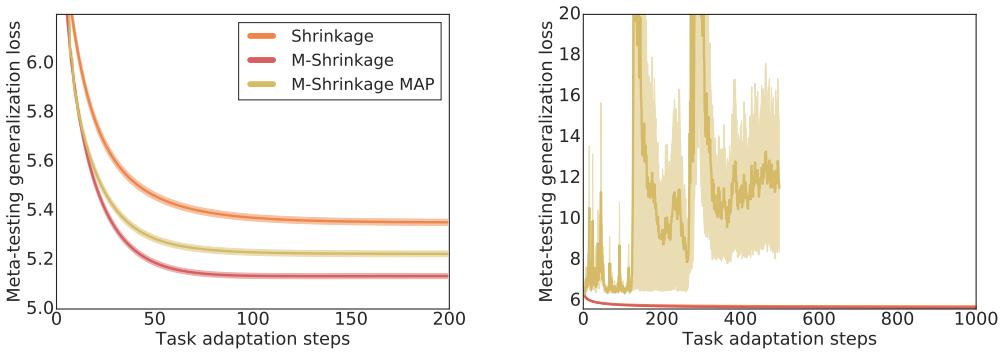
where

$$\mathrm{Rot}(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}, \tag{31}$$

and $\omega = \pi/5$ is the angular velocity of the rotation. This is a nonlinear volume-preserving mapping that forms a spiral in the observation space. Fig. 3b shows an example of this transform in 2-dimensions when applied to a Gaussian.

Results using MAP estimator for $\boldsymbol{\sigma}$

Fig. 6a and Fig. 6b show the test adaptation results from Experiments 1 and 2 for variants of our Shrinkage method. Specifically, these include results using the MAP estimator of $\boldsymbol{\sigma}$ (M-Shrinkage MAP) in place of the predictive likelihood (M-Shrinkage). In the first experiment, M-Shrinkage MAP simply underperforms the predictive likelihood. However, in the second experiment, the MAP estimate is extremely unstable and the learned value causes adaptation to diverge. These results provide empirical evidence for the failure of MAP estimation for $\boldsymbol{\sigma}$ detailed in Appendix B.1.



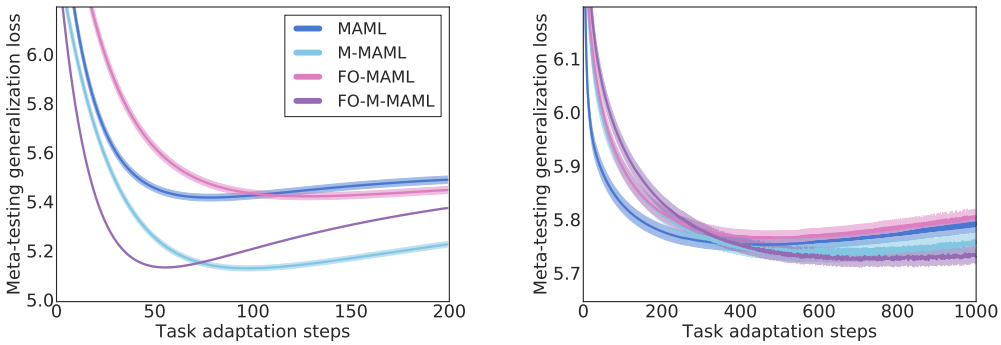
(a) Experiment 1: Linear transform

(b) Experiment 2: Nonlinear spiral transform

Figure 6: Mean and 68% credible interval of the generalization loss as a function of the number of adaptation steps for experiments 1 and 2.

Results comparing MAML with FOMAML

Fig. 7a and Fig. 7b show the test adaptation results from Experiments 1 and 2 for variants of MAML. To try to avoid the instabilities of MAML at large numbers of adaptation steps, we also evaluated its first-order variant, shown as FO-MAML, and a modular variant of first-order MAML (FO-M-MAML). Like our implementations of MAML and M-MAML, FO-MAML learns a single learning rate, and FO-M-MAML learns a learning rate per module. These are more stable during training but performance degrades after 65 inner loop steps. We suspect that this is because first-order MAML ignores the second derivative term and, after many adaptation steps, the direction of its gradient update for $\boldsymbol{\phi}$ becomes uncorrelated with the correct update direction. As such, the performance of MAML and first-order MAML are similar.



(a) Experiment 1: Linear transform

(b) Experiment 2: Nonlinear spiral transform

Figure 7: Mean and 68% credible interval of the generalization loss as a function of the number of adaptation steps for experiments 1 and 2.