How to distribute data across tasks for meta-learning?

Alexandru Cioba **Michael Bromberg** **Qian Wang** **Ritwik Niyogi** **Georgios Batzolis**

Jezabel Garcia **Da-shan Shiu** **Alberto Bernacchia**

MediaTek Research
Cambourne Business park, Cambridge UK

Abstract

Meta-learning models transfer the knowledge acquired from previous tasks to quickly learn new ones. They are trained on benchmarks with a fixed number of data points per task. This number is usually arbitrary and it is unknown how it affects performance at testing. Since labelling of data is expensive, finding the optimal allocation of labels across training tasks may reduce costs. Given a fixed budget of labels, should we use a small number of highly labelled tasks, or many tasks with few labels each? Should we allocate more labels to some tasks and less to others? We show that: 1) If tasks are homogeneous, there is a uniform optimal allocation, whereby all tasks get the same amount of data; 2) At fixed budget, there is a trade-off between number of tasks and number of data points per task, with a unique and constant optimum; 3) When trained separately, harder task should get more data, at the cost of a smaller number of tasks; 4) When training on a mixture of easy and hard tasks, more data should be allocated to easy tasks. Interestingly, Neuroscience experiments have shown that human visual skills also transfer better from easy tasks. We prove these results mathematically on mixed linear regression, and we show empirically that the same results hold for few-shot image classification on CIFAR-FS and mini-ImageNet. Our results provide guidance for allocating labels across tasks when collecting data for meta-learning.

1 Introduction

Deep learning (DL) models require a large amount of data in order to perform well, when trained from scratch, but labeling data is expensive and time consuming. An effective approach to avoid the costs of collecting and labeling a large amount of data is transfer learning: train a model on one big dataset, or a few related datasets that are already available, and then fine-tune the model on the target dataset, which can be of much smaller size [1]. In this context, there has been a recent surge of interest in the field of *meta-learning*, which is inspired by the ability of humans to *learn how to learn* [2]. A model is *meta-trained* on a large number of tasks, each characterized by a small dataset, and *meta-tested* on the target dataset.

The number of data points per task is usually set to an arbitrary number in standard meta-learning benchmarks. For example, in few-shot image classification benchmarks, such as *mini*-ImageNet [3], [4] and CIFAR-FS [5], each task has five classes (5-way) and either one or five images per class is used during testing (1-shot or 5-shots). During training, the number of data points per class is usually set to an arbitrary value, and it remains unclear how this number should be set to achieve the best testing performance. We focus on training, rather than testing data, because the former can be optimized by following specific procedures for data partitioning and collection.

Intuitively, one would think that the performance always improves with the number of training data points. However, if the total number of labels is limited, is it better to have a large number of tasks with little data in each task, or a smaller number of highly labelled tasks? Should some tasks be given

Preprint. Under review.

more labels than other tasks? The answers to these questions remain unknown, although they are important to inform the design of new meta-learning benchmarks and the application of meta-learning algorithms to real problems, especially given that data labelling is costly. Hence, we address these questions for the first time, for a specific meta-learning algorithm: MAML ([6]). Our contributions are:

- We introduce the problem of optimizing data allocation in meta-learning, with a fixed budget of total data points to distribute across training tasks. We show that, when tasks are homogeneous, the optimal solution is distributing data uniformly across tasks: all tasks get the same amount of data. This setting is considered in most meta-learning problems (section 4, Theorem 1).
- When data is distributed uniformly across tasks, we show that the trade-off between number of tasks and number of data points per task, at fixed budget, has a unique and constant optimum for large budgets (section 5, Theorems 2, 3, Figures 1, 2).
- Next, we consider the problem of two sets of tasks, easy and hard. When trained separately, we show that hard tasks need more data (per task) than easy tasks. While it is intuitive that hard tasks require more data for training, we emphasize that the total number of data points is fixed by the given budget, therefore the number of tasks is smaller (section 6.1, Figure 3).
- Finally, we study the problem of training a non-homogeneous mixture of easy and hard tasks. In contrast to when they are trained separately, we show that better performance is obtained by allocating more data to easy tasks. Our interpretation is that, as long as learning transfers from easy to hard tasks, it is better to train more on the former since they are easier to learn. Interestingly, human visual skills also transfer better from easy tasks [7] (section 6.2, Figure 4).

We prove results mathematically on mixed linear regression, and confirm those results empirically on few-shot image classification on CIFAR-FS and *mini*-ImageNet (code in the supplementary material).

2 Related Work

In the context of meta-learning and mixed linear regression, the work of [8] asks whether more tasks with a small amount of data can compensate for a lack of tasks with big data. However, they do not address the problem of finding the optimal allocation of data for a fixed budget, which is the main scope of our work. The work of [9] studies the problem of allocating a fixed budget of data points to a finite set of discrete distributions. In contrast to our work, they do not study the meta-learning problem and their data has no labels. Similar to us, a few theoretical studies looked at the problem of mixed linear regression in the context of meta-learning ([10], [11], [12], [13], [14], [15], [16]). However, none of these studies look into the problem of data allocation, which is our main focus.

An alternative approach to avoid labelling a large amount of data is *active learning*, where a model learns with fewer labels by accurately selecting which data to learn from [17]. In the context of meta-learning, the option of implementing active learning has been considered in a few recent studies [18], [19], [20], [21], [22]. However, they considered the active labeling of data within a given task, for the purpose of improving performance in that task only. Instead, we ask how data should be distributed across tasks.

In the context of recommender systems and text classification, a few studies considered whether labeling a data point, within a given task, may increase performance not only in that task but also in all other tasks. This problem has been referred to as *multi-task active learning* [23], [24], [25], [26], [27], or *multi-domain active learning* [28], [29]. However, none of these studies consider the problem of meta-learning with a fixed budget. A few studies have looked into actively choosing the next task in a sequence of tasks [30], [31], [32], [33], but they do not look at how to distribute data across tasks.

3 Meta-learning

The reader may refer to [2] for a general introduction to meta-learning with neural networks. In this work, we consider the cross-task setting, where we have a distribution of tasks $\tau \sim p(\tau)$ and a distribution of data points for a given task $\mathcal{D}^{\tau} \sim p(\mathcal{D}|\tau)$. Each task has a loss function $\mathcal{L}(\theta; \mathcal{D})$ that depends on a set of parameters θ and data $\mathcal{D}$. Here we assume that the loss has the same functional form across tasks (e.g. square loss if they are all regression tasks, cross-entropy if they are all classification tasks). The goal of meta-learning is minimizing the mean of the loss across tasks and data.

In the *meta-training* phase, m tasks $(\tau_i)_{i=1}^m$ are sampled from $p(\tau)$ and, for each task, n_i^t training data points $\mathcal{D}_i^t = (\mathbf{x}_{ij}^t, y_{ij}^t)_{j=1}^{n_i^t}$ and n_i^v validation data points $\mathcal{D}_i^v = (\mathbf{x}_{ij}^v, y_{ij}^v)_{j=1}^{n_i^v}$, are sampled independently from the same distribution $p(\mathcal{D}|\tau_i)$. We assume that the data is given by input $\mathbf{x}$ - label y pairs. The meta-training loss is a function of the data and the meta-parameters $\boldsymbol{\omega}$, is equal to

$$\mathcal{L}^{meta}\left(\boldsymbol{\omega}; \mathcal{D}^t, \mathcal{D}^v\right) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i^v}\sum_{j=1}^{n_i^v}\mathcal{L}\Big(\boldsymbol{\theta}(\boldsymbol{\omega}, \mathcal{D}_i^t); \mathbf{x}_{ij}^v, y_{ij}^v\Big) \tag{1}$$

The parameters are adapted to each task i by using the transformation $\theta(\boldsymbol{\omega}, \mathcal{D}_i^t)$. Different meta-learning algorithms correspond to a different choice of this transformation. Here we use MAML [6], which performs a fixed number of stochastic gradient descent steps with respect to the data for each task. With a single gradient step, that is equal to

$$\theta(\boldsymbol{\omega}, \mathcal{D}_i^t) = \boldsymbol{\omega} - \frac{\alpha_i}{n_i^t}\sum_{j=1}^{n_i^t}\nabla_{\boldsymbol{\omega}}\mathcal{L}\left(\boldsymbol{\omega}; \mathbf{x}_{ij}^t, y_{ij}^t\right) \tag{2}$$

where α_i is the learning rate for task i. This equation corresponds to a full-batch update, employing all the data for a given task, but mini-batch gradient updates can be performed as well. A number k of gradient steps may be used instead of one. This step is referred to as *inner loop* of meta-learning.

The loss in Eq.(1) is minimized with respect to the meta-parameters $\boldsymbol{\omega}$, namely

$$\boldsymbol{\omega}^{\star}\left(\mathcal{D}^t, \mathcal{D}^v\right) = \arg\min_{\boldsymbol{\omega}}\mathcal{L}^{meta}\left(\boldsymbol{\omega}; \mathcal{D}^t, \mathcal{D}^v\right) \tag{3}$$

This minimum is searched by stochastic gradient descent, using a distinct learning rate α_{meta}. At each gradient step, Eq.(2) is computed for each task and the gradient of Eq.(1) with respect to $\boldsymbol{\omega}$ is taken. This step is referred to as *outer loop* of meta-learning. Note that Eq.(1) includes all m tasks, which translates into full-batch training when taking the gradient. However, a mini-batch of tasks may be also drawn from the set of m tasks at each step of the optimization. Standard optimization procedures such as early stopping and scheduling of the learning rate α_{meta} can be applied. In the case of mixed linear regression (section 5), we solve Eq.(3) exactly by linear algebra.

In the *meta-testing* phase, the test loss $\mathcal{L}^{test}$ is computed using the optimal value $\boldsymbol{\omega}^{\star}$ and test datasets $\tilde{\mathcal{D}}^t, \tilde{\mathcal{D}}^v$

$$\mathcal{L}^{test}\left(\mathcal{D}^t, \mathcal{D}^v, \tilde{\mathcal{D}}^t, \tilde{\mathcal{D}}^v\right) = \mathcal{L}^{meta}\left(\boldsymbol{\omega}^{\star}\left(\mathcal{D}^t, \mathcal{D}^v\right); \tilde{\mathcal{D}}^t, \tilde{\mathcal{D}}^v\right) \tag{4}$$

The test datasets correspond to a new draw of both tasks and data points. The values of hyperparameters m, n^t, n^v, α, k for meta-testing are not necessarily the same as those used during meta-training. The main focus of this work is optimizing m, n_i^t, n_i^v for meta-training, while they are fixed during meta-testing. To evaluate the performance of the model for a given choice of the hyperparameters, we compute the average test loss, defined as

$$\overline{\mathcal{L}}^{test} = \mathop{\mathbb{E}}_{\mathcal{D}_t}\mathop{\mathbb{E}}_{\mathcal{D}_v}\mathop{\mathbb{E}}_{\tilde{\mathcal{D}}_t}\mathop{\mathbb{E}}_{\tilde{\mathcal{D}}_v}\mathcal{L}^{test} \tag{5}$$

4 The data allocation problem

We denote the number of data points per task i during meta-training as $N_i = n_i^t + n_i^v$, equal to the sum of training and validation data. In all experiments we used an equal split of training and validation, $n_i^t = n_i^v = n_i$. We assume that the total number of data points for meta-training, referred to as *budget*, is constant and equal to $b = \sum_{i=1}^m N_i = 2\sum_{i=1}^m n_i$. This is equal to the total number of data points across all training tasks, and is assumed fixed, while the number of data points per task

N_i are allowed to vary. We denote by $\mathbf{n}$ the vector of n_i values, $\mathbf{n} = (n_1, \ldots, n_m)$, and define the *data allocation* problem of finding the value of $\mathbf{n}$ such that the average test loss is minimized

$$\mathbf{n}^\star = \underset{\mathbf{n}\,:\,\sum_{i=1}^m n_i = b/2}{\arg\min}\ \overline{\mathcal{L}}^{test}(n_1, \ldots, n_m) \tag{6}$$

The optimal value $\mathbf{n}^\star$ is referred to as *optimal allocation*, it may depend on the budget and on other hyperparameters of the model. The optimal allocation determines which tasks should get more or less data, for a fixed budget b and number of tasks m. In the following theorem, we provide conditions under which the optimal data allocation is uniform.

Theorem 1. If the test loss $\overline{\mathcal{L}}^{test}$ is invariant under permutations of task allocations, i.e. permutations of its arguments $(n_1, \ldots, n_m)$ then the uniform allocation $\mathbf{n} = (n, \ldots, n)$ with $n = \frac{b}{2m}$ is a local extremum of the constrained optimization problem, provided that it is non-degenerate.

Furthermore, if

$$\overline{\mathcal{L}}^{test}\left(\frac{n_1+n_2}{2}, \frac{n_1+n_2}{2}, n_3, ..., n_m\right) \leq \mathcal{L}^{test}(n_1, ..., n_m), \tag{7}$$

for all $n_1, ..., n_m$, subject to $\sum_{i=1}^k n_i = \frac{b}{2}$, then the uniform allocation is the global minimum of the data allocation problem.

Proof. The proof of the first part (see Modern Purkiss principle) is given by [34], noting that the action of the symmetric group preserves the constraint and is irreducible, while the proof of the second part (global minimum) is given by [35]. □

Note that convexity of the test loss is a sufficient condition for the global minimum. We show in section 5.1 that the Purkiss principle applies to the case of mixed linear regression with homogeneous tasks. This result motivates, in addition to the data allocation problem (6), the study of the *uniform allocation* problem, in which the number of data points is assumed to be equal across tasks, but now the number of tasks m is allowed to vary. The solution of this problem is defined by

$$n^\star = \underset{n\,:\,nm=b/2}{\arg\min}\ \overline{\mathcal{L}}^{test}(n) \tag{8}$$

In this case, the question is whether to have more data and less tasks, or less data and more tasks, for the fixed budget b. In the next sections, we study both problems of data allocation and uniform allocation on mixed linear regression and few-shot image classification on CIFAR-FS and *mini*-ImageNet (Section 5.2).

4.1 Computation of the optimum

In the case of linear regression, we derive exact expressions for $\overline{\mathcal{L}}^{test}$ and $\mathbf{n}^\star$ in some limiting cases. In few-shot image classification, and in further linear regression experiments, we estimate $\overline{\mathcal{L}}^{test}$ empirically by searching a grid of values of $\mathbf{n}$. We average the test loss over multiple repetitions with different data samples and different initial conditions for $\boldsymbol{\omega}$. Then, we determine the mean and standard deviation for the optimum $\mathbf{n}^\star$ by the following procedure: we generate multiple instances of test loss/accuracy vs $\mathbf{n}$ by sampling uniformly from the repetitions at each value of $\mathbf{n}$, we record the optimal $\mathbf{n}^\star$ of each instance and construct a distribution of $\mathbf{n}^\star$ across all instances. We also provide nonlinear (sinusoid) regression experiments in the appendix (section B).

5 Solution of the uniform allocation

In this section we consider the problem of uniform allocation, while the non-uniform case is studied in section 6. We look at the trade-off between having either more tasks or more data per task, for a fixed budget, and we show that this problem has a unique optimum. We study this trade-off on two problems: mixed linear regression, where we compute a closed form expression for the optimum, and few-shot image classification, where we show empirical results.

5.1 Mixed linear regression

In mixed linear regression, each task is characterized by a different linear function, and the loss is the mean squared error:

$$\mathcal{L}\left(\boldsymbol{\theta};\mathbf{x},y\right)=\frac{1}{2}\left(y-\boldsymbol{\theta}^T\mathbf{x}\right)^2 \tag{9}$$

where the label y is a scalar, while the input $\mathbf{x}$ and the parameter $\boldsymbol{\theta}$ are vectors of p components. Each task corresponds to a different value of the generating parameter $\boldsymbol{\theta}$. Across tasks, that is distributed according to a Gaussian

$$\boldsymbol{\theta}\sim\mathcal{N}\left(\boldsymbol{\theta}_0,\frac{\nu^2}{p}I_p\right) \tag{10}$$

where $\boldsymbol{\theta}_0$, ν are hyperparameters, and I_p is the $p\times p$ identity matrix. The distribution of data for a given task is given by

$$y\mid\mathbf{x},\boldsymbol{\theta}\sim\mathcal{N}\left(\boldsymbol{\theta}^T\mathbf{x},\sigma^2\right) \tag{11}$$

$$\mathbf{x}\sim\mathcal{N}\left(\mathbf{0},\lambda^2 I_p\right) \tag{12}$$

where σ is the label noise and λ is the input variability. Each data point is independently drawn from this distribution, for either training or validation set. We distinguish between the case of *homogeneous* tasks, where all tasks have the same values of (σ,λ), and *non-homogeneous* tasks, where we allow those values to vary across tasks. In the following theorem, we compute an approximate expression for the average test loss for mixed linear regression.

Theorem 2. Consider the algorithm of section 3 (MAML one-step) and data generated according to the mixed linear regression model. Let $\sum_{i=1}^m n_i>p$ (underparameterized model), and let $n_i=n_i(\xi)$, $m=m(\xi)$ be any functions of order $\Theta(\xi)$ as $\xi\to\infty$. Then, the average test loss is equal to

$$\begin{aligned}\overline{\mathcal{L}}^{test}&=\frac{\sigma_r^2}{2}\left(1+\frac{\lambda_r^4\alpha_r^2 p}{n_r}\right)+\frac{\lambda_r^2 h_r\nu^2}{2}+\\&+\frac{\lambda_r^2 h_r p}{2}\left[\sum_{i=1}^m\lambda_i^2 h_i\right]^{-2}\sum_{i=1}^m\frac{\lambda_i^2}{n_i}\Bigg\{\sigma_i^2\left[h_i+\frac{\lambda_i^4\alpha_i^2}{n_i}\left[\left(n_i+1\right)g_{1i}+p\,g_{2i}\right]\right]+\\&+\frac{\nu^2}{p}\lambda_i^2\left[\left(n_i+1\right)g_{3i}+p\,g_{4i}\right]\Bigg\}+O\left(\xi^{-3}\right)\end{aligned} \tag{13}$$

where the subscript i denotes meta-training hyperparameters for task i, while the subscript r denotes meta-testing hyperparameters. We have defined the function $h_i=\left(1-\lambda_i^2\alpha_i\right)^2+\lambda_i^4\alpha_i^2\frac{p+1}{n_i}$, and the functions g are polynomials in $\lambda_i^2\alpha_i$ with coefficients of order $O(1)$, defined in the appendix, Equation (111).

Proof. The proof is given in the appendix, section C. It provides a generalization of the results of [10] in the case of non-homogeneous tasks and parametric input variability. □

When tasks are homogeneous ($\sigma_i=\sigma$, $\lambda_i=\lambda$) and a fixed learning rate is used for all meta-training tasks ($\alpha_i=\alpha$), we note that the test loss (13) is permutation invariant, thus the Purkiss principle of Theorem 1 applies. Therefore, in the remainder of this section we consider only the case of uniform allocation ($n_i=n$). Non-homogeneous tasks and non-uniform allocation are studied in section 6. Note also that Theorem 2 assumes an underparameterized model ($p<\sum_{i=1}^m n_i$). For completeness, we also study the overparameterized case in the appendix (section C.4).

Figure 1A plots the meta-test loss of mixed linear regression as a function of n for different budgets. It shows a good agreement between the experiments and the theoretical prediction of equation (13) (see appendix for details, section A.1). According to equation (13), the error between theory and experiment is expected to be of order $O\left(b^{-3/2}\right)$, since $b\sim O\left(\xi^2\right)$, indeed theoretical prediction is more accurate for larger budgets. As expected, test loss decreases with budget, since more data implies better performance. We emphasize that curves have a convex shape, implying that there is

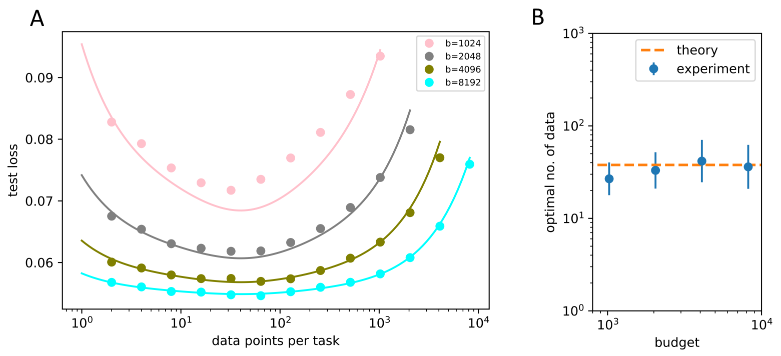

Figure 1: **The optimal number of data points per task is constant for large budgets: linear regression**. **A**: Test loss vs. number of data points per task at fixed budget (more data points imply less tasks). Dots: experimental values; Lines: theoretical prediction Eq.(13), different lines correspond to different budgets (legend). As predicted by Theorem 2, theoretical prediction is more accurate for larger budgets. Each curve has a unique optimum. **B**: Optimal number of data points per task vs. budget, the four points correspond to the four curves in panel A. The theoretical prediction of Eq.(14) (orange line) is close to the estimated experimental optimum (see section 4.1 for its computation).

a unique optimal value of n for each budget. While the curves tend to flatten at large budgets, the optimum remains approximately constant, as shown in Figure 1B. In the following theorem, we compute the unique solution of the uniform allocation problem for mixed linear regression.

Theorem 3. Under the assumptions of Theorem 2, consider the test loss of Equation (13) and the uniform allocation problem (8) of section 4. Furthermore, let $p = p(\xi)$ be a function of order $\Theta(\xi)$ as $\xi \to \infty$, neglect orders $O\left(\xi^{-2}\right)$ in Equation (13). Then, for all sufficiently small values of the learning rate α, the uniform allocation problem has a unique minimum, which does not depend on the budget, equal to the unique real and positive value of (for $k = 0, 1, 2$)

$$n^\star = -\frac{p}{3A}\left[B + \left(\frac{-1+\sqrt{-3}}{2}\right)^k \left(\frac{\Delta_1 + \sqrt{\Delta_1^2 - 4\Delta_0^3}}{2}\right)^{\frac{1}{3}} + \left(\frac{-1+\sqrt{-3}}{2}\right)^{-k} \left(\frac{2\Delta_0^3}{\Delta_1 + \sqrt{\Delta_1^2 - 4\Delta_0^3}}\right)^{\frac{1}{3}}\right] \tag{14}$$

where we have defined $\Delta_0 = B^2 - 3AC$, $\Delta_1 = 2B^3 - 9ABC + 27A^2D$, $A = \nu^2(1-\alpha')^6$, $B = 3\nu^2\alpha'^2(1-\alpha')^4$, $C = 2\alpha'^3\left[\nu^2(2-\alpha'-4\alpha'^2+3\alpha'^3) + \sigma'^2(2-5\alpha'+4\alpha'^2-\alpha'^3)\right]$, $D = 2\alpha'^4\left[\nu^2(2\alpha'^2-1) + \sigma'^2(2\alpha'-1)\right]$, $\alpha' = \lambda^2\alpha$, $\sigma' = \sigma/\lambda$.

Proof. The proof is provided in the appendix, section D. $\square$

This theorem implies that there is a unique and constant optimum for the number of data points per task at large budgets. While the theoretical optimum does not depend on the budget, it may depend on whether tasks are hard or easy (see section 6). Figure 1B shows the optimal $n^\star$ as a function of budget, it shows that the theoretical value of the optimum (orange line) agrees with the experiments.

5.2 Few-shot image classification

We next tested whether the results of mixed linear regression generalize to the more interesting problem of few-show image classification. In this case, the loss function is the cross-entropy, $\mathcal{L}(\theta; x, y) = -y^T \log\left(f_\theta(x)\right)$, where y is a one-hot encoding of the class label, and $f_\theta(x)$ is the output vector of a neural network with parameters θ and input x. We use a convolutional neural network commonly used with MAML on image classification [6] (see appendix A.2 for details).

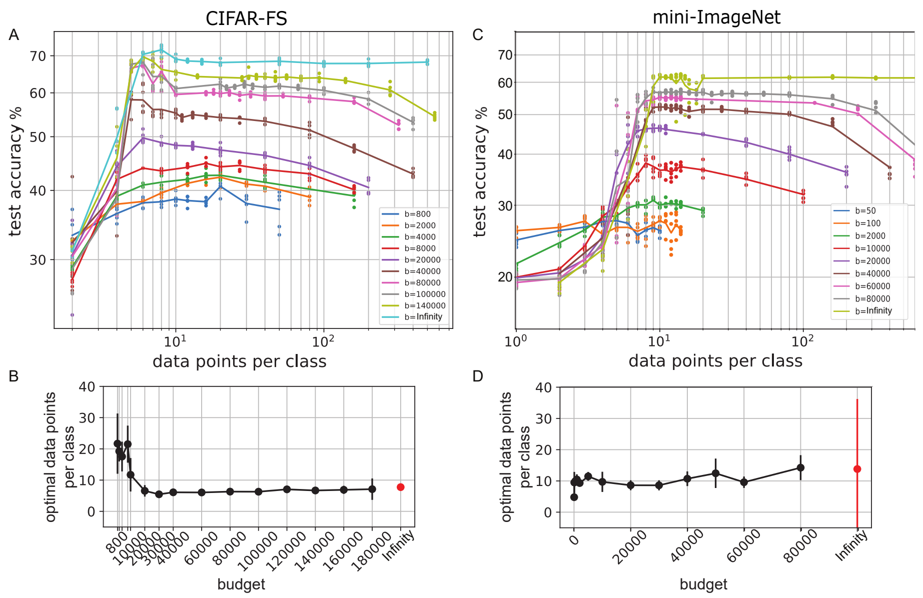

Figure 2: **The optimal number of data points per task is constant for large budgets: Few-shot image classification**: **A,B**: CIFAR-FS, **D,E**: *mini*-ImageNet dataset. Format is the same as of Figure 1. Curves are noisy and tend to flatten at large budgets, but there seems to be a unique optimum for each budget value. The optimum is computed empirically as explained in section 4.1. The optimal number of data points converges to ~ 7 for CIFAR-FS and to ~ 10 for *mini*-ImageNet. Error bars show standard deviation.

We investigate the CIFAR-FS [5] and *mini*-ImageNet [3] datasets, which are few-shot versions of CIFAR-100 and ImageNet, respectively. Both classification problems are 5-way: each task contains 5 classes. We refer to the number of data points *per class*, which has to be multiplied by 5 to find the number of data points *per task*. As in previous studies, we used 5 *shots* during testing (5 data points per class), while the number of shots during training depends on the data allocation.

In previous work [3], [5], we note that tasks are usually re-sampled indefinitely until convergence of the model, thus there is no limit on the number of tasks that can be generated. We instead pre-sample a set of tasks in order to fix the budget constraint. For comparison, we also run experiments in the usual way, and we call this the *infinite budget* case. However, the total number of labels is fixed and, even if tasks are re-sampled indefinitely, it does not imply that the amount of data is infinite, rather the same image may appear in multiple tasks.

As expected, Figure 2 shows that test performance improves with the budget, for both CIFAR-FS and *mini*-ImageNet (Figure 2A,C). For infinite budget, the accuracy is similar to previously reported values ($\sim 63\%$ for *mini*-ImageNet [6], $\sim 71\%$ for CIFAR-FS [5]). For CIFAR-FS, the optimal number of data points per class was ~ 20 at small budgets, but decreased and remained approximately constant at ~ 7 for large budgets (Figure 2B). For *mini*-ImageNet, the optimal number of data points per class was ~ 5 at very small budget and then increased and remain approximately constant at ~ 10 (Figure 2D). The performance curves in Figure 2A,C tend to flatten at higher budgets, but the optimum does not change significantly. Overall, the empirical study of both datasets confirms our prediction that the optimal number of data points per task is constant at large budgets.

6 Easy vs hard tasks

In this section we consider the case of non-homogeneous tasks. We distinguish between two sets of tasks, easy and hard. We use two independent definitions of hard tasks, one affects the input and the other affects the output (label) of a dataset. We apply this definition in a similar way to both mixed linear regression and few-shot image classification.

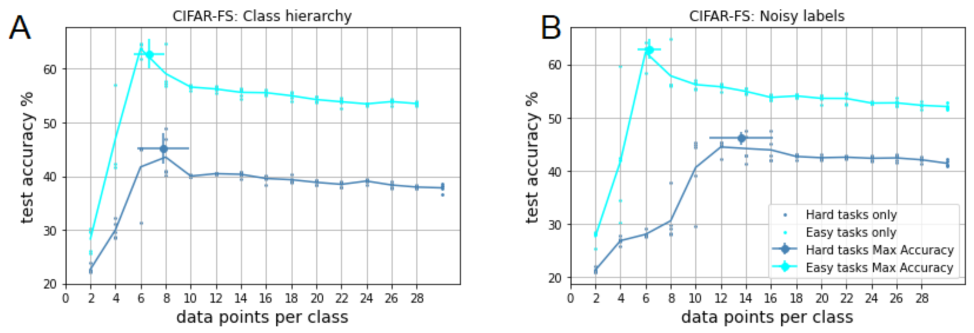

Figure 3: **Hard tasks prefer more data (and less tasks) when trained separately**. Few shot image classification on CIFAR-FS. **A** Tasks are made harder by drawing classes within a hierarchy; **B** Tasks are made harder by adding label noise. Both plots show test accuracy versus the number of data points per class, as in Figure 2A. Each plot shows an estimate of the point of maximum accuracy, with error bars showing standard deviation (see section 4.1 for its computation). In both cases, performance is lower and the optimal number of data points per class is larger for hard tasks.

For the problem of mixed linear regression, we define *task difficulty* in terms of the hyperparameters σ and λ. A task is harder if it has a larger σ (at equal λ) or smaller λ (at equal σ). The case of larger σ is intuitive, a task is harder to learn if labels are more corrupted by noise. In the case of smaller λ, the smaller input range makes it harder to solve the regression problem in presence of noise.

In few-shot image classification, the first method to make a task harder is to introduce label noise [36]: each input image has 20% probability of having its label swapped with another random class. The second method is similar to [15]: we take advantage of the hierarchical tree of the CIFAR-100 dataset and we constraint each task to draw classes from one of three superclasses: 1) animals, 2) vegetations, 3) object and scenes. Therefore, we assume that each task has a smaller variability of its input, not in terms of pixels color or intensity, but in terms of semantic relations. Intuitively, it is harder to distinguish inputs when they are more similar to each other. We refer to the two different definitions as, respectively, *noisy labels* and *class hierarchy*.

6.1 Separate training

Before studying the training of a mixture of easy and hard tasks, we ask what is the optimal uniform allocation when the two types of tasks are trained separately. In mixed linear regression, the expression for the optimum of the uniform allocation $n^\star$ is given by Eq.(14), but is hard to evaluate how it depends on σ and λ. Therefore we computed an approximation that holds for small α' (see equation (136) in the appendix):

$$n^\star = \left[2\left(1+\frac{\sigma'^2}{\nu^2}\right)\right]^{\frac{1}{3}} \alpha'^{\frac{4}{3}} p + O\left(\alpha'^{\frac{5}{3}}\right) \tag{15}$$

where $\alpha' = \lambda^2\alpha$ and $\sigma' = \sigma/\lambda$. The optimum increases with σ, suggesting that harder tasks require more data (and less tasks) at fixed budget. For λ, there are two opposing forces: 1) On one hand a smaller λ is equivalent to amplifying output noise σ' and increasing the optimum $n^\star$; 2) On the other hand, λ rescales the learning rate α' with the opposite and stronger effect that a smaller λ decreases the optimum.

Figure 3 shows that introducing task difficulty in few-shot image classification on CIFAR-FS increases the optimum for both methods (A: class hierarchy; B: noisy labels, see section A.3 for details). As expected, performance is lower for hard tasks in both cases (note that we train and test on the same set of tasks, either only easy or only hard). While it is intuitive that hard tasks require more data to learn, we emphasize that, for a fixed budget, this comes at the expense of a smaller number of tasks.

6.2 Joint training

We now turn to the problem of training on a mixture of easy and hard tasks. In addition to a fixed budget, we further assume an equal number of easy and hard tasks, and a constant sum of easy and hard data points per task. Therefore, the only hyperparameter of interest is the relative number of data points per task for easy vs hard. Note that we use a mixture of easy and hard tasks also for testing, but we always use an equal number of easy and hard data points and tasks in that case (see section A.3 for details).

After the results of section 6.1, we expect better results when allocating more data to hard tasks. Surprisingly we find that the opposite is true. Figure 4 shows that a slightly higher performance is obtained when allocating more data to easy tasks, in few-shot image classification on CIFAR-FS for both methods (Panel A: class hierarchy; Panel B: noisy labels). Intuitively, easy tasks are easier to learn than hard tasks. Therefore, it may be that if training on easy tasks transfers to better performance on hard tasks, then it is better to allocate more data to easy tasks.

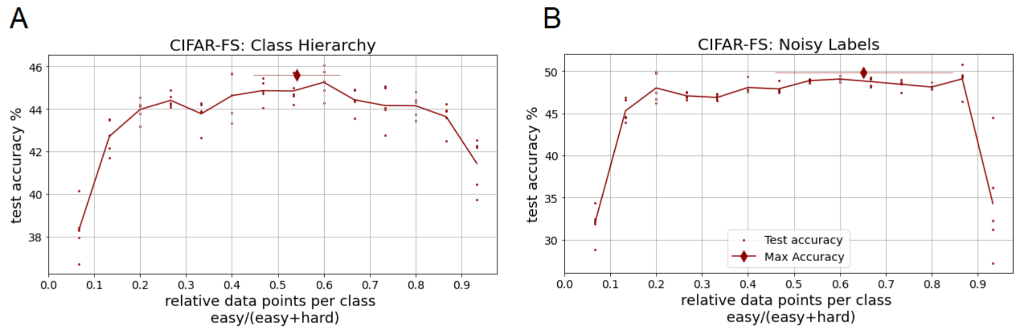

Figure 4: **When training on a mixture of easy and hard tasks, it is better to allocate more data to easy tasks**. Few shot image classification on CIFAR-FS. **A** Tasks are made harder by drawing classes within a hierarchy; **B** Tasks are made harder by adding label noise. Both plots show test accuracy versus the relative amount of easy vs hard data points per class. Each plot shows an estimate of the point of maximum accuracy, with error bars showing standard deviation (see section 4.1 for its computation). In both cases, a slightly higher performance is obtained by allocating more data to easy tasks than to hard ones.

7 Discussion

In this paper we analysed the problem of optimal data allocation in meta-learning when the budget of labelled examples is limited. When tasks are homogeneous, we showed that uniform data allocation across tasks is optimal (under the assumptions of Theorem 1). We further studied whether one should use less tasks with more data or more tasks and less data. For mixed linear regression, we proved that the optimum is unique and constant for large budgets. We confirmed this finding empirically on few-shot image classification (an example of nonlinear regression is also included in the appendix).

In the case of non-homogeneous tasks, with a mixture of easy and hard tasks, we showed how to optimally allocate data between the two types of tasks. In particular, we found that it is better to allocate more data to easy tasks. This result echoes findings in experimental neuroscience, where it was found that human visual skills indeed transfer better from easy tasks than from hard ones [7]. Our findings provide a guideline for collecting meta-learning data in a way that achieves the best performance under a fixed budget. We do not expect our study to have a negative societal impact, at least not in a direct way.

Overall, our study exemplifies the importance of optimal data allocation in meta-learning and gives a series of empirical and theoretical insights on the relation between model performance and data allocation for MAML. While the behaviour of other meta-learners need not be the same, we surmise that the problem of training models close to optimal allocation is important, and leave much space for empirical study in a variety of contexts, as well as for the development of a more general theoretical framework. For example, we have only scratched the surface of the problem of non-uniform allocation, which requires much further study.

References

[1] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *ICML*, page 9, 2014.

[2] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-Learning in Neural Networks: A Survey. *arXiv:2004.05439 [cs, stat]*, April 2020. arXiv: 2004.05439.

[3] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. *arXiv:1606.04080 [cs, stat]*, December 2017. arXiv: 1606.04080.

[4] Sachin Ravi and Hugo Larochelle. OPTIMIZATION AS A MODEL FOR FEW-SHOT LEARNING. *ICLR*, page 11, 2017.

[5] Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv:1805.08136 [cs, stat]*, July 2019. arXiv: 1805.08136.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *arXiv:1703.03400 [cs]*, March 2017. arXiv: 1703.03400.

[7] Merav Ahissar and Shaul Hochstein. Task difficulty and the specificity of perceptual learning. *Nature*, 387(6631):401–406, May 1997.

[8] Weihao Kong, Raghav Somani, Zhao Song, Sham Kakade, and Sewoong Oh. Meta-learning for mixed linear regression. *arXiv:2002.08936 [cs, stat]*, February 2020. arXiv: 2002.08936.

[9] Shubhanshu Shekhar, Tara Javidi, and Mohammad Ghavamzadeh. Adaptive Sampling for Estimating Probability Distributions. *ICML*, page 10, 2020.

[10] Alberto Bernacchia. Meta-learning with negative learning rates. *arXiv:2102.00940 [cs]*, March 2021. arXiv: 2102.00940.

[11] 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. Curran Associates, Inc., 2018.

[12] Yu Bai, Minshuo Chen, Pan Zhou, Tuo Zhao, Jason D. Lee, Sham Kakade, Huan Wang, and Caiming Xiong. How Important is the Train-Validation Split in Meta-Learning? *arXiv:2010.05843 [cs, stat]*, February 2021. arXiv: 2010.05843.

[13] Nilesh Tripuraneni, Chi Jin, and Michael I. Jordan. Provable Meta-Learning of Linear Representations. *arXiv:2002.11684 [cs, stat]*, February 2020. arXiv: 2002.11684.

[14] Simon S. Du, Wei Hu, Sham M. Kakade, Jason D. Lee, and Qi Lei. Few-Shot Learning via Learning the Representation, Provably. *arXiv:2002.09434 [cs, math, stat]*, February 2020. arXiv: 2002.09434.

[15] Liam Collins, Aryan Mokhtari, and Sanjay Shakkottai. Why Does MAML Outperform ERM? An Optimization Perspective. *arXiv:2010.14672 [cs, math, stat]*, December 2020. arXiv: 2010.14672.

[16] Katelyn Gao and Ozan Sener. Modeling and Optimization Trade-off in Meta-learning. *arXiv:2010.12916 [cs, math, stat]*, October 2020. arXiv: 2010.12916.

[17] Burr Settles. Active Learning Literature Survey. *Technical Report*, page 67, 2010.

[18] Philip Bachman, Alessandro Sordoni, and Adam Trischler. Learning Algorithms for Active Learning. *arXiv:1708.00088 [cs]*, July 2017. arXiv: 1708.00088.

[19] Victor Garcia and Joan Bruna. Few-Shot Learning with Graph Neural Networks. *arXiv:1711.04043 [cs, stat]*, February 2018. arXiv: 1711.04043.

[20] Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian Model-Agnostic Meta-Learning. *arXiv:1806.03836 [cs, stat]*, November 2018. arXiv: 1806.03836.

[21] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic Model-Agnostic Meta-Learning. *arXiv:1806.02817 [cs, stat]*, October 2019. arXiv: 1806.02817.

[22] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E. Turner. Fast and Flexible Multi-Task Classification Using Conditional Neural Adaptive Processes. *arXiv:1906.07697 [cs, stat]*, January 2020. arXiv: 1906.07697.

[23] Roi Reichart, Katrin Tomanek, Udo Hahn, and Ari Rappoport. Multi-Task Active Learning for Linguistic Annotations. *ACL*, page 9, 2008.

[24] Yi Zhang. Multi-Task Active Learning with Output Constraints. *AAAI*, page 6, 2010.

[25] Avishek Saha, Piyush Rai, Hal Daume Iii, and Suresh Venkatasubramanian. Online Learning of Multiple Tasks and Their Relationships. *AISTATS*, page 9, 2011.

[26] Abhay Harpale. Multi-Task Active Learning. *PhD thesis*, page 124, 2012.

[27] Meng Fang, Jie Yin, Lawrence O. Hall, and Dacheng Tao. Active Multitask Learning With Trace Norm Regularization Based on Excess Risk. *IEEE Transactions on Cybernetics*, 47(11):3906–3915, November 2017.

[28] Lianghao Li, Xiaoming Jin, Sinno Jialin Pan, and Jian-Tao Sun. Multi-domain active learning for text classification. *KDD*, page 9, 2012.

[29] Zihan Zhang, Xiaoming Jin, Lianghao Li, Guiguang Ding, and Qiang Yang. Multi-Domain Active Learning for Recommendation. *AAAI*, page 7, 2016.

[30] Paul Ruvolo and Eric Eaton. Active Task Selection for Lifelong Machine Learning. *AAAI*, page 7, 2013.

[31] Anastasia Pentina, Viktoriia Sharmanska, and Christoph H. Lampert. Curriculum learning of multiple tasks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5492–5500, Boston, MA, USA, June 2015. IEEE.

[32] Anastasia Pentina and Christoph H. Lampert. Multi-Task Learning with Labeled and Unlabeled Tasks. *arXiv:1602.06518 [cs, stat]*, June 2017. arXiv: 1602.06518 version: 2.

[33] Gan Sun, Yang Cong, and Xiaowei Xu. Active Lifelong Learning with “Watchdog”. *AAAI*, page 8, 2018.

[34] William C. Waterhouse. Do Symmetric Problems Have Symmetric Solutions? *The American Mathematical Monthly*, 90(6):378–387, 1983. Publisher: Mathematical Association of America.

[35] J Keilson. On global extrema for a class of symmetric functions. *Journal of Mathematical Analysis and Applications*, 18(2):218–228, May 1967.

[36] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from Noisy Labels with Deep Neural Networks: A Survey. *arXiv:2007.08199 [cs, stat]*, April 2021. arXiv: 2007.08199.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, December 2014. arXiv: 1412.6980.

A Details of experiments

A.1 Mixed linear regression experiments

For the distributions of Eqs.(10),(11), we used the following values of parameters: $\sigma = 0.2$, $\nu = 0.2$, $p = 128$, $\boldsymbol{\theta}_0 = (0.05, \dots, 0.05)$. During meta-training, we run experiments for several pairs of values of (N, b), as shown in Figure 1. As explained in the main text, the number of training and validation data points was equal, $n_t = n_v = n$, and the total number of data points per task was defined as $N = n_t + n_v = 2n$. We used an inner loop learning rate of $\alpha = 0.3$ during training. We computed the exact minimum $\boldsymbol{\omega}^\star$ by using standard linear algebra, since the loss in Eq.(1) is convex and quadratic.

During meta-testing, we estimated the test loss by averaging over 100 tasks, for each task we used 20 training and 50 validation data points and we used a inner loop learning rate of 0.3. We repeated each experiment 100 times, with different generated data samples, for each pair of values of (N, b). As explained in the main text, we used sampling to determine the optimum of the test loss: for each value of the budget b, we sampled 1000 curves for the test loss, by sampling one out of the 100 repetitions for each value of the number of data points per task N. For each one of the 1000 curves, we determined the empirical value of the minimum $N^\star$ of the test loss, and we computed the mean and standard deviation of $N^\star$ across those 1000 values (shown in Figure 1B,E).

In addition to the empirical experiments on mixed linear regression, that are conducted by sampling the generative model of Eqs.(10),(11), we used the analytical formulas for the average test loss and the optimal data allocation computed in section C and D, respectively.

A.2 Image classification experiments

For the CIFAR-FS and *mini*-ImageNet we apply MAML on a base learner given by a convolutional neural network (CNN) with architecture as described in [6]. Small modifications were made to run on the CIFAR-FS data. We use a network with 4 convolutional blocks. Each block consisted of a sequence of 2D convolutions with kernel size 3, stride 1, same padding, batch normalization, a ReLU nonlinearity, and MaxPooling to half size with the kernel size and the stride of 2. The predictor head is a softmax layer applied to the flattened resulting features. We used 64 filters per layer for CIFAR-FS and 32 for *mini*-ImageNet, in order to reduce overfitting, as in [6]. CIFAR-FS and *mini*-ImageNet image sizes were 32 $\times$ 32 and 84 $\times$ 84, respectively.

For each budget limited run of MAML, data was presampled and arranged into the required number of tasks and data points per class. Whether from the training, testing or validation meta-datasets, a task was sampled by selecting 5 random classes with replacement from the available pool. Thus, it is possible for independent samples of tasks to occasionally contain the same class of images.

For the grid search algorithm, independent runs were performed by independently sampling the initial conditions of the algorithm and also independently sampling the meta-training, meta-validation and meta-test datasets. We performed 5 independent runs this way for each point on the grid. For the meta-test datasets, 1000 tasks were sampled to minimize variance in the estimates. Each task consists of 5 randomly sampled classes (5-way), 5 data points (5-shot) per class for model adaptation and 5 data points per class for model evaluation.

During meta-training, the train-test split for each meta-training task dataset was 0.5. We ran the Adam Algorithm in the meta-update of the MAML parameter ω in the outer loop with the initial learning rate of 0.001. We annealed this learning rate on a plateau of the validation loss for at least 250 meta-updates. This is done four times with the fifth incurring termination of training. For the adaptation step we ran 5 steps of gradient descent during meta-training and 1 step during meta-testing. Our preliminary experimental results demonstrate that using 5 adaptation steps for meta-training gives more stable performance than using only 1 adaptation step although no performance improvement can be obtained by using more adaptation steps. 1 adaptation step was used during training for *mini*-Imagenet experiments to accommodate the larger memory requirements associated with this dataset. The inner loop learning (i.e. adaptation step size) is set to 0.01 in all experiments.

The mini-batch gradient descent is used during meta-training for efficient GPU memory use. Each mini-batch consists of 25 tasks for the experiments with $m \geq 25$ and otherwise m tasks, and each task consists of 50 data points for experiments with $N \geq 50$ and otherwise N data points. We performed preliminary experiments and no significant difference in terms of performance was observed when increasing the batch size further.

All experiments ran on a single Nvidia 2080 GPU with an average runtime of 1.5 hours per run of the CIFAR-FS dataset and 5 hours for miniImageNet. These times depend on the exact hyperparameters such as the allocation and budget. Overall, grid search experiments were distributed in parallel over 15 - 45 GPUs.

A.3 Easy and hard tasks creation in the image classification experiments

In section 6 we introduce hard tasks to test the influence of task difficulty and study the optimal allocation when training only in easy tasks, only in hard tasks a mixture of hard and easy tasks. The experimental set-up follows the description in section A.2. We define easy task as the standard tasks of CIFAR-FS, while we use two definitions of hard task:

- *Class hierarchy*. Task difficulty is represented by the similarity in the input images. The CIFAR-FS dataset can be described by a 4-level hierarchy. At the top level, there are three high-level categories: *animals*, *vegetations*, and *objects and scenes* (see Table 1). In a hard task all classes belong to the same high-level category.
- *Noisy labels* Task difficulty is represented by label noise [36]. In hard tasks, each image has a 20% probability of being mislabeled, by randomly drawing a label from another class with a uniform distribution.

The train, test and validation split in hard and easy tasks is the original CIFAR-FS split. In the experiments with only hard tasks, the procedure is exactly the same as described in section A.2 but only employing hard tasks for both training and testing. In joint training with easy and hard tasks, we keep the budget constant at $50,000$ data points. The number of easy tasks is also kept constant, at 333, and equal to the number of hard tasks. The sum of easy and hard data points per class is kept constant and equal to 30 data points per class. The balance of easy and hard tasks varies between 2 and 28 data points per class.

Table 1: CIFAR-FS, 4-level Hierarchy

3 Class	10 Class	20 Class	100 Class
animals	large animals	reptiles	crocodile, dinosaur, lizard, snake, turtle
		large carnivores	bear, leopard, lion, tiger, wolf
		large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
	medium animals	aquatic mammals	beaver, dolphin, otter, seal, whale
		medium-sized mammals	fox, porcupine, possum, raccoon, skunk
	small animals	small mammals	hamster, mouse, rabbit, shrew, squirrel
		fish	aquarium fish, flatfish, ray, shark, trout
	invertebrates	insects	bee, beetle, butterfly, caterpillar, cockroach
		non-insect	crab, lobster, snail, spider, worm
	people	people	baby, boy, girl, man, woman
vegetations	vegetations	flowers	orchids, poppies, roses, sunflowers, tulips
		fruit and vegetables	apples, mushrooms, oranges, pears, peppers
		trees	maple, oak, palm, pine, willow
objects and scenes	household objects	food containers	bottles, bowls, cans, cups, plates
		household electrical devices	clock, keyboard, lamp, telephone, television
		household furniture	bed, chair, couch, table, wardrobe
	construction	large man-made outdoor things	bridge, castle, house, road, skyscraper
	natural scenes	large natural outdoor scenes	cloud, forest, mountain, plain, sea
	vehicles	vehicles 1	bicycle, bus, motorcycle, pickup truck, train
		vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

B Nonlinear regression

We investigated non-linear regression on a 1-dimensional sinusoid wave dataset. The loss is again the mean squared error:

$$\mathcal{L}\left(\theta; x, y\right) = \frac{1}{2}\left(y - f_\theta(x)\right)^2 \tag{16}$$

where $f_\theta(x)$ is the output of a neural network with parameters θ and input x. We use a simple Multi-Layer Perceptron, following the architecture used in MAML [6], see below for details on this experiment.

The task distribution is a joint over 2 parameters $\tau = (A, \phi)$, where both distributions are uniform within their range, given by

$$A \sim \mathcal{U}\,(0.1, 5) \qquad\qquad \phi \sim \mathcal{U}\,(0, \pi) \tag{17}$$

The distribution of data for a given task $\tau = (A, \phi)$ is given by

$$x \sim \mathcal{U}\,(-5, 5) \qquad\qquad y = A \cdot \sin(x + \phi) \tag{18}$$

Each data point, of either training or validation sets, is independently drawn from this distribution. We don't add any label noise, therefore the minimum achievable value of the test loss is zero.

Figure 5 shows the data allocation results for the sinusoid regression, the format is the same as in Figure 1. The curves for the test loss are qualitatively similar to one of the two cases analysed for linear regression (Figure 1A,B,C). Again, performance generally increases with budget and test loss curves tend to have a unique minimum. Therefore, different data allocations for a given budget can result in significantly different performance and there exists an optimal allocation for a given budget. Figure 5B shows that the optimum $N^\star$ increases from a small value at small budgets to an approximately constant value for large budgets, of about 150 data points per task. Our theoretical results of a constant optimum at large budgets, obtained in section 5 for mixed linear regression, seems to be confirmed in the case of nonlinear regression.

The neural network used in all sinusoid experiments was a MLP with 2 hidden layers of 40 nodes each with ReLU nonlinearities. During meta-training we used full batch gradient descent. The initial learning rate for performing gradient updates in the outer loop of the MAML algorithm was 0.001, while the learning rate for the adaptation procedure (inner loop of MAML) was 0.01. In all experiments we used a learning rate annealing schedule upon reaching a plateau in the training loss. This was done three times with the fourth incurring termination of training. Additionally, we only started the learning rate scheduler after a predetermined number of meta-iterations which depended on the budget.

Unlike the experimental settings in other meta-learning papers, we do not use a fixed number of shots for meta-training since we focus on the investigation of different numbers of data points per task. The train-test split for each meta-training task dataset was 0.5. As a result, the minimum number of data points per task was 2 (i.e. 1 for training and the other for validation). The maximum number of data points per task was equal to the budget b in which case there was only one task (m=1). The training was done using full batches of data and the Adam optimizer ([37]).

During meta-testing, we randomly sampled 1000 tasks and 500 data points per task for adaptation, and 500 data points per task for computing the test loss. We used one step adaptation during meta-training and five steps during meta-testing.

For the grid search, we performed up to 10 independent runs for each point on the grid with randomly sampled meta-training data and randomly initialised model weights. To obtain the statistics of the optimal $N^\star$ for each budget in the grid search experiments, we employed the following strategy. We randomly sampled one run for each point on the grid and derived the optimal $N^\star$ from this sample and repeated the sampling for 1000 times. The mean and standard deviation of $N^\star$ are computed and presented in Figure 5B.

C Computation of the average test loss

This section largely follows the results of [10], but here the calculations are generalized to the case of heterogeneous tasks and parametric input variability.

C.1 Definition of the loss

We consider the problem of mixed linear regression $\mathbf{y} = X\mathbf{w} + \mathbf{z}$ with squared loss, where X is a $n \times p$ matrix of input data, each row is one of n data vectors of dimension p, $\mathbf{z}$ is a $n \times 1$ noise vector, $\mathbf{w}$ is a $p \times 1$ vector of generating parameters and $\mathbf{y}$ is a $n \times 1$ output vector. Data is collected

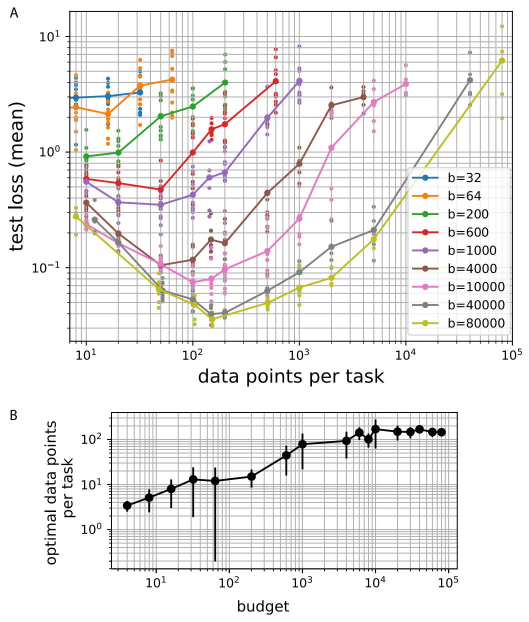

Figure 5: **Nonlinear regression.** Format is the same as of Figure 1. Results are qualitatively similar to Figure 1A,B with the optimal number of data points per task increasing and then converging to a constant number (~ 150) at large budgets (panel B).

for m tasks, each with a different value of the parameters $\mathbf{w}$ and a different realization of the input X and noise $\mathbf{z}$. We denote by $\mathbf{w}^{(i)}$ the parameters for task i, for $i = 1, \dots, m$. For a given task i, we denote by X_i^t, X_i^v the input data for, respectively, the training and validation sets, by $\mathbf{z}_i^t, \mathbf{z}_i^v$ the corresponding noise vectors and by $\mathbf{y}_i^t, \mathbf{y}_i^v$ the output vectors. We denote by n_i^t, n_i^v the data sample size for task i, respectively for training and validations sets.

For a given task i, the training output is equal to

$$\mathbf{y}_i^t = X_i^t \mathbf{w}^{(i)} + \mathbf{z}_i^t \tag{19}$$

Similarly, the validation output is equal to

$$\mathbf{y}_i^v = X_i^v \mathbf{w}^{(i)} + \mathbf{z}_i^v. \tag{20}$$

We consider MAML as a model for meta-learning (Finn et al 2017). The meta-training loss is equal to

$$\mathcal{L}^{meta} = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2n_i^v} \left| \mathbf{y}_i^v - X_i^v \boldsymbol{\theta}_i(\boldsymbol{\omega}) \right|^2 \tag{21}$$

where vertical brackets denote euclidean norm, and the estimated parameters $\boldsymbol{\theta}_i(\boldsymbol{\omega})$ are equal to the one-step gradient update on the single-task training loss $\mathcal{L}^{(i)} = |\mathbf{y}_i^t - X_i^t \boldsymbol{\theta}_i|^2 / 2n_i^t$, with initial condition given by the meta-parameter $\boldsymbol{\omega}$. The single gradient update is equal to

$$\boldsymbol{\theta}_i(\boldsymbol{\omega}) = \left(I_p - \frac{\alpha_i}{n_i^t} X_i^{t\,T} X_i^t \right) \boldsymbol{\omega} + \frac{\alpha_i}{n_i^t} X_i^{t\,T} \mathbf{y}_i^t \tag{22}$$

where I_p is the $p \times p$ identity matrix and α_i is the learning rate. We seek to minimize the meta-training loss with respect to the meta-parameter $\boldsymbol{\omega}$, namely

$$\boldsymbol{\omega}^\star = \arg\min_{\boldsymbol{\omega}} \mathcal{L}^{meta} \tag{23}$$

We evaluate the solution $\boldsymbol{\omega}^\star$ by calculating the meta-test loss

$$\mathcal{L}^{test} = \frac{1}{2n_s} \left| \mathbf{y}^s - X^s \boldsymbol{\theta}^\star \right|^2 \tag{24}$$

Note that the test loss is calculated over test data $X^s, \mathbf{z}^s$, and test parameters $\mathbf{w}'$, namely

$$\mathbf{y}^s = X^s \mathbf{w}' + \mathbf{z}^s \tag{25}$$

Furthermore, the estimated parameters $\boldsymbol{\theta}^\star$ are calculated on a separate set of target data $X^r, \mathbf{z}^r$, namely

$$\boldsymbol{\theta}^\star = \left(I_p - \frac{\alpha_r}{n_r} X^{rT} X^r\right) \boldsymbol{\omega}^\star + \frac{\alpha_r}{n_r} X^{rT} \mathbf{y}^r \tag{26}$$

$$\mathbf{y}^r = X^r \mathbf{w}' + \mathbf{z}^r \tag{27}$$

Note that the learning rate and sample size can be different at testing, denoted by α_r, n_r, n_s. We are interested in calculating the average test loss, that is the test loss of Eq.24 averaged over the entire data distribution, equal to

$$\overline{\mathcal{L}}^{test} = \underset{\mathbf{w}}{\mathbb{E}} \underset{\mathbf{z}^t}{\mathbb{E}} \underset{X^t}{\mathbb{E}} \underset{\mathbf{z}^v}{\mathbb{E}} \underset{X^v}{\mathbb{E}} \underset{\mathbf{w}'}{\mathbb{E}} \underset{\mathbf{z}^s}{\mathbb{E}} \underset{X^s}{\mathbb{E}} \underset{\mathbf{z}^r}{\mathbb{E}} \underset{X^r}{\mathbb{E}} \frac{1}{2n_s} \left|\mathbf{y}^s - X^s \boldsymbol{\theta}^\star\right|^2 \tag{28}$$

C.2 Probability distributions and averaging

We assume that all random variables are Gaussian. In particular, we assume that the rows of the matrix X are independent, and each row, denoted by $\mathbf{x}$, is distributed according to a multivariate Gaussian with zero mean and unit covariance

$$\mathbf{x} \sim \mathcal{N}\left(0, \lambda^2 I_p\right) \tag{29}$$

where I_p is the $p \times p$ identity matrix. Similarly, the noise is distributed following a multivariate Gaussian with zero mean and variance equal to σ^2, namely

$$\mathbf{z} \sim \mathcal{N}\left(0, \sigma^2 I_n\right) \tag{30}$$

Finally, the generating parameters are also distributed according to a multivariate Gaussian of variance ν^2/p, namely

$$\mathbf{w} \sim \mathcal{N}\left(\mathbf{w}_0, \frac{\nu^2}{p} I_p\right) \tag{31}$$

The generating parameter $\mathbf{w}$ is drawn once and kept fixed within a task, and drawn independently for different tasks. The values of $\mathbf{x}$ and $\mathbf{z}$ are drawn independently in all tasks and datasets (training, validation, target, test). In order to perform the calculations in the next section, we need the following results.

Lemma 1. Let X be a Gaussian $n \times p$ random matrix with independent rows, and each row has covariance equal to I_p, the $p \times p$ identity matrix. Then:

$$\mathbb{E}\left[X^T X\right] = \lambda^2 n I_p \tag{32}$$

$$\mathbb{E}\left[\left(X^T X\right)^2\right] = \lambda^4 n (n + p + 1) I_p = \lambda^4 n^2 \mu_2 I_p \tag{33}$$

$$\mathbb{E}\left[\left(X^T X\right)^3\right] = \lambda^6 n \left(n^2 + p^2 + 3np + 3n + 3p + 4\right) I_p = \lambda^6 n^3 \mu_3 I_p \tag{34}$$

$$\mathbb{E}\left[\left(X^T X\right)^4\right] = \lambda^8 n \left(n^3 + p^3 + 6n^2 p + 6np^2 + \right. \tag{35}$$

$$\left. + 6n^2 + 6p^2 + 17np + 21n + 21p + 20\right) I_p = \lambda^8 n^4 \mu_4 I_p \tag{36}$$

$$\mathbb{E}\left[X^T X \ \mathrm{Tr}\left(X^T X\right)\right] = \lambda^4 \left(n^2 p + 2n\right) I_p = \lambda^4 p n^2 \mu_{1,1} I_p \tag{37}$$

$$\mathbb{E}\left[\left(X^T X\right)^2 \mathrm{Tr}\left(X^T X\right)\right] = \lambda^6 n \left(n^2 p + np^2 + np + 4n + 4p + 4\right) I_p = \lambda^6 p n^3 \mu_{2,1} I_p \tag{38}$$

$$\mathbb{E}\left[X^T X \mathrm{Tr}\left(\left(X^T X\right)^2\right)\right] = \lambda^6 n \left(n^2 p + np^2 + np + 4n + 4p + 4\right) I_p = \lambda^6 p n^3 \mu_{1,2} I_p \tag{39}$$

$$\mathbb{E}\left[\left(X^T X\right)^2 \mathrm{Tr}\left(\left(X^T X\right)^2\right)\right] = \lambda^8 n \left(n^3 p + np^3 + 2n^2 p^2 + 2n^2 p + 2np^2 + \right. \tag{40}$$

$$\left. + 8n^2 + 8p^2 + 21np + 20n + 20p + 20\right) I_p = \lambda^8 p n^4 \mu_{2,2} I_p \tag{41}$$

where the last equality in each of these expressions defines the variables μ. Furthermore, for any $n \times n$ symmetric matrix C and any $p \times p$ symmetric matrix D, independent of X:

$$\mathbb{E}\left[X^T C X\right] = \lambda^2 \mathrm{Tr}\,(C)\, I_p \tag{42}$$

$$\mathbb{E}\left[X^T X D X^T X\right] = \lambda^4 n (n+1) D + \lambda^4 n \mathrm{Tr}\,(D)\, I_p \tag{43}$$

Proof. The Lemma follows by direct computations of the above expectations, using Isserlis' theorem. Particularly, for higher order exponents, combinatorics plays a crucial role in counting products of different Gaussian variables in an effective way.

□

Lemma 2. Let X_i^v, X_i^t be Gaussian random matrices, of size respectively $n_v \times p$ and $n_t \times p$, with independent rows, and each row has covariance equal to I_p, the $p \times p$ identity matrix. Let $p(\xi)$ and $n_t(\xi)$ be any function of order $O(\xi)$ as $\xi \to \infty$. Then:

$$X_i^v X_i^{vT} = \lambda^2 p\, I_{n_v} + O\left(\xi^{1/2}\right) \tag{44}$$

$$X_i^v X_i^{t^T} X_i^t X_i^{vT} = \lambda^4 p n_t\, I_{n_v} + O\left(\xi^{3/2}\right) \tag{45}$$

$$X_i^v X_i^{t^T} X_i^t X_i^{t^T} X_i^t X_i^{vT} = \lambda^6 p n_t (n_t + p + 1) I_{n_v} + O\left(\xi^{5/2}\right) \tag{46}$$

Note that the order $O\left(\xi\right)$ applies to all elements of the matrix in each expression. For $i \neq j$

$$X_i^v X^{v(j)^T} = O\left(\xi^{1/2}\right) \tag{47}$$

$$X_i^v X_i^{t^T} X_i^t X^{v(j)^T} = O\left(\xi^{3/2}\right) \tag{48}$$

$$X_i^v X_i^{t^T} X_i^t X^{t(j)^T} X^{t(j)} X^{v(j)^T} = O\left(\xi^{5/2}\right) \tag{49}$$

Furthermore, for any positive real number δ and for any $p \times p$ symmetric matrix D independent of X, where $\mathrm{Tr}(D)$ and $\mathrm{Tr}(D^2)$ are both of order $O(\xi^\delta)$

$$X_i^v D X_i^{vT} = \lambda^2 \mathrm{Tr}\left(D\right) I_{n_v} + O\left(\xi^{\delta/2}\right) \tag{50}$$

$$X_i^v X_i^{t^T} X_i^t D X_i^{vT} = \lambda^4 \mathrm{Tr}\left(D\right) n_t I_{n_v} + O\left(\xi^{1+\delta/2}\right) \tag{51}$$

$$X_i^v X_i^{t^T} X_i^t D X_i^{t^T} X_i^t X_i^{vT} = \lambda^6 \mathrm{Tr}\left(D\right) n_t (n_t + p + 1) I_{n_v} + O\left(\xi^{2+\delta/2}\right) \tag{52}$$

$$X_i^v D X^{v(j)^T} = O\left(\xi^{\delta/2}\right) \tag{53}$$

$$X_i^v X_i^{t^T} X_i^t D X^{v(j)^T} = O\left(\xi^{1+\delta/2}\right) \tag{54}$$

$$X_i^v X_i^{t^T} X_i^t D X^{t(j)^T} X^{t(j)} X^{v(j)^T} = O\left(\xi^{2+\delta/2}\right) \tag{55}$$

Proof. The Lemma follows by direct computations of the expectations and variances of each term.

□

Lemma 3. Let X^v, X^t be Gaussian random matrices, of size respectively $n_v \times p$ and $n_t \times p$, with independent rows, and each row has covariance equal to I_p, the $p \times p$ identity matrix. Let $n_v(\xi)$ and $n_t(\xi)$ be any function of order $O(\xi)$ for $\xi \to \infty$. Then:

$$X^{vT} X^v = \lambda^2 n_v\, I_p + O\left(\xi^{1/2}\right) \tag{56}$$

$$X^{t^T} X^t X^{vT} X^v = \lambda^4 n_t n_v\, I_p + O\left(\xi^{3/2}\right) \tag{57}$$

$$X^{t^T} X^t X^{vT} X^v X^{t^T} X^t = \lambda^6 n_v n_t (n_t + p + 1) I_p + O\left(\xi^{5/2}\right) \tag{58}$$

Note that the order $O\left(\xi\right)$ applies to all elements of the matrix in each expression.

Proof. The Lemma follows by direct computations of the expectations and variances of each term.

□

C.3 Averaging over test data

We calculate the average test loss as a function of the hyperparameters n_i^t, n_i^v, n_r, p, m, α_i, α_r, σ_i, σ_r, λ_i, λ_r, ν, $\mathbf{w}_0$. The subscript i denotes meta-training hyperparameters for task i, while the subscript r denotes meta-testing hyperparameters. Using the expression in Eq.25 for the test output, we rewrite the test loss in Eq.28 as

$$\overline{\mathcal{L}}^{test} = \mathbb{E}\,\frac{1}{2n_s}\left|X^s\left(\mathbf{w}'-\boldsymbol{\theta}^\star\right)+\mathbf{z}^s\right|^2 \tag{59}$$

We start by averaging this expression with respect to $X^s, \mathbf{z}^s$, noting that $\boldsymbol{\theta}^\star$ does not depend on test data. We further average with respect to $\mathbf{w}'$, but note that $\boldsymbol{\theta}^\star$ depends on test parameters, so we average only terms that do not depend on $\boldsymbol{\theta}^\star$. Using Eq.32, the result is

$$\overline{\mathcal{L}}^{test} = \frac{\sigma_r^2}{2}+\frac{\lambda_r^2}{2}\left(\nu^2+|\mathbf{w}_0|^2\right)+\lambda_r^2\mathbb{E}\left[\frac{|\boldsymbol{\theta}^\star|^2}{2}-\left(\mathbf{w}_0+\delta\mathbf{w}'\right)^T\boldsymbol{\theta}^\star\right] \tag{60}$$

where we define $\delta\mathbf{w}'=\mathbf{w}'-\mathbf{w}_0$. The second term in the expectation is linear in $\boldsymbol{\theta}^\star$ and can be averaged over $X^r, \mathbf{z}^r$, using Eq.26 and noting that $\boldsymbol{\omega}^\star$ does not depend on target data. The result is

$$\underset{X^r}{\mathbb{E}}\,\underset{\mathbf{z}^r}{\mathbb{E}}\,\boldsymbol{\theta}^\star=(1-\lambda_r^2\alpha_r)\boldsymbol{\omega}^\star+\lambda_r^2\alpha_r\left(\mathbf{w}_0+\delta\mathbf{w}'\right) \tag{61}$$

Using Eq.61 we average over $\mathbf{w}'$ the second term in the expectation of Eq.60 and find

$$\overline{\mathcal{L}}^{test} = \frac{\sigma_r^2}{2}+\lambda_r^2\left(\frac{1}{2}-\lambda_r^2\alpha_r\right)\left(\nu^2+|\mathbf{w}_0|^2\right)-\lambda_r^2\left(1-\lambda_r^2\alpha_r\right)\mathbf{w}_0^T\,\mathbb{E}\,\boldsymbol{\omega}^\star+\lambda^2\mathbb{E}\frac{|\boldsymbol{\theta}^\star|^2}{2} \tag{62}$$

We average the last term of this expression over $\mathbf{z}^r, \mathbf{w}'$, using Eq.26 and noting that $\boldsymbol{\omega}^\star$ does not depend on target data and test parameters. The result is

$$\underset{\mathbf{w}'}{\mathbb{E}}\,\underset{\mathbf{z}^r}{\mathbb{E}}\,|\boldsymbol{\theta}^\star|^2=|\boldsymbol{\omega}^\star|^2+\frac{\alpha_r^2}{n_r^2}\left(\boldsymbol{\omega}^\star-\mathbf{w}_0\right)^T\left(X^{rT}X^r\right)^2\left(\boldsymbol{\omega}^\star-\mathbf{w}_0\right)- \tag{63}$$

$$-\frac{2\alpha_r}{n_r}\boldsymbol{\omega}^{\star T}X^{rT}X^r\left(\boldsymbol{\omega}^\star-\mathbf{w}_0\right)+\frac{\alpha_r^2\sigma_r^2}{n_r^2}\mathrm{Tr}\left[X^rX^{rT}\right]+\frac{\alpha_r^2\nu^2}{n_r^2p}\mathrm{Tr}\left[\left(X^rX^{rT}\right)^2\right] \tag{64}$$

We now average over X^r, again noting that $\boldsymbol{\omega}^\star$ does not depend on target data. Using Eqs.32, 33, we find

$$\underset{X^r}{\mathbb{E}}\,\underset{\mathbf{w}'}{\mathbb{E}}\,\underset{\mathbf{z}^r}{\mathbb{E}}\,|\boldsymbol{\theta}^\star|^2=|\boldsymbol{\omega}^\star|^2+\lambda_r^4\alpha_r^2\left(1+\frac{p+1}{n_r}\right)\left(\nu^2+|\boldsymbol{\omega}^\star-\mathbf{w}_0|^2\right)-2\lambda_r^2\alpha_r\boldsymbol{\omega}^{\star T}\left(\boldsymbol{\omega}^\star-\mathbf{w}_0\right)+\frac{\lambda_r^2\alpha_r^2\sigma_r^2p}{n_r} \tag{65}$$

We can now rewrite the average test loss 62 as

$$\overline{\mathcal{L}}^{test} = \frac{\sigma_r^2}{2}\left(1+\frac{\lambda_r^4\alpha_r^2p}{n_r}\right)+\frac{\lambda_r^2}{2}\left[\left(1-\lambda_r^2\alpha_r\right)^2+\lambda_r^4\alpha_r^2\frac{p+1}{n_r}\right]\left(\nu^2+\mathbb{E}\,|\boldsymbol{\omega}^\star-\mathbf{w}_0|^2\right) \tag{66}$$

In order to average the last term, we need an expression for $\boldsymbol{\omega}^\star$. We note that the loss in Eq.21 is quadratic in $\boldsymbol{\omega}$, therefore the solution of Eq.23 can be found using standard linear algebra. In particular, the loss in Eq.21 can be rewritten as

$$\mathcal{L}^{meta}=\frac{1}{2m}\left|\boldsymbol{\gamma}-B\boldsymbol{\omega}\right|^2 \tag{67}$$

where $\boldsymbol{\gamma}$ is a column vector of $\sum_{i=1}^m n_i^v$ components, and B is a matrix of shape $\sum_{i=1}^m n_i^v \times p$. The vector $\boldsymbol{\gamma}$ is a stack of m vectors

$$\boldsymbol{\gamma}=\begin{pmatrix}\frac{1}{\sqrt{n_1^v}}\left[X_1^v\left(I_p-\frac{\alpha_1}{n_1^t}X_1^{tT}X_1^t\right)\mathbf{w}_1-\frac{\alpha_1}{n_1^t}X_1^vX_1^{tT}\mathbf{z}_1^t+\mathbf{z}_1^v\right]\\ \vdots \\ \frac{1}{\sqrt{n_m^v}}\left[X_m^v\left(I_p-\frac{\alpha_m}{n_m^t}X_m^{t\,T}X_m^t\right)\mathbf{w}_m-\frac{\alpha_m}{n_m^t}X_m^vX_m^{t\,T}\mathbf{z}_m^t+\mathbf{z}_m^v\right]\end{pmatrix} \tag{68}$$

Similarly, the matrix B is a stack of m matrices

$$B = \begin{pmatrix} \frac{1}{\sqrt{n_1^v}} \left[X_1^v \left(I_p - \frac{\alpha_1}{n_1^t} X_1^{t\,T} X_1^t \right) \right] \\ \vdots \\ \frac{1}{\sqrt{n_m^v}} \left[X_m^v \left(I_p - \frac{\alpha_m}{n_m^t} X_m^{t\,T} X_m^t \right) \right] \end{pmatrix} \tag{69}$$

We denote by I_p the $p \times p$ identity matrix. The expression for $\boldsymbol{\omega}$ that minimizes Eq.67 depends on whether the problem is overparameterized ($p > \sum_{i=1}^m n_i^v$) or underparameterized ($p < \sum_{i=1}^m n_i^v$), therefore we distinguish these two cases in the following sections.

C.4 Averaging over training data: overparameterized case

In the overparameterized case ($p > \sum_{i=1}^m n_i^v$), under the assumption that the inverse of BB^T exists, the value of $\boldsymbol{\omega}$ that minimizes Eq.67 is equal to

$$\boldsymbol{\omega}^\star = B^T \left(BB^T\right)^{-1} \boldsymbol{\gamma} + \left[I_p - B^T \left(BB^T\right)^{-1} B \right] \boldsymbol{\omega}_0 \tag{70}$$

The vector $\boldsymbol{\omega}_0$ is interpreted as the initial condition of the parameter optimization of the outer loop, when optimized by gradient descent. Note that the matrix B does not depend on $\mathbf{w}, \mathbf{z}^t, \mathbf{z}^v$, and $\mathbb{E}_{\mathbf{w}} \mathbb{E}_{\mathbf{z}^t} \mathbb{E}_{\mathbf{z}^v} \;\boldsymbol{\gamma} = B\mathbf{w}_0$. We denote by $\delta\boldsymbol{\gamma}$ the deviation from the average, and we have

$$\boldsymbol{\omega}^\star - \mathbf{w}_0 = B^T \left(BB^T\right)^{-1} \delta\boldsymbol{\gamma} + \left[I_p - B^T \left(BB^T\right)^{-1} B \right] (\boldsymbol{\omega}_0 - \mathbf{w}_0) \tag{71}$$

We square this expression and average over $\mathbf{w}, \mathbf{z}^t, \mathbf{z}^v$. We use the cyclic property of the trace and the fact that $B^T \left(BB^T\right)^{-1} B$ is a projection. The result is

$$|\boldsymbol{\omega}^\star - \mathbf{w}_0|^2 = \text{Tr} \left[\Gamma \left(BB^T\right)^{-1} \right] + (\boldsymbol{\omega}_0 - \mathbf{w}_0)^T \left[I_p - B^T \left(BB^T\right)^{-1} B \right] (\boldsymbol{\omega}_0 - \mathbf{w}_0) \tag{72}$$

The matrix Γ is defined as

$$\Gamma = \mathop{\mathbb{E}}_{\mathbf{w}} \mathop{\mathbb{E}}_{\mathbf{z}^t} \mathop{\mathbb{E}}_{\mathbf{z}^v} \delta\boldsymbol{\gamma}\; \delta\boldsymbol{\gamma}^T = \begin{pmatrix} \Gamma^{(1)} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \Gamma^{(m)} \end{pmatrix} \tag{73}$$

Where matrix blocks are given by the following expression

$$\Gamma^{(i)} = \frac{\nu^2}{n_i^v p} X_i^v \left(I_p - \frac{\alpha_i}{n_i^t} X_i^{t\,T} X_i^t \right)^2 X_i^{v\,T} + \frac{\sigma_i^2}{n_i^v} \left(I_{n_i^v} + \frac{\alpha_i^2}{{n_i^t}^2} X_i^v X_i^{t\,T} X_i^t X_i^{v\,T} \right) \tag{74}$$

It is convenient to rewrite the scalar product of Eq.72 in terms of the trace of outer products

$$|\boldsymbol{\omega}^\star - \mathbf{w}_0|^2 = \text{Tr} \left[\left(BB^T\right)^{-1} \left(\Gamma - B\,(\boldsymbol{\omega}_0 - \mathbf{w}_0)\,(\boldsymbol{\omega}_0 - \mathbf{w}_0)^T B^T \right) \right] + |\boldsymbol{\omega}_0 - \mathbf{w}_0|^2 \tag{75}$$

In order to calculate $\mathbb{E}\,|\boldsymbol{\omega}^\star - \mathbf{w}_0|^2$ in Eq.66 we need to average this expression over training and validation data. These averages are hard to compute since they involve nonlinear functions of the data. However, we can approximate these terms by assuming that p and n_i^t are large, both of order $O(\xi)$, where ξ is a large number. Furthermore, we assume that $|\boldsymbol{\omega}_0 - \mathbf{w}_0|$ is of order $O(\xi^{-1/4})$. Using Lemma 2, together with the expressions of B (Eq.69) and Γ (Eqs.73,74), we can prove that

$$\frac{1}{p} BB^T = \begin{pmatrix} {n_1^v}^{-1} {\lambda_1}^2 h_1 I_{n_1^v} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & {n_m^v}^{-1} {\lambda_m}^2 h_m I_{n_m^v} \end{pmatrix} + O\left(\xi^{-1/2}\right) \tag{76}$$

$$\Gamma^{(i)} = \left\{ \frac{\nu^2 \lambda_i^2 h_i}{n_i^v} + \frac{\sigma_i^2}{n_i^v} \left(1 + \frac{\lambda_i^4 \alpha_i^2 p}{n_i^t} \right) \right\} I_{n_i^v} + O\left(\xi^{-1/2}\right) \tag{77}$$

$$B\left(\boldsymbol{\omega}_0-\mathbf{w}_0\right)\left(\boldsymbol{\omega}_0-\mathbf{w}_0\right)^T B^T=\left|\boldsymbol{\omega}_0-\mathbf{w}_0\right|^2\begin{pmatrix} {n_1^v}^{-1}{\lambda_1}^2 h_1 I_{n_1^v} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & {n_m^v}^{-1}{\lambda_m}^2 h_m I_{n_m^v}\end{pmatrix}+O\left(\xi^{-1/2}\right) \tag{78}$$

where we define the following expression

$$h_i=\left(1-\lambda_i^2\alpha_i\right)^2+\lambda_i^4\alpha_i^2\frac{p+1}{n_i^t} \tag{79}$$

Using Eq.76 and Taylor expansion, the inverse $\left(BB^T\right)^{-1}$ is equal to

$$\left(BB^T\right)^{-1}=\frac{1}{p}\begin{pmatrix} n_1^v{\lambda_1}^{-2}{h_1}^{-1}I_{n_1^v} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & n_m^v{\lambda_m}^{-2}{h_m}^{-1}I_{n_m^v}\end{pmatrix}+O\left(\xi^{-3/2}\right), \tag{80}$$

Substituting the three expressions above in Eq.75, and ignoring terms of lower order, we find

$$\mathbb{E}\left|\boldsymbol{\omega}^\star-\mathbf{w}_0\right|^2=\left(1-\frac{1}{p}\sum_{i=1}^m n_i^v\right)\left|\boldsymbol{\omega}_0-\mathbf{w}_0\right|^2+ \tag{81}$$

$$+\frac{\nu^2}{p}\sum_{i=1}^m n_i^v+\frac{1}{p}\sum_{i=1}^m\frac{\sigma_i^2 n_i^v}{\lambda_i^2 h_i}\left(1+\frac{\lambda_i^4\alpha_i^2 p}{n_i^t}\right)+O\left(\xi^{-3/2}\right) \tag{82}$$

Substituting this expression into in Eq.66, we find the value of average test loss

$$\overline{\mathcal{L}}^{test}=\frac{\sigma_r^2}{2}\left(1+\frac{\lambda_r^4\alpha_r^2 p}{n_r}\right)+\frac{\lambda_r^2 h_r}{2}\left(1-\frac{1}{p}\sum_{i=1}^m n_i^v\right)\left|\boldsymbol{\omega}_0-\mathbf{w}_0\right|^2+ \tag{83}$$

$$+\frac{\lambda_r^2 h_r\nu^2}{2}\left(1+\frac{1}{p}\sum_{i=1}^m n_i^v\right)+\frac{\lambda_r^2 h_r}{2p}\sum_{i=1}^m\frac{\sigma_i^2 n_i^v}{\lambda_i^2 h_i}\left(1+\frac{\lambda_i^4\alpha_i^2 p}{n_i^t}\right)+O\left(\xi^{-3/2}\right) \tag{84}$$

where we define the following expression

$$h_r=\left(1-\lambda_r^2\alpha_r\right)^2+\lambda_r^4\alpha_r^2\frac{p+1}{n_r} \tag{85}$$

C.5 Averaging over training data: underparameterized case

In the underparameterized case ($p<\sum_{i=1}^m n_i^v$), under the assumption that the inverse of B^TB exists, the value of $\boldsymbol{\omega}$ that minimizes Eq.67 is equal to

$$\boldsymbol{\omega}^\star=\left(B^TB\right)^{-1}B^T\boldsymbol{\gamma} \tag{86}$$

Note that the matrix B does not depend on $\mathbf{w},\mathbf{z}^t,\mathbf{z}^v$, and $\mathbb{E}_{\mathbf{w}}\,\mathbb{E}_{\mathbf{z}^t}\,\mathbb{E}_{\mathbf{z}^v}\;\boldsymbol{\gamma}=B\mathbf{w}_0$. We denote by $\delta\boldsymbol{\gamma}$ the deviation from the average, and we have

$$\left|\boldsymbol{\omega}^\star-\mathbf{w}_0\right|^2=\mathrm{Tr}\left[\left(B^TB\right)^{-1}B^T\delta\boldsymbol{\gamma}\,\delta\boldsymbol{\gamma}^TB\left(B^TB\right)^{-1}\right] \tag{87}$$

We need to average this expression in order to calculate $\mathbb{E}\left|\boldsymbol{\omega}^\star-\mathbf{w}_0\right|^2$ in Eq.66. We start by averaging $\delta\boldsymbol{\gamma}\;\delta\boldsymbol{\gamma}^T$ over $\mathbf{w},\mathbf{z}^t,\mathbf{z}^v$, since B does not depend on those variables. Note that $\mathbf{w},\mathbf{z}^t,\mathbf{z}^v$ are independent on each other and across tasks. As in previous section, we denote by Γ the result of this operation, given by Eq.s73, 74. Finally, we need to average over the training and validation data

$$\mathbb{E}\left|\boldsymbol{\omega}^\star-\mathbf{w}_0\right|^2=\mathop{\mathbb{E}}_{X^t}\mathop{\mathbb{E}}_{X^v}\mathrm{Tr}\left[\left(B^TB\right)^{-1}B^T\Gamma B\left(B^TB\right)^{-1}\right] \tag{88}$$

It is hard to average this expression because it includes nonlinear functions of the data. However, we can approximate these terms by assuming that either m or ξ (or both) is a large number, where ξ is defined by assuming that n_i^t and n_i^v are of order $O(\xi)$. Using Lemma 3, together with the expression of B (Eq.69), and noting that each factor in Eq.88 has a sum over m independent terms, we can prove that

$$B^T B = \sum_{i=1}^{m} \lambda_i^2 h_i I_p + O\left((m\xi)^{1/2}\right) \tag{89}$$

Using this result and a Taylor expansion, the inverse is equal to

$$\left(B^T B\right)^{-1} = \left[\sum_{i=1}^{m} \lambda_i^2 h_i\right]^{-1} I_p + O\left((m\xi)^{-3/2}\right) \tag{90}$$

Similarly, the term $B^T \Gamma B$ is equal to its average plus a term of smaller order

$$B^T \Gamma B = \mathbb{E}\left(B^T \Gamma B\right) + O\left((m\xi)^{1/2}\right) \tag{91}$$

We substitute these expressions in Eq.88 and neglect lower orders. Here we show how to calculate explicitly the expectation of $B^T \Gamma B$. For ease of notation, we define the matrix $A_i^t = I - \frac{\alpha_i}{n_i^t} X_i^{t^T} X_i^t$. Using the expressions of B (Eq.69) and Γ (Eqs.73,74), the expression for $B^T \Gamma B$ is given by

$$B^T \Gamma B = \sum_{i=1}^{m} n_i^{v-2} \sigma_i^2 A_i^{t^T} X_i^{v^T} X_i^v A_i^t + \frac{\nu^2}{p} \sum_{i=1}^{m} n_i^{v-2} \left(A_i^{t^T} X_i^{v^T} X_i^v A_i^t\right)^2 +$$

$$+ \sum_{i=1}^{m} n_i^{v-2} n_i^{t-2} \alpha_i^2 \sigma_i^2 A_i^{t^T} X_i^{v^T} X_i^v X_i^{t^T} X_i^t X_i^{v^T} X_i^v A_i^t \tag{92}$$

We use Eqs.32, 33 to calculate the average of the first term in Eq.92

$$\mathop{\mathbb{E}}_{X^t} \mathop{\mathbb{E}}_{X^v} \sum_{i=1}^{m} n_i^{v-2} \sigma_i^2 A_i^{t^T} X_i^{v^T} X_i^v A_i^t = \sum_{i=1}^{m} n_i^{v-1} \lambda_i^2 \sigma_i^2 h_i I_p \tag{93}$$

We use Eqs.32, 33, 34, 42, 37, 38, 39, 40 to calculate the average of the second term

$$\mathop{\mathbb{E}}_{X^t} \mathop{\mathbb{E}}_{X^v} \sum_{i=1}^{m} n_i^{v-2} \left(A_i^{t^T} X_i^{v^T} X_i^v A_i^t\right)^2 = \mathop{\mathbb{E}}_{X^t} \sum_{i=1}^{m} n_i^{v-1} \lambda_i^4 \left[\left(n_i^v + 1\right) A_i^{t^4} + A_i^{t^2} \mathrm{Tr}\left(A_i^{t^2}\right)\right] = \tag{94}$$

$$= \sum_{i=1}^{m} n_i^{v-1} \lambda_i^4 \left(n_i^v + 1\right) \left(1 - 4\lambda_i^2 \alpha_i + 6\lambda_i^4 \alpha_i^2 \mu_2^{(i)} - 4\lambda_i^6 \alpha_i^3 \mu_3^{(i)} + \lambda_i^8 \alpha_i^4 \mu_4^{(i)}\right) I_p +$$

$$+ \sum_{i=1}^{m} n_i^{v-1} \lambda_i^4 p \left(1 - 4\lambda_i^2 \alpha_i + 2\lambda_i^4 \alpha_i^2 \mu_2^{(i)} + 4\lambda_i^4 \alpha_i^2 \mu_{1,1}^{(i)} - 4\lambda_i^6 \alpha_i^3 \mu_{2,1}^{(i)} + \lambda_i^8 \alpha_i^4 \mu_{2,2}^{(i)}\right) I_p \tag{95}$$

where we have defined

$$\mu_2^{(i)} = \frac{1}{n_i^t} \left(n_i^t + p + 1\right) \tag{96}$$

$$\mu_3^{(i)} = \frac{1}{n_i^{t^2}} \left(n_i^{t^2} + p^2 + 3n_i^t p + 3n_i^t + 3p + 4\right) \tag{97}$$

$$\mu_4^{(i)} = \frac{1}{n_i^{t^3}} \left(n_i^{t^3} + p^3 + 6n_i^{t^2} p + 6n_i^t p^2 + 6n_i^{t^2} + 6p^2 + 17n_i^t p + 21n_i^t + 21p + 20\right) \tag{98}$$

$$\mu_{1,1}^{(i)} = \frac{1}{n_i^{t^2} p} \left(n_i^{t^2} p + 2n_i^t\right) \tag{99}$$

$$\mu_{2,1}^{(i)} = \frac{1}{n_i^{t^2} p} \left(n_i^{t^2} p + n_i^t p^2 + n_i^t p + 4n_i^t + 4p + 4\right) \tag{100}$$

$$\mu_{2,2}^{(i)} = \frac{1}{n_i^{t^3} p} \left(n_i^{t^3} p + n_i^t p^3 + 2n_i^{t^2} p^2 + 2n_i^{t^2} p + 2n_i^t p^2 + 8n_i^{t^2} + 8p^2 + 21n_i^t p + 20n_i^t + 20p + 20\right) \tag{101}$$

Finally, we compute the average of the third term, using Eqs.32, 33, 34, 35, 42, 37, 38

$$\underset{X^t}{\mathbb{E}}\underset{X^v}{\mathbb{E}}\sum_{i=1}^{m} n_i^{v-2} n_i^{t-2}\alpha_i^2\sigma_i^2 A_i^{t^T} X_i^{v^T} X_i^v X_i^{t^T} X_i^t X_i^{v^T} X_i^v A_i^t = \tag{102}$$

$$= \underset{X^t}{\mathbb{E}}\sum_{i=1}^{m}\frac{\lambda_i^4\alpha_i^2\sigma_i^2}{n_i^v n_i^{t^2}}\left[\left(n_i^v+1\right)A_i^{t^T}X_i^{t^T}X_i^t A_i^t + A_i^{t^T}A_i^t\text{Tr}\left(X_i^{t^T}X_i^t\right)\right] = \tag{103}$$

$$= \sum_{i=1}^{m}\frac{\lambda_i^6\alpha_i^2\sigma_i^2}{n_i^v n_i^t}\left[\left(n_i^v+1\right)\left(1-2\lambda_i^2\alpha_i\mu_2^{(i)}+\lambda_i^4\alpha_i^2\mu_3^{(i)}\right)+p\left(1-2\lambda_i^2\alpha_i\mu_{1,1}^{(i)}+\lambda_i^4\alpha_i^2\mu_{2,1}^{(i)}\right)\right]I_p \tag{104}$$

Putting everything together in Eq.88, and applying the trace operator, we find the following expression for the meta-parameter variance

$$\begin{aligned}\mathbb{E}\left|\boldsymbol{\omega}^\star-\mathbf{w}_0\right|^2 = p\left[\sum_{i=1}^{m}\lambda_i^2 h_i\right]^{-2}\sum_{i=1}^{m}\frac{\lambda_i^2}{n_i^v}\Bigg\{&\sigma_i^2 h_i + \\ &+\frac{\lambda_i^4\alpha_i^2\sigma_i^2}{n_i^t}\left[\left(n_i^v+1\right)\left(1-2\lambda_i^2\alpha_i\mu_2^{(i)}+\lambda_i^4\alpha_i^2\mu_3^{(i)}\right)+p\left(1-2\lambda_i^2\alpha_i\mu_{1,1}^{(i)}+\lambda_i^4\alpha_i^2\mu_{2,1}^{(i)}\right)\right] \\ &+\frac{\nu^2}{p}\lambda_i^2\Bigg[\left(n_i^v+1\right)\left(1-4\lambda_i^2\alpha_i+6\lambda_i^4\alpha_i^2\mu_2^{(i)}-4\lambda_i^6\alpha_i^3\mu_3^{(i)}+\lambda_i^8\alpha_i^4\mu_4^{(i)}\right)+ \\ &+p\left(1-4\lambda_i^2\alpha_i+2\lambda_i^4\alpha_i^2\mu_2^{(i)}+4\lambda_i^4\alpha_i^2\mu_{1,1}^{(i)}-4\lambda_i^6\alpha_i^3\mu_{2,1}^{(i)}+\lambda_i^8\alpha_i^4\mu_{2,2}^{(i)}\right)\Bigg]\Bigg\}+O\left((m\xi)^{-3/2}\right)\end{aligned} \tag{105}$$

We rewrite this expression as

$$\begin{aligned}\mathbb{E}\left|\boldsymbol{\omega}^\star-\mathbf{w}_0\right|^2 = p\left[\sum_{i=1}^{m}\lambda_i^2 h_i\right]^{-2}\sum_{i=1}^{m}\frac{\lambda_i^2}{n_i^v}\Bigg\{&\sigma_i^2\left[h_i+\frac{\lambda_i^4\alpha_i^2}{n_i^t}\left[\left(n_i^v+1\right)g_{1i}+pg_{2i}\right]\right]+ \\ &+\frac{\nu^2}{p}\lambda_i^2\left[\left(n_i^v+1\right)g_{3i}+pg_{4i}\right]\Bigg\}+O\left((m\xi)^{-3/2}\right)\end{aligned} \tag{106}$$

where we defined the following expressions for g_i

$$g_{1i} = 1-2\lambda_i^2\alpha_i\mu_2^{(i)}+\lambda_i^4\alpha_i^2\mu_3^{(i)} \tag{107}$$

$$g_{2i} = 1-2\lambda_i^2\alpha_i\mu_{1,1}^{(i)}+\lambda_i^4\alpha_i^2\mu_{2,1}^{(i)} \tag{108}$$

$$g_{3i} = 1-4\lambda_i^2\alpha_i+6\lambda_i^4\alpha_i^2\mu_2^{(i)}-4\lambda_i^6\alpha_i^3\mu_3^{(i)}+\lambda_i^8\alpha_i^4\mu_4^{(i)} \tag{109}$$

$$g_{4i} = 1-4\lambda_i^2\alpha_i+2\lambda_i^4\alpha_i^2\mu_2^{(i)}+4\lambda_i^4\alpha_i^2\mu_{1,1}^{(i)}-4\lambda_i^6\alpha_i^3\mu_{2,1}^{(i)}+\lambda_i^8\alpha_i^4\mu_{2,2}^{(i)} \tag{110}$$

Substituting this expression back into Eq.66 returns the final expression for the average test loss, equal to

$$\begin{aligned}\overline{\mathcal{L}}^{test} = &\frac{\sigma_r^2}{2}\left(1+\frac{\lambda_r^4\alpha_r^2 p}{n_r}\right)+\frac{\lambda_r^2 h_r\nu^2}{2}+ \\ &+\frac{\lambda_r^2 h_r p}{2}\left[\sum_{i=1}^{m}\lambda_i^2 h_i\right]^{-2}\sum_{i=1}^{m}\frac{\lambda_i^2}{n_i^v}\Bigg\{\sigma_i^2\left[h_i+\frac{\lambda_i^4\alpha_i^2}{n_i^t}\left[\left(n_i^v+1\right)g_{1i}+pg_{2i}\right]\right]+ \\ &+\frac{\nu^2}{p}\lambda_i^2\left[\left(n_i^v+1\right)g_{3i}+pg_{4i}\right]\Bigg\}+O\left((m\xi)^{-3/2}\right)\end{aligned} \tag{111}$$

D Optimal uniform allocation in mixed linear regression

In order to prove Theorem 3 in the main text, under the assumptions of Theorem 2, we start by rewriting Equation (13) in the case of homogeneous tasks ($\sigma_i = \sigma$, $\lambda_i = \lambda$), fixed learning rate

($\alpha_i = \alpha$) and uniform allocation ($n_i = n$). That is similar to the expression of Theorem 2 in [10]. In this case, the budget is equal to $b = 2nm$. We further assume that, in addition to n and m, p is also large, of order $\Theta(\xi)$. Then the average test loss is equal to

$$\overline{\mathcal{L}}^{test} = C_1 + \frac{C_2 p}{b} g_2^{-2} \left\{ \sigma'^2 \left[g_2 + \alpha'^2 \left(g_1 + \frac{p}{n} g_2 \right) \right] + \right.$$
$$\left. + \nu^2 \left[\frac{n}{p} g_3 + g_4 \right] \right\} + O\left(\xi^{-2}\right) \tag{112}$$

where $\sigma' = \frac{\sigma}{\lambda}$, $\alpha' = \lambda^2 \alpha$, and we have defined g_1, g_2, g_3, g_4 as

$$g_1 = (1-\alpha')^2 - 2\alpha' \frac{p}{n} + \alpha'^2 \left(3\frac{p}{n} + \frac{p^2}{n^2} \right) \tag{113}$$
$$g_2 = (1-\alpha')^2 + \alpha'^2 \frac{p}{n} \tag{114}$$
$$g_3 = (1-\alpha')^4 + 6\alpha'^2 \frac{p}{n} - \alpha'^3 \left(12\frac{p}{n} + 4\frac{p^2}{n^2} \right) + \tag{115}$$
$$+ \alpha'^4 \left(6\frac{p}{n} + 6\frac{p^2}{n^2} + \frac{p^3}{n^3} \right) \tag{116}$$
$$g_4 = (1-\alpha')^4 + 2\alpha'^2 \frac{p}{n} - 4\alpha'^3 \frac{p}{n} + \alpha'^4 \left(2\frac{p}{n} + \frac{p^2}{n^2} \right) \tag{117}$$

and C_1, C_2 do not depend on n and are equal to

$$C_1 = \frac{\sigma_r^2}{2} \left(1 + \frac{\lambda_r^4 \alpha_r^2 p}{n_r} \right) + \frac{\nu^2}{2} C_2 \tag{118}$$
$$C_2 = \lambda_r^2 \left(1 - \lambda_r^2 \alpha_r \right)^2 + \lambda_r^6 \alpha_r^2 \frac{p}{n_r} \tag{119}$$

where the subscript r denotes meta-testing hyperparameters.

We drop the term $O\left(\xi^{-2}\right)$, and we optimize the test loss in Eq.(112) for the number of data points per task n, by taking the gradient of $\overline{\mathcal{L}}^{test}$ with respect to n and set it to zero. We obtain the following expression

$$\frac{\partial \overline{\mathcal{L}}^{test}}{\partial n} = \frac{C_2 p}{2b} \frac{\Gamma}{\left(\frac{n g_2}{p} \right)^3} \tag{120}$$

where Γ is a cubic expression in n/p, equal to

$$\Gamma = \nu^2 (1-\alpha')^6 \frac{n^3}{p^3} + 3\nu^2 \alpha'^2 (1-\alpha')^4 \frac{n^2}{p^2} + \tag{121}$$
$$+ 2\alpha'^3 \Big[\nu^2 (2 - \alpha' - 4\alpha'^2 + 3\alpha'^3) + \tag{122}$$
$$+ \sigma'^2 (2 - 5\alpha' + 4\alpha'^2 - \alpha'^3) \Big] \frac{n}{p} + \tag{123}$$
$$+ 2\alpha'^4 \left[\nu^2 (2\alpha'^2 - 1) + \sigma'^2 (2\alpha' - 1) \right] = 0 \tag{124}$$

The zeros of the cubic represent stationary points, maxima or minima, of the loss as a function of n. For ease of notation, we rewrite the cubic as

$$A \left(\frac{n}{p} \right)^3 B \left(\frac{n}{p} \right)^2 + C \left(\frac{n}{p} \right) + D = 0 \tag{125}$$

where the coefficients A, B, C, D are defined as

$$A = \nu^2(1-\alpha')^6 \tag{126}$$

$$B = 3\nu^2\alpha'^2(1-\alpha')^4 \tag{127}$$

$$C = 2\alpha'^3\Big[\nu^2(2-\alpha'-4\alpha'^2+3\alpha'^3)+ \tag{128}$$

$$+\,\sigma'^2(2-5\alpha'+4\alpha'^2-\alpha'^3)\Big] \tag{129}$$

$$D = 2\alpha'^4\left[\nu^2(2\alpha'^2-1)+\sigma'^2(2\alpha'-1)\right] \tag{130}$$

For small α', these coefficients are of different orders, in particular A is $O(1)$, B is of order $O(\alpha'^2)$, C is of order $O(\alpha'^3)$, D is of order $O(\alpha'^4)$. The discriminant of the cubic is equal to

$$-27A^2D^2 + 18ABCD - 4AC^3 - 4B^3D + B^2C^2 \tag{131}$$

We arranged the terms in this expression according to their α' order, which is equal to, respectively, $O(\alpha'^8)$, $O(\alpha'^9)$, $O(\alpha'^9)$, $O(\alpha'^{10})$, $O(\alpha'^{10})$. Note that the leading term is always negative, therefore there exists a value $\alpha'^\star$ such that, for $|\alpha'| < \alpha'^\star$, the discriminant of the cubic is negative, which means that the cubic has only one real solution. This stationary point has to be a minimum, since the derivative of the average test loss exists everywhere (note that $g_2 > 0$) and it becomes large and positive for large positive or negative n.

The cubic equation can be solved using standard methods, such as Cardano's method and depressed cubic transformation. The solution is equal to

$$n^\star = -\frac{p}{3A}\left[B + \left(\frac{-1+\sqrt{-3}}{2}\right)^k\left(\frac{\Delta_1+\sqrt{\Delta_1^2-4\Delta_0^3}}{2}\right)^{\frac{1}{3}} + \left(\frac{-1+\sqrt{-3}}{2}\right)^{-k}\left(\frac{2\Delta_0^3}{\Delta_1+\sqrt{\Delta_1^2-4\Delta_0^3}}\right)^{\frac{1}{3}}\right] \tag{132}$$

for $k = 0, 1, 2$, where the real and positive solution among the three solutions should be selected, and Δ_0, Δ_1 are defined as

$$\Delta_0 = B^2 - 3AC \tag{133}$$

$$\Delta_1 = 2B^3 - 9ABC + 27A^2D \tag{134}$$

Note that in the main text we defined the number of data points per task as $N = n_t + n_v = 2n$, therefore we have $N^\star = 2n^\star$ and

$$N^\star = -\frac{2p}{3A}\left[B + \left(\frac{-1+\sqrt{-3}}{2}\right)^k\left(\frac{\Delta_1+\sqrt{\Delta_1^2-4\Delta_0^3}}{2}\right)^{\frac{1}{3}} + \left(\frac{-1+\sqrt{-3}}{2}\right)^{-k}\left(\frac{2\Delta_0^3}{\Delta_1+\sqrt{\Delta_1^2-4\Delta_0^3}}\right)^{\frac{1}{3}}\right] \tag{135}$$

for $k = 0, 1, 2$, where the real and positive solution among the three solutions should be taken.

The expression for $N^\star$ is rather complicated and it is hard to evaluate how the optimum depends on the hyperparameters of the model. Therefore we computed an approximation that holds for small $|\alpha'|$. Of the three terms in square brackets of Eq.(135), the first one is of order $O(\alpha'^3)$, the second is $O(\alpha'^{4/3})$ and the third is $O(\alpha'^{5/3})$, thus we neglect the first and last term. Furthermore, Δ_0^3 is of order $O(\alpha'^9)$, while Δ_1^2 is $O(\alpha'^8)$, thus we neglect Δ_0, and we keep only the leading term in Δ_1. We obtain

$$N^\star \simeq 2\left[2\left(1+\frac{\sigma'^2}{\nu^2}\right)\right]^{\frac{1}{3}}\alpha'^{\frac{4}{3}}p \tag{136}$$