Learning-to-learn non-convex piecewise-Lipschitz functions

Maria-Florina Balcan, Mikhail Khodak, Dravyansh Sharma
Carnegie Mellon University
{ninamf,mkhodak,dravyans}@cs.cmu.edu

Ameet Talwalkar
Carnegie Mellon University & Hewlett Packard Enterprise
talwalkar@cmu.edu

Abstract

We analyze the meta-learning of the initialization and step-size of learning algorithms for piecewise-Lipschitz functions, a non-convex setting with applications to both machine learning and algorithms. Starting from recent regret bounds for the exponential forecaster on losses with dispersed discontinuities, we generalize them to be initialization-dependent and then use this result to propose a practical meta-learning procedure that learns both the initialization and the step-size of the algorithm from multiple online learning tasks. Asymptotically, we guarantee that the average regret across tasks scales with a natural notion of task-similarity that measures the amount of overlap between near-optimal regions of different tasks. Finally, we instantiate the method and its guarantee in two important settings: robust meta-learning and multi-task data-driven algorithm design.

1 Introduction

While learning-to-learn or *meta-learning* has long been an object of study [45], in recent years it has gained significant attention as a multi-task paradigm for developing algorithms for learning in dynamic environments, from multiple data sources, and in federated settings. Such methods focus on using data from multiple tasks to improve performance when facing a new, potentially related task. A popular approach is *initialization-based* meta-learning, in which the meta-learner uses multi-task data to output an initialization for an iterative algorithm such as stochastic gradient descent (SGD) [25]. The flexibility of this approach has led to its widespread adoption, e.g. in robotics [23] and federated learning [18], and to a growing number of attempts to understand it, both empirically and theoretically [20, 29, 24, 40, 42]. However, outside some stylized setups our learning-theoretic understanding of how to meta-learn an initialization is largely restricted to the convex Lipschitz setting.

We relax both assumptions to study the meta-learning of online algorithms over piecewise-Lipschitz functions, which can be nonconvex and highly discontinuous. As no-regret online learning over such functions is impossible in-general, we study the case of piecewise-Lipschitz functions with *dispersed* discontinuities that do not concentrate in any small compact subset of the domain [11]. Such functions arise frequently in *data-driven algorithm design*, where the goal is to learn the optimal parameter settings of algorithms for difficult (often NP-Hard) problems over a distribution or sequence of instances [4]; for example, a small change to the metric in cluster linkage can lead to a discontinuous change in the classification error [7]. In this paper, we also demonstrate that such losses are relevant in the setting of adversarial robustness, where we introduce a novel online formulation. For both cases, the associated problems are often solved across many time periods or for many different problem domains, resulting in natural multi-task structure that might improve performance. To the best of our knowledge, ours is the first theoretical study of meta-learning in both of these application settings.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

In the single-task setting the problem of learning dispersed functions can be solved using simple methods such as the exponentially-weighted forecaster. To design an algorithm for learning to initialize online learners in this setting, we propose a method that optimizes a sequence of data-dependent upper-bounds on the within-task regret [29]. The result is an averaged bound that improves upon the regret of the single-task exponential forecaster so long as there exists an initial distribution that can compactly contain many of the within-task optima of the different tasks. Designing the meta-procedure is especially challenging in our setting because it involves online learning over a set of distributions on the domain. To handle this we study a "prescient" form of the classic follow-the-regularized leader (FTRL) scheme that is run over an unknown discretization; we then show the existence of another algorithm that plays the same actions but uses only known information, thus attaining the same regret while being practical to implement.

To demonstrate the usefulness of our method, we study this algorithm in two settings.

Multi-task data-driven algorithm design. We consider data-driven tuning of the parameters of combinatorial optimization algorithms for hard problems such as knapsack and clustering. The likely intractability of these problems on worst case instances have led to several approaches to study them in more realistic settings, such as smoothed analysis [44] and data-driven algorithm configuration [4]. Our setting is more realistic than those considered in prior work. It is more challenging than learning from iid instances [27, 12], but at the same time less pessimistic than online learning over adversarial problem instances [11] as it allows us to leverage similarity of problem instances coming from different but related distributions. We instantiate our bounds theoretically on several problems where the cost functions are piecewise-constant in the tuned parameters, allowing our meta-procedure to learn the right initial distribution for exponential forecasters. This includes well-known combinatorial optimization problems like finding the maximum weighted independent set (MWIS) of vertices on a graph, solving quadratic programs with integer constraints using algorithms based on the celebrated Goemans-Williamson algorithm, and mechanism design for combinatorial auctions. Then we consider experimentally the problem of tuning the right α for the α-Lloyd's family of clustering algorithms [15]. In experimental evaluations on two datasets—a synthetic Gaussian mixture model and the well-known Omniglot dataset from meta-learning [33]—our meta-procedure leads to improved clustering accuracy compared to single-task learning to cluster. We also study our results for a family of greedy algorithms for the knapsack problem introduced by [27] and obtain similar results for a synthetic dataset. The results holds for both one-shot and five-shot tasks.

Online robust meta-learning. The second instantiation of our meta-learning procedure is to a new notion of adversarial robustness for the setting of online learning, where our results imply robust meta-learning in the presence of outliers. In this setting, the adversary can make (typically small) modifications to some example $x \in \mathcal{X}$, which can result in potentially large changes to the corresponding loss value $l_h(x)$, where $h \in \mathcal{H}$ is our hypothesis. For instance, consider the well-studied setting of adversarial examples for classification of images using deep neural networks [36, 17]. Given a neural network f, the adversary can perturb a datapoint x to a point x', say within a small L_p-ball around x, such that $f(x) = f(x')$ but the true label of x' does not match x, and therefore $l_f(x) \neq l_f(x')$. In general, under the adversarial influence, we observe a *perturbed loss* function $\tilde{l}_h(x) = l_h(x) + a_h(x)$. Typically we are interested in optimizing both the perturbed loss $\tilde{l}_h(x)$, i.e. measuring performance relative to optimum for adversarially perturbed losses, and the *true loss* $l_h(x)$ (performance on the unobserved, unperturbed loss). For example, in the online learning setting, [1] consider perturbed loss minimization for linear dynamical systems, while [41] look at true $\{0, 1\}$ loss minimization in the presence of adversarial noise. Our approach ensures that regret for both the perturbed and true loss are small, for piecewise-Lipschitz but dispersed adversaries.

2 Preliminaries and initialization-dependent learning of dispersed functions

In this section we introduce our setup and notation for online learning of piecewise-Lipschitz functions in a multi-task environment. We then generalize existing results for the single-task setting in order to obtain within-task regret bounds that depend on both the initialization and the task data. This is critical for both defining a notion of task similarity and devising a meta-learning procedure.

Meta-learning setup Following past setups [2, 21, 29], for some $T, m > 0$ and all $t \in [T]$ and $i \in [m]$ we consider a meta-learner faced with a sequence of Tm loss functions $\ell_{t,i} : C \mapsto [0, 1]$ over a compact subset $C \subset \mathbb{R}^d$ that lies within a ball $\mathcal{B}(\rho, R)$ of radius R around some point $\rho \in \mathbb{R}^d$.

Algorithm 1 Exponential Forecaster

Input: step size parameter $\lambda \in (0, 1]$, initialization $w : C \to \mathbb{R}_{\geq 0}$.
Initialize $w_1 = w$
for $i = 1, 2, \ldots, m$ **do**
  $W_i := \int_C w_i(\rho) d\rho$
  Sample ρ_i with probability proportional to $w_i(\rho_i)$, i.e. with probability $p_i(\rho_i) = \frac{w_i(\rho_i)}{W_i}$
  Suffer $\ell_i(\rho_i)$ and observe $\ell_i(\cdot)$
  For each $\rho \in C$, set $w_{i+1}(\rho) = e^{-\lambda \ell_i(\rho)} w_i(\rho)$

Here we used the notation $[n] = \{1, \ldots, n\}$. Before each loss function $\ell_{t,i}$ the meta-learner must pick an element $\rho_{t,i} \in C$ before then suffering a loss or cost $\ell_{t,i}(\rho_{t,i})$. For a fixed t, the subsequence $\ell_{t,1}, \ldots, \ell_{t,m}$ defines a **task** for which we expect a single element $\rho_t^* \in C$ to do well, and thus we will use the **within-task regret** on task t to describe the quantity

$$\mathbf{R}_{t,m} = \sum_{i=1}^{m} \ell_{t,i}(\rho_{t,i}) - \ell_{t,i}(\rho_t^*) \quad \text{where} \quad \rho_t^* \in \arg\min_{\rho \in C} \sum_{i=1}^{m} \ell_{t,i}(\rho) \tag{1}$$

In the single-task setting the goal is usually to show that $R_{t,m}$ is sublinear in m, i.e. that the average loss decreases with more rounds. A key point here is that the functions we consider can have numerous global optima. In this work we will assume, after going through the m rounds of task t, that we have oracle access to a single fixed optimum for t, which we will refer to using ρ_t^* and use in both our algorithm and to define the task-similarity. Note that in the types of applications we are interested in—piecewise-Lipschitz functions—the complexity of computing optima scales with the number of discontinuities. In the important special case of piecewise-constant functions, this dependency becomes logarithmic [19]. Thus this assumption does not affect the usefulness of the result.

Our goal will be to improve the guarantees for regret in the single-task case by using information obtained from solving multiple tasks. In particular, we expect average performance across tasks to improve as we see more tasks; to phrase this mathematically we define the **task-averaged regret**

$$\bar{\mathbf{R}}_{T,m} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{R}_{t,m} = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{m} \ell_{t,i}(\rho_{t,i}) - \ell_{t,i}(\rho_t^*) \tag{2}$$

and claim improvement over single-task learning if in the limit of $T \to \infty$ it is smaller than $\mathbf{R}_{t,m}$. Note that for simplicity in this work we assume all tasks have the same number of rounds within-task, but as with past work our results are straightforward to extend to the more general setting.

Learning piecewise-Lipschitz functions We now turn to our target functions and within-task algorithms for learning them: piecewise-Lipschitz losses, i.e. functions that are L-Lipschitz w.r.t. the Euclidean norm everywhere except on measure zero subsets of the space; here they may have arbitrary jump discontinuities so long they still bounded between $[0, 1]$. Apart from being a natural setting of interest due to its generality compared to past work on meta-learning, this class of functions has also been shown to have important applications in data-driven algorithm configuration [11]; there these functions represent the cost, e.g. an objective value or time-complexity, of algorithms for difficult problems such as integer programming, auction design, and clustering.

This literature has also shown lower bounds demonstrating that no-regret learning piecewise-Lipschitz function is impossible in general, necessitating assumptions about the sequence. One such condition is *dispersion*, which requires that the discontinuities are not too concentrated.

Definition 2.1 ([11]). *The sequence of random loss functions $\ell_1, \ldots, \ell_m$ is said to be β-***dispersed** *with Lipschitz constant L if, for all m and for all $\epsilon \geq m^{-\beta}$, we have that, in expectation over the randomness of the functions, at most $\tilde{O}(\epsilon m)$ functions (the soft-O notation suppresses dependence on quantities beside ϵ, m and β, as well as logarithmic terms) are not L-Lipschitz for any pair of points at distance ϵ in the domain $\mathbb{C}$. That is, for all m and for all $\varepsilon \geq m^{-\beta}$,*

$$\mathbb{E}\left[\max_{\substack{\rho, \rho' \in \mathbb{C} \\ \|\rho - \rho'\|_2 \leq \epsilon}} \left\| \left\{ i \in [m] \mid \ell_i(\rho) - \ell_i(\rho') > L\|\rho - \rho'\|_2 \right\} \right| \right] \leq \tilde{O}(\epsilon m) \tag{3}$$

Assuming a sequence of m β-dispersed loss functions and initial distribution w_1 set to the uniform distribution over C and optimize the step size parameter, the exponential forecaster presented in Algorithm 1 achieves sublinear regret $\tilde{O}(\sqrt{dm\log(Rm)}+(L+1)m^{1-\beta})$. While this result achieves a no-regret procedure, its lack of dependence on both the task-data and on the chosen initialization makes it difficult to meta-learn. In the following theorem, we generalize the regret bound for the exponential forecaster to make it data-dependent and hyperparameter-dependent:

Theorem 2.1. *Let $\ell_1,\ldots,\ell_m : C \mapsto [0,1]$ be any sequence of piecewise L-Lipschitz functions that are β-dispersed. Suppose $C \subset \mathbb{R}^d$ is contained in a ball of radius R. The exponentially weighted forecaster (Algorithm 1) has expected regret $\mathbf{R}_m \leq m\lambda + \frac{\log(1/Z)}{\lambda} + \tilde{O}((L+1)m^{1-\beta})$, where $Z = \frac{\int_{\mathcal{B}(\rho^*,m^{-\beta})} w(\rho)d\rho}{\int_C w(\rho)d\rho}$ for ρ^* the optimal action in hindsight.*

The proof of this result adapts past analyses of Algorithm 1; setting step-size λ appropriately recovers the previously mentioned bound. The new bound is useful due to its explicit dependence on both the initialization w and the optimum in hindsight via the $\log(1/Z)$ term. Assuming w is a (normalized) distribution, this effectively measures the overlap between the chosen initialization and a small ball around the optimum; we thus call $-\log Z = -\log \frac{\int_{\mathcal{B}(\rho^*,m^{-\beta})} w(\rho)d\rho}{\int_C w(\rho)d\rho}$ the **negative log-overlap** of initialization $w(.)$ with the optimum ρ^*.

We also obtain an asymptotic lower bound on the expected regret of any algorithm by extending the argument of [10] to the multi-task setting. We show that for finite D^* we must suffer $\tilde{\Omega}(m^{1-\beta})$ regret, which limits the improvement we can hope to achieve from task-similarity.

Theorem 2.2. *There is a sequence of piecewise L-Lipschitz β-dispersed functions $\ell_{i,j} : [0,1] \mapsto [0,1]$, whose optimal actions in hindsight $\arg\min_\rho \sum_{i=1}^m l_{t,i}(\rho)$ are contained in some fixed ball of diameter D^*, for which any algorithm has expected regret $\mathbf{R}_m \geq \tilde{\Omega}(m^{1-\beta})$.*

2.1 Task-similarity

Before proceeding to our discussion of meta-learning, we first discuss what we might hope to achieve with it; specifically, we consider what a reasonable notion of task-similarity is in this setting. Note that the Theorem 2.1 regret bound has three terms, of which two depend on the hyperparameters and the last is due to dispersion and cannot be improved via better settings. Our focus will thus be on improving the first two terms, which are the dominant ones due to the dependence on the dimensionality and the distance from the initialization encoded in the negative log overlap. In particular, when the initialization is the uniform distribution then this quantity depends inversely on the size of a small ball around the optimum, which may be quite small. Via meta-learning we hope to assign more of the probability mass of the initializer to areas close to the optimum, which will decrease these terms. On average, rather than a dependence on the volume of a small ball we aim to achieve a dependence on the **average negative log-overlap**

$$V^2 = -\min_{w:C\mapsto\mathbb{R}_{\geq 0},\int_C w(\rho)d\rho=1} \frac{1}{T}\sum_{t=1}^{T} \log \int_{\mathcal{B}(\rho_t^*,m^{-\beta})} w(\rho)d\rho \tag{4}$$

which can be much smaller if the task optima ρ_t^* are close together; for example, if they are the same then $V=0$, corresponding to assigning all the initial weight within the common ball $\mathcal{B}(\rho^*,m^{-\beta})$ around the shared optima. This is also true if $\mathrm{vol}(\cap_{t\in T}\mathcal{B}(\rho_t^*,m^{-\beta})) > 0$, as one can potentially initialize with all the weight in the intersection of the balls. On the other hand if $\mathrm{vol}(\cap_{t\in T}\mathcal{B}(\rho_t^*,m^{-\beta})) = 0$, $V>0$. For example, if a p-fraction of tasks have optima ρ_0 and the remaining at ρ_1 with $||\rho_0-\rho_1|| > 2m^{-\beta}$ the task similarity is given by the binary entropy function $V = H_b(p) = -p\log p - (1-p)\log(1-p)$.

The settings of Algorithm 1 that achieve the minimum in the definition of V are directly related to V itself: the optimal initializer is the distribution achieving V and the optimal step-size is $V/\sqrt{m}$. Note that while the explicit definition requires computing a minimum over a set of functions, the task-similarity can be computed using the discretization constructed in Section 3.1.

3 An algorithm for meta-learning the initialization and step-size

Having established a single-task algorithm and shown how its regret depends on the initialization and step-size, we move on to meta-learning these hyperparameters. Recall that our goal is to make the task-averaged regret (2) small, in particular to improve upon the baseline of repeatedly running Algorithm 1 from the uniform distribution, up to $o_T(1)$ terms that vanish as we see more tasks. This accomplishes the meta-learning goal of using multiple tasks to improve upon single-task learning.

In this paper, we use the strategy of running online learning algorithms on the data-dependent regret guarantees from above [29]. If we can do so with sublinear regret in T, then we will improve upon the single-task guarantees up to $o_T(1)$ terms, as desired. Specifically, we are faced with a sequence of regret-upper-bounds $U_t(w, v) = (v + f_t(w)/v)\sqrt{m} + g(m)$ on nonnegative functions w over C and positive scalars $v > 0$. Note that $g(m)$ cannot be improved via meta-learning, so we will focus on learning w and v. To do so, we run two online algorithms, one over the functions f_t and the other over $h_t(v) = v + f_t(w_t)/v$, where w_t is set by the first procedure. As shown in the following result, if both procedures have sublinear regret then our task-averaged regret will have the desired properties:

Theorem 3.1. *Assume each task $t \in [T]$ consists of a sequence of m β-dispersed piecewise L-Lipschitz functions $\ell_{t,i} : C \mapsto [0, 1]$. Let f_t and g be functions such that the regret of Algorithm 1 run with step-size $\lambda = v\sqrt{m}$ for $v > 0$ and initialization $w : C \mapsto \mathbb{R}_{\geq 0}$ is bounded by $U_t(w, v) = (v + f_t(w)/v)\sqrt{m} + g(m)$. Suppose we have a procedure that achieves $F_T(w)$ regret w.r.t. any $w : C \mapsto \mathbb{R}_{\geq 0}$ by playing actions $w_t : C \mapsto \mathbb{R}_{\geq 0}$ on f_t and another procedure that achieves $H_T(v)$ regret w.r.t. any $v > 0$ by playing actions $v_t > 0$ on $h_t(v) = v + f_t(w_t)/v$, where H_T is non-increasing on the positive reals. Then by setting $\rho_{t,i}$ using Algorithm 1 with step-size $v_t/\sqrt{m}$ and initialization w_t at each task t, for $w^* = \arg\min_{w:C\mapsto\mathbb{R}_{\geq 0}} \sum_{t=1}^T f_t(w)$ the optimal initialization and V the task-similarity* (4) *we get task-averaged regret bounded by*

$$\left(\frac{H_T(V)}{T} + \min\left\{\frac{F_T(w^*)}{VT}, 2\sqrt{F_T(w^*)/T}\right\} + 2V\right)\sqrt{m} + g(m) \tag{5}$$

This result is an analog of [29, Theorem 3.1] and follows by manipulating the definition of regret. It reduces the problem of obtaining a small task-averaged regret to solving two online learning problems, one to set the initialization and one to set the step-size. So long as both have sublinear regret then we will improve over single-task learning. In the next two sections we derive suitable procedures.

3.1 Meta-learning the initialization

The most technically challenging component of the procedure is learning the initialization. As discussed, this can be done via a no regret procedure for the sequence $f_t(w) = -\log \frac{\int_{\mathcal{B}(\rho_t^*, m^{-\beta})} w(\rho)d\rho}{\int_C w(\rho)d\rho}$. This is nontrivial as the optimization domain is a set of nonnegative functions or measures on the domain C. To handle this, we introduce some notations and abstractions. At each task t we face a function f_t associated with an unknown closed subset $C_t \subset C$—in particular $C_t = \mathcal{B}(\rho_t^*, m^{-\beta})$—with positive volume $\mathrm{vol}(C_t) > 0$ that is revealed after choosing $w_t : C \mapsto \mathbb{R}_{\geq 0}$ For each time t define the discretization $\mathcal{D}_t = \{D = \bigcap_{s\leq t} C_s^{(\mathbf{c}_{[s]})} : \mathbf{c} \in \{0, 1\}^t, \mathrm{vol}(D) > 0\}$ of C, where $C_t^{(0)} = C_t$ and $C_t^{(1)} = C\backslash C_t$. We will use elements of these discretizations to index nonnegative vectors in $\mathbb{R}_{\geq 0}^{|\mathcal{D}_t|}$; specifically, for any measure $w : C \mapsto \mathbb{R}_{\geq 0}$ let $\mathbf{w}(t) \in \mathbb{R}_{\geq 0}^{|\mathcal{D}_t|}$ denote the vector with entries $\mathbf{w}(t)_{[D]} = \int_D w(\rho)d\rho$ for $D \in \mathcal{D}_t$. Note that we will exclusively use p, q, v, w for measures, with v specifically referring to the uniform measure, i.e. $\mathbf{v}(t)_{[D]} = \mathrm{vol}(D)$. For convenience, for all real vectors $\mathbf{x}$ we will use $\hat{\mathbf{x}}$ to denote $\mathbf{p}/\|\mathbf{p}\|_1$. Finally, we abuse notation and remove the parentheses to refer those vectors associated with the final discretization, i.e. $\mathbf{v} = \mathbf{v}(T)$ and $\mathbf{w} = \mathbf{w}(T)$.

Now that we have this notation we can turn back to the functions we are interested in: $f_t(w) = -\log \frac{\int_{C_t} w(\rho)d\rho}{\int_C w(\rho)d\rho}$, where $C_t = \mathcal{B}(\rho_t^*, m^{-\beta})$. Observe that we can equivalently write this as $f_t(\mathbf{w}) = -\log\langle \mathbf{w}_t^*, \hat{\mathbf{w}}\rangle$, where $\mathbf{w}_{t[D]}^* = 1_{D\subset C_t}$; this translates our online learning problem from the domain of measures on C to the simplex on $|\mathcal{D}_T|$ elements. However, we cannot play in this domain explicitly as we do not have access to the final discretization $\mathcal{D}_T$, nor do we get access to $\mathbf{w}_t^*$ after task t, except implicitly via C_t. In this section we design a method that implicitly run an online convex optimization procedure over $\mathbb{R}_{\geq 0}^{|\mathcal{D}_T|}$ while explicitly playing probability measures $w : C \mapsto \mathbb{R}_{\geq 0}$.

Algorithm 2 Follow-the-Regularized-Leader (prescient form)

Input: discretization $\mathcal{D}_T$ of C, mixture parameter $\gamma \in [0,1]$, step-size $\eta > 0$
Initialize $\mathbf{w}_1 = \hat{\mathbf{v}}$
for $t = 1, 2, \ldots, T$ **do**
  Play $\mathbf{w}_t$ and suffer $f_t(\mathbf{w}_t) = -\log\langle \mathbf{w}_t^*, \mathbf{w}_t\rangle$.
  Observe f_t.
  Update $\mathbf{w}_{t+1} = \arg\min_{\|\mathbf{w}\|_1=1, \mathbf{w}\geq\gamma\hat{\mathbf{v}}} D_{KL}(\mathbf{w}||\hat{\mathbf{v}}) + \eta\sum_{s\leq t} f_s(\mathbf{w})$

As the functions f_t are exp-concave, one might first consider applying a method attaining logarithmic regret on such losses [28, 37]; however, such algorithms have regret that depends linearly on the dimension, which in our case is poly(T). We thus turn to the the follow-the-regularized-leader (FTRL) family of algorithms, which in the case of entropic regularization are well-known to have regret logarithmic in the dimension [43]. In Algorithm 2 we display the pseudo-code of a modification with regularizer $D_{KL}(\cdot||\hat{\mathbf{v}})$, where recall $\mathbf{v}$ is the vector of volumes of the discretization $\mathcal{D}_T$ of C, and we constrain the played distribution to have measure at least $\gamma\hat{\mathbf{v}}_{[D]}$ over every set $D \in \mathcal{D}_T$. While this requires knowing the discretization $\mathcal{D}_T$ of C in advance, the following lemma shows that we can run the procedure knowing only the discretization $\mathcal{D}_t$ after task t by simply minimizing the same objective over distributions discretized on $\mathcal{D}_t$. This crucially depends on the re-scaling of the entropic regularizer by $\hat{\mathbf{v}}$ (i.e. the uniform distribution over C) and the fact that $\mathbf{w}_t^* \in \{0,1\}^{|\mathcal{D}_T|}$.

Lemma 3.1. *Let $w : C \mapsto \mathbb{R}_{\geq 0}$ be the probability measure corresponding to the minimizer*

$$\mathbf{w} = \operatorname*{arg\,min}_{\|\mathbf{q}\|_1=1, \mathbf{q}\geq\gamma\hat{\mathbf{v}}} D_{KL}(\mathbf{q}||\hat{\mathbf{v}}) - \eta\sum_{s\leq t}\log\langle \mathbf{w}_s^*, \mathbf{q}\rangle \tag{6}$$

and let $\tilde{w} : C \mapsto \mathbb{R}_{\geq 0}$ be the probability measure corresponding to the minimizer

$$\tilde{\mathbf{w}}(t) = \operatorname*{arg\,min}_{\|\mathbf{q}\|_1=1, \mathbf{q}\geq\gamma\hat{\mathbf{v}}(t)} D_{KL}(\mathbf{q}||\hat{\mathbf{v}}(\mathbf{t})) - \eta\sum_{s\leq t}\log\langle \mathbf{w}_s^*(t), \mathbf{q}\rangle \tag{7}$$

Then $\mathbf{w} = \tilde{\mathbf{w}}$.

We can thus move on to proving a regret guarantee for Algorithm 2. This follows from Jensen's inequality together with standard results for FTRL once we show that the loss functions are $\frac{1}{\gamma\,\mathrm{vol}(C_t)}$-Lipschitz over the constrained domain, yielding the following guarantee for Algorithm 2:

Theorem 3.2. *Algorithm 2 has regret w.r.t. $\mathbf{w}^* \in \arg\min_{\|\mathbf{w}\|_1=1, \mathbf{w}\geq\mathbf{0}}\sum_{t=1}^T f_t(\mathbf{w})$ bounded by*

$$\frac{1-\gamma}{\eta}D_{KL}(\mathbf{w}^*||\hat{\mathbf{v}}) + \frac{\eta}{\gamma^2}\sum_{t=1}^T\frac{1}{(\mathrm{vol}(C_t))^2} + \gamma\sum_{t=1}^T\log\frac{1}{\mathrm{vol}(C_t)} \tag{8}$$

Setting $\gamma^2 = GB/\sqrt{T}$ and $\eta^2 = \frac{B^2\gamma^2}{TG^2}$, where $B^2 = D_{KL}(\mathbf{w}^*||\hat{\mathbf{v}})$ and $G^2 = \frac{1}{T}\sum_{t=1}^T\frac{1}{(\mathrm{vol}(C_t))^2}$, yields sublinear regret $\tilde{O}(\sqrt{BG}T^{\frac{3}{4}})$.

Proof. Algorithm 2 is standard FTRL with regularizer $\frac{1}{\eta}D_{KL}(\cdot||\hat{\mathbf{v}})$, which has the same Hessian as the standard entropic regularizer over the simplex and is thus $\frac{1}{\eta}$-strongly-convex w.r.t. $\|\cdot\|_1$ [43, Example 2.5]. Applying Jensen's inequality, the standard regret bound for FTRL [43, Theorem 2.11] together with the Lipschitz guarantee of Claim B.1, and Jensen's inequality again yields the result:

$$\begin{aligned}
\sum_{t=1}^T f_t(\mathbf{w}_t) - f_t(\mathbf{w}^*) &= \sum_{t=1}^T f_t(\mathbf{w}_t) - (1-\gamma)f_t(\mathbf{w}^*) - \gamma f_t(\hat{\mathbf{v}}) + \gamma(f_t(\hat{\mathbf{v}}) - f_t(\mathbf{w}^*)) \\
&\leq \sum_{t=1}^T f_t(\mathbf{w}_t) - f_t(\gamma\hat{\mathbf{v}} + (1-\gamma)\mathbf{w}^*) + \gamma\log\frac{\langle \mathbf{w}_t^*, \mathbf{w}^*\rangle}{\langle \mathbf{w}_t^*, \hat{\mathbf{v}}\rangle} \\
&\leq \frac{1}{\eta}D_{KL}(\gamma\hat{\mathbf{v}} + (1-\gamma)\mathbf{w}^*||\hat{\mathbf{v}}) + \frac{\eta}{\gamma^2}\sum_{t=1}^T\frac{1}{(\mathrm{vol}(C_t))^2} + \gamma\sum_{t=1}^T\log\frac{1}{\mathrm{vol}(C_t)} \\
&\leq \frac{1-\gamma}{\eta}D_{KL}(\mathbf{w}^*||\hat{\mathbf{v}}) + \frac{\eta}{\gamma^2}\sum_{t=1}^T\frac{1}{(\mathrm{vol}(C_t))^2} + \gamma\sum_{t=1}^T\log\frac{1}{\mathrm{vol}(C_t)}
\end{aligned}$$

□

Since the regret is sublinear in T, this result satisfies our requirement for attaining asymptotic improvement over single-task learning via Theorem 3.1. However, there are several aspects of this bound that warrant some discussion. The first is the rate of $T^{\frac{3}{4}}$, which is less sublinear than the standard $\sqrt{T}$ and certainly the $\log T$ regret of exp-concave functions. However, the functions we face are (a) non-Lipschitz and (b) over a domain that has dimensionality $\Omega(T)$; both violate conditions for good rates in online convex optimization [28, 43], making our problem much more difficult.

A more salient aspect is the dependence on $B^2 = D_{KL}(\mathbf{w}^*||\hat{\mathbf{v}})$, effectively the negative entropy of the optimal initialization. This quantity is in-principle unbounded but is analogous to standard online convex optimization bounds that depend on the norm of the optimum, which in e.g. the Euclidean case are also unbounded. In our case, if the optimal distribution is highly concentrated on a very small subset of the space it will be difficult to compete with. Note that our setting of η depends on knowing or guessing B; this is also standard but is certainly a target for future work to address. For example, past work on parameter-free algorithms has solutions for optimization over the simplex [38]; however, it is unclear whether this is straightforward to do while preserving the property given by Lemma 3.1 allowing us to implicitly work with an unknown discretization. A more reasonable approach may be to compete only with smooth measures that only assign probability at most $\kappa \operatorname{vol}(D)$ to any subset $D \subset C$ for some constant $\kappa \geq 1$; in this case we will simply have B bounded by $\log \kappa$.

A final issue is the dependence on $\sqrt{G}$, which is bounded by the reciprocal of the smallest volume $\operatorname{vol}(C_t)$, which in the dispersed case is roughly $O(m^{\beta d})$; this means that the task-averaged regret will have a term that, while decreasing as we see additional tasks, is *increasing* in the number of within-task iterations and the dispersion parameter, which is counter-intuitive. It is also does so exponentially in the dimension. Note that in the common algorithm configuration setting of $\beta = 1/2$ and $d = 1$ this will simply mean that for each task we suffer an extra $o_T(1)$ loss at each within-task round, a quantity which vanishes asymptotically.

3.2 Meta-learning the step-size

In addition to learning the initialization, Theorem 3.1 requires learning the task-similarity to set the within-task step-size $\lambda > 0$. This involves optimizing functions of form $h_t(v) = v + f_t(w_t)/v$. Since we know that the measures w_t are lower-bounded in terms of γ, we can apply a previous result [29] that solves this by running the EWOO algorithm [28] on the modified sequence $v + \frac{f_t(w_t)+\varepsilon^2}{v}$:

Corollary 3.1. *For any $\varepsilon > 0$, running the EWOO algorithm on the modified sequence $v + \frac{f_t(w)+\varepsilon^2}{v}$ over the domain $[\varepsilon, \sqrt{D^2 - \log\gamma + \varepsilon^2}]$, where $D^2 \geq \frac{1}{T}\sum_{t=1}^{T} \log \frac{1}{\operatorname{vol}(C_t)}$, attains regret*

$$\min\left\{\frac{\varepsilon^2}{v^*}, \varepsilon\right\} T + \frac{\sqrt{D^2 - \log\gamma}}{2} \max\left\{\frac{D^2 - \log\gamma}{\varepsilon^2}, 1\right\} (1 + \log(T+1)) \tag{9}$$

on the original sequence $h_t(v) = v + f_t(w)/v$ for all $v^* > 0$.

Setting $\varepsilon = 1/\sqrt[4]{T}$ gives a guarantee of form $\tilde{O}((\min\{1/v^*, \sqrt[4]{T}\})\sqrt{T})$. Note this rate might be improvable by using the fact that v is lower-bounded due to the γ-constraint; however, we do not focus on this since this component is not the dominant term in the regret. In fact, because of this we can adapt a related method that simply runs follow-the-leader (FTL) on the same modified sequence [29] without affecting the dominant terms in the regret:

Corollary 3.2. *For any $\varepsilon > 0$, running the FTL algorithm on the modified sequence $v + \frac{f_t(w)+\varepsilon^2}{v}$ over the domain $[\varepsilon, \sqrt{D^2 - \log\gamma + \varepsilon^2}]$, where $D^2 \geq \frac{1}{T}\sum_{t=1}^{T} \log \frac{1}{\operatorname{vol}(C_t)}$, attains regret*

$$\min\left\{\frac{\varepsilon^2}{v^*}, \varepsilon\right\} T + 2\sqrt{D^2 - \log\gamma} \max\left\{\frac{(D^2 - \log\gamma)^{\frac{3}{2}}}{\varepsilon^3}, 1\right\} (1 + \log(T+1)) \tag{10}$$

on the original sequence $h_t(v) = v + f_t(w)/v$ for all $v^* > 0$.

Setting $\varepsilon = 1/\sqrt[5]{T}$ gives a guarantee of form $\tilde{O}((\min\{1/v^*, \sqrt[5]{T}\})T^{\frac{3}{5}})$. The alternatives are described in pseudocode at the bottom of Algorithm 3; while the guarantee of the FTL-based approach is worse, it is almost as simple to compute as the task-similarity and does not require integration, making it easier to implement.

Algorithm 3 Meta-learning the parameters of the exponential forecaster (Algorithm 1). Recall that $\mathbf{p}(t)$ refers to the time-t discretization of the measure $p : C \mapsto \mathbb{R}_{\geq 0}$ (c.f. Section 3.1).

Input: domain $C \subset \mathbb{R}^d$, dispersion $\beta > 0$, step-size $\eta > 0$, constraint parameter $\gamma \in [0, 1]$, offset parameter $\varepsilon > 0$, domain parameter $D > 0$.
Initialize w_1 to the uniform measure on C and set $\lambda_1 = \frac{\varepsilon + \sqrt{D^2 + \varepsilon^2 - \log \gamma}}{2\sqrt{m}}$.
for task $t = 1, 2, \ldots, T$ **do**
  Run Algorithm 1 with initialization w_t and step-size λ_t and obtain task-t optimum $\rho_t^* \in C$.
  Set $w_t^* = 1_{\mathcal{B}(\rho_t^*, m^{-\beta})}$ to be the function that is 1 in the $m^{-\beta}$-ball round ρ_t^* and 0 elsewhere.
  Set w_{t+1} to $\mathbf{w}_{t+1}(t) = \arg\min_{\|\mathbf{w}\|_1 = 1, \mathbf{w} \geq \gamma \hat{\mathbf{v}}(t)} D_{KL}(\mathbf{w} || \hat{\mathbf{v}}(t)) - \eta \sum_{s \leq t} \log \langle \mathbf{w}_s^*(t), \mathbf{w} \rangle$.
  if using EWOO **then**
    Define $\mu_t(x) = \exp\left(-\alpha\left(tx + \frac{t\varepsilon^2 - \sum_{s \leq t} \log \langle \mathbf{w}_s^*(s), \mathbf{w}_s(s) \rangle}{x}\right)\right)$ for $\alpha = \frac{2}{D} \min\left\{\frac{\varepsilon^2}{D^2}, 1\right\}$.
    Set $\lambda_{t+1} = \frac{\int_\varepsilon^{\sqrt{D^2 + \varepsilon^2 - \log \gamma}} x \mu_t(x) dx}{\sqrt{m} \int_\varepsilon^{\sqrt{D^2 + \varepsilon^2 - \log \gamma}} \mu_t(x) dx}$.
  else
    Set $\lambda_{t+1} = \sqrt{\frac{\sum_{s \leq t} \varepsilon^2 - \log \langle \mathbf{w}_s^*(s), \mathbf{w}_s(s) \rangle}{tm}}$.

3.3 Putting the two together

We now combine our methods for the initialization and step-size in Algorithm 3 to meta-learn the parameters of the exponential forecaster. This yields a task-averaged regret bound via Theorem 3.1:

Theorem 3.3. *Define* $B^2 = D_{KL}(\mathbf{w}^* || \hat{\mathbf{v}})$, $G^2 = \frac{1}{T} \sum_{t=1}^T \frac{1}{(\mathrm{vol}(C_t))^2}$, *and* $D^2 \geq \frac{1}{T} \sum_{t=1}^T \log \frac{1}{\mathrm{vol}(C_t)} = O(\beta d \log m)$. *Then Algorithm 3 with* η, γ *set as in Theorem 3.2 and* $\varepsilon = 1/\sqrt[4]{T}$ *(if using EWOO) or* $1/\sqrt[5]{T}$ *(otherwise) yields task-averaged regret*

$$\tilde{O}\left(\min\left\{\frac{\sqrt{BG}}{V\sqrt[4]{T}}, \frac{\sqrt[4]{BG}}{\sqrt[8]{T}}\right\} + 2V\right)\sqrt{m} + g(m) \tag{11}$$

As in past meta-learning work this achieves the goal of adapting to the task-similarity, attaining asymptotic average regret of $2V\sqrt{m} + O(m^{-\beta})$, where we substitute for the dispersion term g and V^2 is the task-similarity encoding the average probability mass assigned to different task balls by the optimal initialization. We include the minimum of two rates in the bound: $1/\sqrt[4]{T}$ if the task-similarity is a constant $\Theta_T(1)$ and $1/\sqrt[8]{T}$ if it is extremely small. As discussed, this reflects the difficulty of our meta-problem, in which we are optimizing non-smooth functions over distributions; in contrast, past meta procedures take advantage of nice properties of Bregman divergences to obtain faster rates [29].

4 Meta-learning for data-driven algorithm design

We demonstrate the utility of our bounds in a series of applications across data-driven algorithm design and robust learning. This section focuses on the former and demonstrates how our results imply guarantees for meta-learning the tuning of solvers for difficult combinatorial problems. We also demonstrate the practical utility of our approach for tuning clustering on real and synthetic datasets.

4.1 Instantiations for tuning combinatorial optimization algorithms

Algorithm configuration for combinatorial optimization algorithms involves learning algorithm parameters from multiple instances of combinatorial problems [27, 12, 4]. For well-known problems like MWIS (maximum weighted independent set), IQP (integer quadratic programming) and mechanism design for auctions, the algorithmic performance on a fixed instance is typically a piecewise Lipschitz function of the algorithm parameters. Prior work has looked at learning these parameters in the distributional setting (i.e. assuming iid draws of problem instances) [12] or the online setting where the problem instances may be adversarially drawn [11, 10]. On the other hand, instantiating our results for these problems provide upper bounds for much more realistic settings where different tasks may be related and our bounds improve with this relatedness.

We demonstrate how to apply our results to combinatorial problems under mild smoothness assumptions. The key idea is to show that if inputs come from a smooth distribution, the performance is dispersed (as a sequence of functions in the algorithm parameters). We demonstrate this for the greedy algorithm for the knapsack problem and for initialization in k-center clustering. Similar results may be obtained for other problems, e.g. MWIS, IQP, and auction design (c.f. Appendix C).

Greedy Knapsack. Knapsack is a well-known NP-complete problem. We are given a knapsack with capacity $\texttt{cap}$ and items $i \in [m]$ with sizes w_i and values v_i. The goal is to select a subset S of items to add to the knapsack such that $\sum_{i \in S} w_i \leq \texttt{cap}$ while maximizing the total value $\sum_{i \in S} v_i$ of selected items. The greedy heuristic to add items in decreasing order of v_i/w_i gives a 2-approximation. We consider a generalization to use v_i/w_i^ρ proposed by [27] for $\rho \in [0, 10]$. For example, for the value-weight pairs $\{(0.99, 1), (0.99, 1), (1.01, 1.01)\}$ and capacity $\texttt{cap} = 2$ the classic heuristic $\rho = 1$ gives value 1.01 but using $\rho = 3$ gives the optimal value 1.98. We can learn the optimal ρ from similar tasks, and obtain the following formal guarantees (proof in Appendix C).

Theorem 4.1. *Consider instances of the knapsack problem given by bounded weights $w_{i,j} \in [1, C]$ and κ-bounded independent values $v_{i,j} \in [0, 1]$ for $i \in [m], j \in [T]$. Then the task-averaged regret for learning the parameter ρ for the greedy heuristic family above is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

k-center clustering. We consider the α-Llyod's algorithm family introduced in [15]. In the seeding phase, each point x is sampled with probability $\propto \min_{c \in C} d(v, c)^\alpha$, where $d(\cdot, \cdot)$ is the distance metric and C is the set of centers chosen so far. The family contains an algorithm for each $\alpha \in [0, \infty) \cup \infty$, and includes popular clustering heuristics like randomly initialized k-means ($\alpha = 0$), k-means++ ($\alpha = 2$) and farthest-first traversal ($\alpha = \infty$). Performance is measured using the Hamming distance to the best clustering and is a piecewise constant function of α. Our result can be instantiated for this problem even without smoothness by leveraging the randomness of the clustering algorithm.

Theorem 4.2. *Consider instances of the k-center clustering problem on n points, with Hamming loss $l_{i,j}$ for $i \in [m], j \in [T]$ against some (unknown) ground truth. Then the task-averaged regret for learning the parameter α for the α-Lloyd's algorithm family [15] is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

4.2 Experiments for greedy knapsack and k-center clustering

Next we demonstrate our meta-initialization algorithm empirically on knapsack and k-center clustering. We design experiments on real and simulated data that show the usefulness of our techniques in learning a sequence of piecewise-Lipschitz functions. For knapsack, we generate a synthetic dataset of instances as follows. For each problem instance of each task, we have $\texttt{cap} = 100$ and $m = 50$. We have 10 'heavy' items with $w_i \sim \mathcal{N}(27, 0.5)$ and $v_i \sim \mathcal{N}(27, 0.5)$, and 40 items with $w_i \sim \mathcal{N}(19 + w_t, 0.5)$ and $v_i \sim \mathcal{N}(18, 0.5)$, where $w_t \in [0, 2]$ is task-dependent.

We also consider the α-Lloyd's algorithm family introduced in [15]. The performance of the algorithm is measured using the Hamming loss relative to the best clustering and is a piecewise constant function of α. We can compute the pieces for $\alpha \in [0, 10]$ by iteratively computing the subset of parameter values where a candidate point can be the next center. We use the small split of the *Omniglot* dataset [33], and create clustering tasks by drawing random samples consisting of five characters each, where four characters are constant throughout. We also create a Gaussian mixture binary classification dataset where each class is a 2D diagonal Gaussian distribution consisting of 100 points each with variance σ and 2σ and centers $(0, 0)$ and $(d\sigma, 0)$. We pick $d \in [2, 3]$ to create different tasks.

We learn using 30 instances each of 10 training tasks and evaluate average loss over 5 test tasks. We use 100 trials to average over the randomization of the clustering algorithm and the exponential forecaster algorithm. We perform meta-initialization with parameters $\gamma = \eta = 0.01$ (no hyperparameter search performed). The step-size is set to minimize the regret term in Theorem 2.1, and not meta-learned.

The relative improvement in task-averaged regret due to meta-learning in our formal guarantees depend on the task-similarity V and how it compares to the dispersion-related $O(m^{1-\beta})$ term, and can be significant when the latter is small. Our results in Table 1 show that meta-learning an initialization, i.e. a distribution over the algorithm parameter, for the exponential forecaster in this setting yields improved performance on each dataset. We observe this for both the one-shot and five-shot settings, i.e. the number of within-task iterations of the test task are one and five, respectively. The benefit of meta-learning is most pronounced for the Gaussian mixture (well-dispersed and similar tasks), and gains for Omniglot may increase with more tasks (dispersed but less similar tasks). For knapsack, the relative gains are smaller (similar tasks, but less dispersed). See Appendix D for further experiments.

Table 1: Effect of meta-initialization on few-shot learning of algorithmic parameters. Performance is as a fraction of the average value (Hamming accuracy, or knapsack value) of the offline optimum.

Dataset	Omniglot		Gaussian Mixture		Knapsack	
	One-shot	Five-shot	One-shot	Five-shot	One-shot	Five-shot
Single task	$88.67 \pm 0.47\%$	$95.02 \pm 0.19\%$	$90.10 \pm 1.10\%$	$91.43 \pm 0.44\%$	$84.74 \pm 0.29\%$	$98.89 \pm 0.17\%$
Meta-initialized	$89.65 \pm 0.49\%$	$96.05 \pm 0.15\%$	$95.76 \pm 0.60\%$	$96.39 \pm 0.27\%$	$85.66 \pm 0.57\%$	$99.12 \pm 0.15\%$

5 Robust online meta-learning

In online learning, we aim to minimize a sequence of losses and want to perform well relative to the optimum in hindsight. It is possible for the observed losses to be noisy on some inputs, either naturally or due to an adversary. We explore the conditions under which robustness to adversarial influence (i.e. outlier injection) is possible, which is common in meta-learning with diverse sources.

Setup: At round i, we play x_i, observe perturbed loss $\tilde{l}_i : \mathcal{X} \to [0,1]$ which is set by the adversary by modifying the true loss $l_i : \mathcal{X} \to [0,1]$ using an *attack function* $a_i : \mathcal{X} \to [0,1]$ such that $\tilde{l}_i = l_i + a_i$ and may be non-Lipschitz, and suffer perturbed loss $\tilde{l}_i(x_i)$ and true loss $l_i(x_i)$. We seek to minimize regret relative to best fixed action in hindsight, i.e. $\tilde{R}_m = \sum_{i=1}^m \tilde{l}_i(x_i) - \min_{x \in \mathcal{X}} \sum_{i=1}^m \tilde{l}_i(x)$ for the perturbed loss and regret $R_m = \sum_{i=1}^m l_i(x_i) - \min_{x \in \mathcal{X}} \sum_{i=1}^m l_i(x)$ for the true loss.

No regret can be achieved provided the adversary distribution is sufficiently smooth, i.e. satisfies β-dispersion for some $\beta > 0$, as this corresponds to online optimization of the perturbed loss function. We can show this for both perturbed and true loss. The perturbed loss guarantee is immediate from standard results on online learning of piecewise Lipschitz functions [11, 10]. For the true loss, we can achieve no regret if the adversary perturbation a_i is limited to small balls and the centers of the balls are dispersed, which we capture using the following definition.

Definition 5.1 (δ-bounded, β_a-dispersed attack)**.** *An attack function a_i is δ-bounded if there exists a ball $\mathcal{B}(x_a, \delta)$ of radius δ such that $a_i(x) = 0$ for each $x \in \mathcal{X} \setminus \mathcal{B}(x_a, \delta)$. x_a is called a center c_{a_i} for attack a_i. A sequence of attack functions $a_1, \ldots, a_m$ is said to be β_a-dispersed, if the positions of attack centers x_a are dispersed i.e. for all m and for all $\epsilon \geq m^{-\beta_a}$, $\mathbb{E}\left[\max_{x,x' \in \mathcal{X}, x \in \mathcal{B}(x',\epsilon)} \left|\{i \in [m] \mid x = c_{a_i}\}\right|\right] \leq \tilde{O}(\epsilon m)$.*

Theorem 5.1. *Given a sequence of β-dispersed adversarially perturbed losses $\tilde{l}_i = l_i + a_i$, where $\tilde{l}_i, l_i, a_i$ are piecewise L-Lipschitz functions $\mathcal{X} \to [0,1]$ for $i = 1, \ldots, m$ and $\mathcal{X} \subset \mathbb{R}^d$, the exponential forecaster algorithm has $\mathbb{E}[\tilde{R}_m] = \tilde{O}(m\lambda + \frac{\log(1/Z)}{\lambda} + (L+1)m^{1-\beta})$ (with Z as in Theorem 2.1). If in addition we have that a_i is a $m^{-\beta_a}$-bounded, β_a-dispersed attack, then $\mathbb{E}[R_m] = \tilde{O}(m\lambda + \frac{\log(1/Z)}{\lambda} + (L+1)m^{1-\min\{\beta,\beta_a\}})$.*

Together with Theorem 3.3, this implies no regret meta-learning in the presence of dispersed adversaries, in particular the occurrence of unreliable data in small dispersed parts of the domain. We also show a lower bound which establishes that our bounds are essentially optimal in the attack dispersion.

Theorem 5.2. *There exist sequences of piecewise L-Lipschitz functions $\tilde{l}_i, l_i, a_i$ $[0,1] \to [0,1]$ for $i = 1, \ldots, m$ such that for any online algorithm*

1. *$\tilde{l}_i$ is β-dispersed and $\mathbb{E}[\tilde{R}_m] = \Omega(m^{1-\beta})$,*
2. *$\tilde{l}_i$ is β-dispersed, a_i is $m^{-\beta}$-bounded, β_a-dispersed and $\mathbb{E}[R_m] = \Omega(m^{1-\min\{\beta,\beta_a\}})$.*

6 Conclusion

In this paper we studied the initialization-based meta-learning of piecewise-Lipschitz functions, demonstrating how online convex optimization over an adaptive discretization can find an initialization that improves the performance of the exponential forecaster across tasks, assuming the tasks have related optima. We then applied this result in two settings: online configuration of clustering algorithms and adversarial robustness in online learning. For the latter we introduced a dispersion-based understanding of robustness that we believe to be of independent interest. In addition, there are further interesting applications of our work to other algorithm configuration problems.

Acknowledgments

This material is based on work supported by the National Science Foundation under grants CCF-1535967, CCF-1910321, IIS-1618714, IIS-1901403, SES-1919453, IIS-1705121, IIS-1838017, IIS-2046613 and IIS-2112471; the Defense Advanced Research Projects Agency under cooperative agreements HR00112020003 and FA875017C0141; an AWS Machine Learning Research Award; an Amazon Research Award; a Bloomberg Research Grant; a Microsoft Research Faculty Fellowship; an Amazon Web Services Award; a Facebook Faculty Research Award; funding from Booz Allen Hamilton Inc.; a Block Center Grant; and a Two Sigma Fellowship Award. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of any of these funding agencies.

References

[1] Naman Agarwal, Brian Bullins, Elad Hazan, Sham Kakade, and Karan Singh. Online control with adversarial disturbances. In *International Conference on Machine Learning*, pages 111–119. PMLR, 2019.

[2] Pierre Alquier, The Tien Mai, and Massimiliano Pontil. Regret bounds for lifelong learning. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2938–2948. PMLR, 2020.

[4] Maria-Florina Balcan. Book chapter Data-Driven Algorithm Design. In *Beyond Worst Case Analysis of Algorithms, T. Roughgarden (Ed)*. Cambridge University Press, 2020.

[5] Maria-Florina Balcan, Avrim Blum, Dravyansh Sharma, and Hongyang Zhang. On the power of abstention and data-driven decision making for adversarial robustness. *arXiv preprint arXiv:2010.06154*, 2020.

[6] Maria-Florina Balcan, Avrim Blum, and Santosh Vempala. Efficient representations for lifelong learning and autoencoding. In *Proceedings of the 28th Annual Conference on Learning Theory*, 2015.

[7] Maria-Florina Balcan, Travis Dick, and Manuel Lang. Learning to link. In *International Conference on Learning Representations*, 2019.

[8] Maria-Florina Balcan, Travis Dick, and Wesley Pegden. Semi-bandit optimization in the dispersed setting. In *Conference on Uncertainty in Artificial Intelligence*, pages 909–918. PMLR, 2020.

[9] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *International conference on machine learning*, pages 344–353. PMLR, 2018.

[10] Maria-Florina Balcan, Travis Dick, and Dravyansh Sharma. Learning piecewise Lipschitz functions in changing environments. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, pages 3567–3577, 2020.

[11] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Dispersion for data-driven algorithm design, online learning, and private optimization. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 603–614, 2018.

[12] Maria-Florina Balcan, Vaishnavh Nagarajan, Ellen Vitercik, and Colin White. Learning-theoretic foundations of algorithm configuration for combinatorial partitioning problems. In *Annual Conference on Learning Theory*, pages 213–274, 2017.

[13] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. A general theory of sample complexity for multi-item profit maximization. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 173–174, 2018.

[14] Maria-Florina Balcan and Dravyansh Sharma. Data driven algorithms for limited labeled data learning. *arXiv preprint arXiv:2103.10547*, 2021.

[15] Maria-Florina F Balcan, Travis Dick, and Colin White. Data-driven clustering via parameterized lloyd's families. *Advances in Neural Information Processing Systems*, 31:10641–10651, 2018.

[16] Avrim Blum. Technical perspective: Algorithm selection as a learning problem. *Communications of the ACM*, 63(6):86–86, 2020.

[17] Wieland Brendel, Jonas Rauber, Alexey Kurakin, Nicolas Papernot, Behar Veliqi, Sharada P Mohanty, Florian Laurent, Marcel Salathé, Matthias Bethge, Yaodong Yu, et al. Adversarial vision challenge. In *The NeurIPS'18 Competition*, pages 129–153. Springer, 2020.

[18] Fei Chen, Zhenhua Dong, Zhenguo Li, and Xiuqiang He. Federated meta-learning for recommendation. arXiv, 2018.

[19] Vincent Cohen-Addad and Varun Kanade. Online optimization of smoothed piecewise constant functions. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[20] Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Learning-to-learn stochastic gradient descent with biased regularization. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[21] Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Online-within-online meta-learning. In *Advances in Neural Information Processing Systems*, 2019.

[22] Simon S. Du, Wei Hu, Sham M. Kakade, Jason D. Lee, and Qi Lei. Few-shot learning via learning the representation, provably.

[23] Yan Duan, Marcin Andrychowicz, Bradly Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in Neural Information Processing Systems*, 2017.

[24] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics*, 2020.

[25] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[26] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[27] Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017.

[28] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69:169–192, 2007.

[29] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, 2019.

[30] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[31] Mikhail Khodak, Renbo Tu, Tian Li, Liam Li, Maria-Florina Balcan, Virginia Smith, and Ameet Talwalkar. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. arXiv, 2021.

[32] Weihao Kong, Raghav Somani, Sham Kakade, and Sewoong Oh. Robust meta-learning for mixed linear regression with small batches. *Advances in Neural Information Processing Systems*, 33, 2020.

[33] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[34] Jeffrey Li, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar. Differentially private meta-learning. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[35] Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17(1):2853–2884, 2016.

[36] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[37] Francesco Orabona, Nicolo Cesa-Bianchi, and Claudio Gentile. Beyond logarithmic bounds in online learning. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, 2012.

[38] Francesco Orabona and David Pal. Coin betting and parameter-free online learning. In *Advances in Neural Information Processing Systems*, 2016.

[39] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

[40] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[41] Alon Resler and Yishay Mansour. Adversarial online learning with noise. In *International Conference on Machine Learning*, pages 5429–5437. PMLR, 2019.

[42] Nikunj Saunshi, Yi Zhang, Mikhail Khodak, and Sanjeev Arora. A sample complexity separation between non-convex and convex meta-learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[43] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

[44] Daniel A Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM (JACM)*, 51(3):385–463, 2004.

[45] Sebastian Thrun and Lorien Pratt. *Learning to Learn*. Springer Science & Business Media, 1998.

[46] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. Data poisoning attacks against federated learning systems. In *European Symposium on Research in Computer Security*, pages 480–501. Springer, 2020.

[47] Nilesh Tripuraneni, Chi Jin, and Michael I. Jordan. Provable meta-learning of linear representations. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[48] Xiang Wang, Shuai Yuan, Chenwei Wu, and Rong Ge. Guarantees for tuning the step size using a learning-to-learn approach. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[49] Yufan Zhou, Zhenyi Wang, Jiayi Xian, Changyou Chen, and Jinhui Xu. Meta-learning with neural tangent kernels. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Alongside contributions in context, e.g. the end of Section 3.1.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our concern w.r.t. the negative societal impact of this theoretical work is limited to standard risks associated with ML, e.g. for privacy or fair treatment.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A] Experiments run on personal computer (16GB, 2.3 GHz Dual-Core).
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Related work

The success of meta-learning has led to significant theoretical effort to understand it. Most efforts studying initialized-based meta-learning focus on the convex Lipschitz setting [20, 30]; work studying inherently nonconvex modeling approaches instead usually study multi-task representation learning [6, 35, 22, 47] or target optimization, e.g. stationary point convergence [24]. An exception is a study of linear models over Gaussian data showing that nonconvexity is critical to meta-learning an initialization that exploits low-rank task structure [42]. There is also work extending results from the neural tangent kernel literature to meta-learning [49], but in this case the objective becomes convex. On the other hand, we study initializations for learning a class of functions that can be highly non-convex and have numerous discontinuities. Theoretically, our work uses the Average Regret-Upper-Bound Analysis (ARUBA) strategy [29] for obtaining a meta-update procedure for initializing within-task algorithms, which has been applied elsewhere for privacy [34] and federated learning [31]; the main technical advance in our work is in providing the guarantees for it in our setting, which is challenging due to the need to learn over a space of probability measures. Another aspect of our work is to learn the step-size of the within-task algorithm, in addition to its initialization; our approach is similar to that of ARUBA [29] but the step-size alone has also been tuned in several works in a variety of different settings [27, 31, 48].

Data-driven configuration is the selection of an algorithm from a parameterized family, by doing learning over multiple problem instances [27, 12]. In other words, it is 'hyperparameter tuning' with formal guarantees, and has applications to integer programming, clustering and learning with limited labeled data [9, 7, 14]. In this work, we show how this general approach can be made even more effective by enabling it to adapt to task similarity. We also show applications of our results to robust meta-learning in the presence of outliers in the dataset [39, 32]. While previous work on robust online learning has considered adversaries with bounded perturbation in the online learning setting [1, 41], our results allow potentially unbounded perturbations provided the adversary uses a smooth distribution. That is, the adversarial attack can be thought of as a distribution of perturbations, similar to the smoothed analysis approach of [44]. In the offline setting, a similar attack is studied in the context of deep network feature-space attacks by [5]. We also remark that our formulation has a poisoning aspect, since we do not observe the clean loss $l_h(x)$, which is of particular interest in federated learning [3, 46]. Also, note that unlike the typical applications of data-driven design where optimization is over the dual loss function, i.e. loss as a function of the algorithm parameter for a fixed sample $x \in \mathcal{X}$, here we consider learning loss or confidence functions over the input space $\mathcal{X}$.

More background on data-driven algorithm selection, an algorithm design paradigm for setting algorithm parameters when multiple instances of a problem are available or need to be solved, can be found in [16, 4]. By modeling the problem of identifying a good algorithm from data as a statistical learning problem, general learning algorithms have been developed which exploit smoothness of the underlying algorithmic distribution [11]. This provides a new algorithmic perspective, along with tools and insights for good performance under this smoothed analysis for fundamental problems including clustering, mechanism design, and mixed integer programs, and providing guarantees like differential privacy, adaptive online learning and adversarial robustness [7, 13, 10, 5].

B Proofs

B.1 Proof of Theorem 2.1

Proof. The proof adapts the analysis of the exponential forecaster in [11]. Let $W_t = \int_C w_t(\rho)d\rho$ be the normalizing constant and $P_t = \mathbb{E}_{\rho\sim p_t}[u_t(\rho)]$ be the expected payoff at round t. Also let $U_t(\rho) = \sum_{j=1}^{t} u_j(\rho)$. We seek to bound $R_T = OPT - P(T)$, where $OPT = U_T(\rho^*)$ for optimal parameter ρ^* and $P(T) = \sum_{t=1}^{T} P_t$ is the expected utility of Algorithm 1 in T rounds. We will do this by lower bounding $P(T)$ and upper bounding OPT by analyzing the normalizing constant W_t.

Lower bound for $P(T)$: This follows from standard arguments, included for completeness. Using the definitions in Algorithm 1, it follows that

$$\frac{W_{t+1}}{W_t} = \frac{\int_{\mathbb{C}} e^{\lambda u_t(\rho)} w_t(\rho)d\rho}{W_t} = \int_{\mathbb{C}} e^{\lambda u_t(\rho)} \frac{w_t(\rho)}{W_t} d\rho = \int_{\mathbb{C}} e^{\lambda u_t(\rho)} p_t(\rho)d\rho.$$

Use inequalities $e^{\lambda x} \leq 1 + (e^\lambda - 1)x$ for $x \in [0,1]$ and $1 + x \leq e^x$ to conclude

$$\frac{W_{t+1}}{W_t} \leq \int_{\mathbb{C}} p_t(\rho) \left(1 + (e^\lambda - 1)u_t(\rho)\right) d\rho = 1 + (e^{H\lambda} - 1)P_t \leq \exp\left((e^\lambda - 1)P_t\right).$$

Finally, we can write W_{T+1}/W_1 as a telescoping product to obtain

$$\frac{W_{T+1}}{W_1} = \prod_{t=1}^{T} \frac{W_{t+1}}{W_t} \leq \exp\left((e^\lambda - 1)\sum_t P_t\right) = \exp\left(P(T)(e^\lambda - 1)\right),$$

or, $W_{T+1} \leq \exp\left(P(T)(e^\lambda - 1)\right) \int_C w_1(\rho)d\rho$.

Upper bound for OPT: Let $\mathcal{B}^*(r)$ be the ball of radius r around ρ^*. If there are at most k discontinuities in any ball of radius r, we can conclude that for all $\rho \in \mathcal{B}^*(r)$, $U_T(\rho) \geq OPT - k - LTr$. Now, since $W_{T+1} = \int_C w_1(\rho) \exp(\lambda U_T(\rho))d\rho$, we have

$$\begin{aligned} W_{T+1} &\geq \int_{\mathcal{B}^*(r)} w_1(\rho) e^{\lambda U_T(\rho)} d\rho \\ &\geq \int_{\mathcal{B}^*(r)} w_1(\rho) e^{\lambda(OPT - k - LTr)} d\rho \\ &= e^{\lambda(OPT-k-LTr)} \int_{\mathcal{B}^*(r)} w_1(\rho) d\rho. \end{aligned}$$

Putting together with the lower bound, and rearranging, gives

$$\begin{aligned} OPT - P_T &\leq \frac{P(T)(e^\lambda - 1 - \lambda)}{\lambda} + \frac{\log(1/Z)}{\lambda} + k + LTr \\ &\leq T\lambda + \frac{\log(1/Z)}{\lambda} + k + LTr, \end{aligned}$$

where we use that $P(T) \leq T$ and for all $x \in [0,1], e^x \leq 1 + x + (e-2)x^2$. Take expectation over the sequence of utility functions and apply dispersion to conclude the result. □

B.2 Proof of Theorem 2.2

We extend the construction in [10] to the multi-task setting. The main difference is that we generalize the construction for any task similarity, and show that we get the same lower bound asymptotically.

Proof. Define $u^{(b,x)}(\rho) = I[b = 0] * I[\rho > x] + I[b = 1] * I[\rho \leq x]$, where $b \in \{0,1\}$, $x, \rho \in [0,1]$ and $I[\cdot]$ is the indicator function. For each iteration the adversary picks $u^{(0,x)}$ or $u^{(1,x)}$ with equal probability for some $x \in [a, a + D^*]$, the ball of diameter D^* containing all the optima.

For each task t, $m - \frac{3}{D^*}m^{1-\beta}$ functions are presented with the discontinuity $x \in [a + D^*/3, a + 2D^*/3]$ while ensuring β-dispersion. The remaining $\frac{3}{D^*}m^{1-\beta}$ are presented with discontinuities located in successively halved intervals (the 'halving adversary') containing the optima in hindsight, any algorithm gets half of these wrong in expectation. It is readily verified that the functions are β-dispersed. The construction works provided m is sufficiently large ($m > \left(\frac{3}{D^*}\right)^{1/\beta}$). The task averaged regret is therefore also $\tilde{\Omega}(m^{1-\beta})$. □

B.3 Proof of Theorem 3.1

Proof.

$$\begin{aligned}
&\sum_{t=1}^{T}\sum_{m=1}^{m}\ell_{t,i}(\rho_{t,i}) - \min_{\rho_t^*\in C}\sum_{i=1}^{m}\ell_{t,i}(\rho_t^*) \\
&\qquad\leq \sum_{t=1}^{T} U_t(w_t, v_t) \\
&\qquad\leq \min_{v>0} H_T(v)\sqrt{m} + \sum_{t=1}^{T}\left(v + \frac{f_t(w_t)}{v}\right)\sqrt{m} + g(m) \\
&\qquad\leq \min_{w:C\mapsto\mathbb{R}_{\geq 0}, v>0} H_T(v)\sqrt{m} + \frac{F_T(w)\sqrt{m}}{v} + \sum_{t=1}^{T}\left(v + \frac{f_t(w)}{v}\right)\sqrt{m} + g(m) \\
&\qquad\leq \left(H_T(V) + \min\left\{\frac{F_T(w^*)}{V}, 2\sqrt{F_T(w^*)T}\right\} + 2TV\right)\sqrt{m} + Tg(m)
\end{aligned}$$

where the last step is achieved by substituting $w = w^*$ and $v = \max\left\{V, \sqrt{F_T(w^*)/T}\right\}$. □

B.4 Proof of Lemma 3.1

Proof. Define a probability measure $p : C \mapsto \mathbb{R}_{\geq 0}$ that is constant on all elements $\tilde{D} \in \mathcal{D}_t$ of the discretization at time t, taking the value $p(\rho) = \frac{1}{\mathrm{vol}(\tilde{D})}\sum_{D\in D_T, D\subset\tilde{D}} \mathbf{w}_{[D]}\ \forall\ \rho \in \tilde{D}$. Note that for any $D \in \mathcal{D}_T$ that is a subset of $\tilde{D}$ we have that

$$\mathbf{p}_{[D]} = \int_D \tilde{w}(\rho)d\rho = \frac{\mathbf{v}_{[D]}}{\sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}}\mathbf{v}_{[D']}} \sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}} \mathbf{w}_{[D']}$$

Then

$$\begin{aligned}
&D_{KL}(\mathbf{p}||\hat{\mathbf{v}}) - \eta\sum_{s\leq t}\log\langle\mathbf{w}_s^*, \mathbf{p}\rangle \\
&\quad= \sum_{\tilde{D}\in\mathcal{D}_t}\sum_{D\in\mathcal{D}_T, D\subset\tilde{D}} \mathbf{p}_{[D]}\log\frac{\mathbf{p}_{[D]}}{\hat{\mathbf{v}}_{[D]}} - \eta\sum_{s\leq t}\log\sum_{\tilde{D}\in\mathcal{D}_t}\sum_{D\in\mathcal{D}_T, D\subset\tilde{D}} \mathbf{w}_{s[D]}^*\mathbf{p}_{[D]} \\
&\quad= \sum_{\tilde{D}\in\mathcal{D}_t}\sum_{D\in\mathcal{D}_T, D\subset\tilde{D}} \frac{\mathbf{v}_{[D]}}{\sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}}\mathbf{v}_{[D']}} \sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}} \mathbf{w}_{[D']}\log\frac{\sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}}\mathbf{w}_{[D']}}{\sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}}\hat{\mathbf{v}}_{[D']}} \\
&\qquad - \eta\sum_{s\leq t}\log\sum_{\tilde{D}\in\mathcal{D}_t}\sum_{D\in\mathcal{D}_T, D\subset\tilde{D}} \frac{\mathbf{w}_{s[D]}^*\mathbf{v}_{[D]}}{\sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}}\mathbf{v}_{[D']}} \sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}} \mathbf{w}_{[D']} \\
&\quad\leq \sum_{\tilde{D}\in\mathcal{D}_t}\sum_{D\in\mathcal{D}_T, D\subset\tilde{D}} \frac{\mathbf{v}_{[D]}}{\sum_{D'\in\mathcal{D}_T, D'\in\tilde{D}}\mathbf{v}_{[D']}} \sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}} \mathbf{w}_{D'}\log\frac{\mathbf{w}_{[D']}}{\hat{\mathbf{v}}_{[D']}} \\
&\qquad - \eta\sum_{s\leq t}\log\sum_{\tilde{D}\in\mathcal{D}_t, \tilde{D}\subset C_s}\sum_{D\in\mathcal{D}_T, D\subset\tilde{D}} \frac{\mathbf{v}_{[D]}}{\sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}}\mathbf{v}_{[D']}} \sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}} \mathbf{w}_{[D']} \\
&\quad= \sum_{\tilde{D}\in\mathcal{D}_t}\sum_{D'\in\mathcal{D}_T, D'\in\tilde{D}} \mathbf{w}_{[D']}\log\frac{\mathbf{w}_{[D']}}{\hat{\mathbf{v}}_{[D']}} - \eta\sum_{s\leq t}\log\sum_{\tilde{D}\in\mathcal{D}_t, \tilde{D}\subset C_s}\sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}} \mathbf{w}_{[D']} \\
&\quad= D_{KL}(\mathbf{w}||\hat{\mathbf{v}}) - \eta\sum_{s\leq t}\log\langle\mathbf{w}_s^*, \mathbf{w}\rangle
\end{aligned}$$

where the inequality follows from applying the log-sum inequality to the first term and the fact that $\mathbf{w}_{s[D]}^* = \mathbf{1}_{D\subset C_s}$ in the second term. Note that we also have

$$\|\mathbf{p}\|_1 = \sum_{\tilde{D}\in\mathcal{D}_t}\sum_{D\in\mathcal{D}_T, D\subset\tilde{D}} \frac{\mathbf{v}_{[D]}}{\sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}}\mathbf{v}_{[D']}} \sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}} \mathbf{w}_{[D']} = \sum_{\tilde{D}\in\mathcal{D}_t}\sum_{D'\in\mathcal{D}_T, D'\subset\tilde{D}} \mathbf{w}_{[D']} = 1$$

and

$$\mathbf{p}_{[D]} = \frac{\mathbf{v}_{[D]}}{\sum_{D' \in \mathcal{D}_T, D' \subset \tilde{D}} \mathbf{v}_{[D']}} \sum_{D' \in \mathcal{D}_T, D' \subset \tilde{D}} \mathbf{w}_{[D']} \geq \frac{\gamma \mathbf{v}_{[D]}}{\sum_{D' \in \mathcal{D}_T, D' \subset \tilde{D}} \mathbf{v}_{[D']}} \sum_{D' \in \mathcal{D}_T, D' \subset \tilde{D}} \hat{\mathbf{v}}_{[D']} = \gamma \hat{\mathbf{v}}_{[D]}$$

so $\mathbf{p}$ satisfies the optimization constraints. Therefore, since $\mathbf{w}$ was defined to be the minimum of the sum of the KL-divergence (a strongly-convex function [43, Example 2.5]) and a convex function, it is unique and so coincides with $\mathbf{p}$.

On the other hand

$$\begin{aligned} D_{KL}(\mathbf{p}(t)||\hat{\mathbf{v}}(\mathbf{t})) - \eta \sum_{s \leq t} \log\langle \mathbf{w}_s^*(t), \mathbf{p}(t)\rangle &\leq D_{KL}(\mathbf{p}||\hat{\mathbf{v}}) - \eta \sum_{s \leq t} \log\langle \mathbf{w}_s^*, \mathbf{p}\rangle \\ &= D_{KL}(\mathbf{w}||\hat{\mathbf{v}}) - \eta \sum_{s \leq t} \log\langle \mathbf{w}_s^*, \mathbf{w}\rangle \\ &\leq D_{KL}(\tilde{\mathbf{w}}||\hat{\mathbf{v}}) - \eta \sum_{s \leq t} \log\langle \mathbf{w}_s^*, \tilde{\mathbf{w}}\rangle \\ &= D_{KL}(\tilde{\mathbf{w}}(t)||\hat{\mathbf{v}}(t)) - \eta \sum_{s \leq t} \log\langle \mathbf{w}_s^*(t), \tilde{\mathbf{w}}(t)\rangle \end{aligned}$$

where the first inequality follows from above and the second from the optimality of $\mathbf{w}$. Note that by nonnegativity the discretization of p does not affect its measure over C, so $\|\mathbf{p}\|_1 = 1 \implies \|\mathbf{p}(t)\|_1 = 1$. Finally, also from above we have

$$\mathbf{p}(t)_{[D]} = \sum_{D' \in \mathcal{D}_T, D' \subset D} \mathbf{p}_{[D']} \geq \gamma \sum_{D' \in \mathcal{D}_T, D' \subset D} \mathbf{p}_{[D']} \hat{\mathbf{v}}_{[D']} = \gamma \hat{\mathbf{v}}(t)_{[D]}$$

Thus as before $\mathbf{p}(t)$ satisfies the optimization constraints, which with the previous inequality and the uniqueness of the optimum $\tilde{\mathbf{w}}(t)$ implies that $\mathbf{p}(t) = \tilde{\mathbf{w}}(t)$. Finally, since $\tilde{w}$ is constant on all elements of the discretization $\mathcal{D}_t$ of C this last fact implies that $\mathbf{p} = \tilde{\mathbf{w}}$, which together with $\mathbf{p} = \mathbf{w}$ implies the result. □

B.5 Lipschitzness for Algorithm 2

Claim B.1. *The loss f_t is $\frac{1}{\gamma \operatorname{vol}(C_t)}$-Lipschitz w.r.t. $\|\cdot\|_1$ over the set $\{\mathbf{w} \in \mathbb{R}^{|\mathcal{D}_T|} : \|\mathbf{w}\|_1 = 1, \mathbf{w} \geq \gamma \hat{\mathbf{v}}\}$.*

Proof.

$$\max_{\|\mathbf{w}\|_1 = 1, \mathbf{w} \geq \gamma \hat{\mathbf{v}}} \|\nabla \log\langle \mathbf{w}_t^*, \mathbf{w}\rangle\|_\infty = \max_{D, \|\mathbf{w}\|_1 = 1, \mathbf{w} \geq \gamma \hat{\mathbf{v}}} \frac{\mathbf{w}_{t\,[D]}^*}{\langle \mathbf{w}_t^*, \mathbf{w}\rangle} \leq \frac{1}{\langle \mathbf{w}_t^*, \gamma \hat{\mathbf{v}}\rangle} = \frac{1}{\gamma \operatorname{vol}(C_t)}$$

□

B.6 Proof of Corollary 3.1

Proof. Using first-order conditions we have that the optimum in hindsight of the functions h_t satisfies

$$v^2 = \frac{1}{T} \sum_{t=1}^{T} f_t(w_t) = -\frac{1}{T} \sum_{t=1}^{T} \log\langle \mathbf{w}_t^*, \mathbf{w}_t\rangle \leq \frac{1}{T} \sum_{t=1}^{T} \log \frac{1}{\gamma \operatorname{vol}(C_t)}$$

Applying [29, Corollary C.2] with $\alpha_t = 1$, $B_t^2 = f_t(w_t)$, and $D^2 - \log \gamma$ instead of D^2 yields the result. □

B.7 Proof of Corollary 3.2

Proof. Using first-order conditions we have that the optimum in hindsight of the functions h_t satisfies

$$v^2 = \frac{1}{T} \sum_{t=1}^{T} f_t(w_t) = -\frac{1}{T} \sum_{t=1}^{T} \log\langle \mathbf{w}_t^*, \mathbf{w}_t\rangle \leq \frac{1}{T} \sum_{t=1}^{T} \log \frac{1}{\gamma \operatorname{vol}(C_t)}$$

Applying [29, Proposition B.2] with $\alpha_t = 1$, $B_t^2 = f_t(w_t)$, and $D^2 - \log \gamma$ instead of D^2 yields the result. □

B.8 Proof of Theorem 3.3

Proof. We have $F_T(w^*) = \tilde{O}(\sqrt{BG}T^{\frac{3}{4}})$ and $H_T(V) = \tilde{O}(\min\{1/V, \sqrt[5]{T}\}T^{\frac{3}{5}})$ from Corollaries 3.1 and 3.2. Substituting into Lemma 3.1 and simplifying yields

$$\tilde{O}\left(\frac{\min\left\{\frac{1}{V}, \sqrt[4]{T}\right\}}{\sqrt{T}} + \min\left\{\frac{\sqrt{BG}}{V\sqrt[4]{T}}, \frac{\sqrt[4]{BG}}{\sqrt[8]{T}}\right\} + 2V\right)\sqrt{m} + g(m)$$

Simplifying further yields the result. □

B.9 Proof of Theorem 5.1

Proof. The bound on $\mathbb{E}[\tilde{R}_T]$ is immediate from Theorem 2.1. For $\mathbb{E}[R_T]$, we can upper bound the natural regret with the sum of robust regret, total adversarial perturbation at the optimum and a term corresponding to the difference between the loss of natural and robust optima.

$$\begin{aligned}
R_T &= \sum_{t=1}^{T} l_t(x_t) - \min_{x\in\mathcal{X}} \sum_{t=1}^{T} l_t(x) \\
&= \tilde{R}_T + \sum_{t=1}^{T} l_t(x_t) - \sum_{t=1}^{T} \tilde{l}_t(x_t) + \min_{x\in\mathcal{X}} \sum_{t=1}^{T} \tilde{l}_t(x) - \min_{x\in\mathcal{X}} \sum_{t=1}^{T} l_t(x) \\
&= \tilde{R}_T - \sum_{t=1}^{T} a_t(x_t) + \sum_{t=1}^{T} a_t(\tilde{x}^*) + \sum_{t=1}^{T} l_t(\tilde{x}^*) - \sum_{t=1}^{T} l_t(x^*) \\
&\leq \tilde{R}_T + \sum_{t=1}^{T} a_t(\tilde{x}^*) + \left|\sum_{t=1}^{T} l_t(\tilde{x}^*) - \sum_{t=1}^{T} l_t(x^*)\right|
\end{aligned}$$

where $\tilde{x}^* = \arg\min_{x\in\mathcal{X}} \sum_{t=1}^{T} \tilde{l}_t(x)$ and $x^* = \arg\min_{x\in\mathcal{X}} \sum_{t=1}^{T} l_t(x)$. We now use the β_a-dispersedness of the attack to show an excess expected regret of $\tilde{O}(T^{1-\beta_a})$. Using attack dispersion on a ball of radius $T^{-\beta_a}$ around $\tilde{x}^*$, the number of attacks that have non-zero $a_t(\tilde{x}^*)$ is at most $\tilde{O}(T^{1-\beta_a})$, and therefore $\sum_{t=1}^{T} a_t(\tilde{x}^*) \leq \tilde{O}(T^{1-\beta_a})$. Further, observe that the robust and natural optima coincide unless some attack occurs at the natural optimum x^*. We can use attack dispersion at x^*, and a union bound across rounds, to conclude $\mathbb{E}|\sum_{t=1}^{T} l_t(\tilde{x}^*) - \sum_{t=1}^{T} l_t(x^*)| \leq \tilde{O}(T^{1-\beta_a})$ which concludes the proof. □

B.10 Proof of Theorem 5.2

Proof. Part 1 follows from the lower bound in Theorem 2.2, by setting $\tilde{l}_i = l_i$ as the loss sequence used in the proof.

To establish Part 2, we extend the construction as follows. $\tilde{l}_i = l_i$ are both equal and correspond to the 'halving adversary' from the proof of Theorem 2.2 for the first $\Theta(m^{1-\beta})$ rounds. If $\beta \leq \beta_a$ we are done, so assume otherwise. Let I denote the interval containing the optima over the rounds so far. Notice that the length of I is at most $|I| \leq (\frac{1}{2})^{\Theta(m^{1-\beta})} \leq (\frac{1}{2})^{\beta \log m} = m^{-\beta}$ for $\beta > 0$. For further rounds l_i continues to be the halving adversary for $\Theta(m^{1-\beta_a})$ rounds, which implies any algorithm suffers $\Omega(m^{1-\beta_a})$ regret. We set attack a_i on interval I such that $\tilde{l}_i = 0$ on I on these rounds. This ensures that a_i is β_a-dispersed and $\tilde{l}_i$ is β-dispersed. Putting together with the case $\beta \leq \beta_a$, we obtain $\Omega(m^{1-\min\{\beta,\beta_a\}})$ bound on the regret of any algorithm. □

C Learning algorithmic parameters for combinatorial problems

We discuss implications of our results for several combinatorial problems of widespread interest including integer quadratic programming and auction mechanism design. We will need the following theorem from [14], which generalizes the recipe for establishing dispersion given by [8] for $d = 1, 2$ dimensions to arbitrary constant d dimendions. It is straightforward to apply the recipe to establish

dispersion for these problems, which in turn implies that our meta-learning results are applicable. We demonstrate this for a few important problems below for completeness.

Theorem C.1 ([14]). *Let $l_1, \ldots, l_m : \mathbb{R}^d \to \mathbb{R}$ be independent piecewise L-Lipschitz functions, each having discontinuities specified by a collection of at most K algebraic hypersurfaces of bounded degree. Let $\mathcal{L}$ denote the set of axis-aligned paths between pairs of points in $\mathbb{R}^d$, and for each $s \in \mathcal{L}$ define $D(m, s) = |\{1 \le t \le m \mid l_t \text{ has a discontinuity along } s\}|$. Then we have $\mathbb{E}[\sup_{s\in\mathcal{L}} D(m, s)] \le \sup_{s\in\mathcal{L}} \mathbb{E}[D(m, s)] + O(\sqrt{m \log(mK)})$.*

C.1 Greedy knapsack

We are given a knapsack with capacity $\texttt{cap}$ and items $i \in [m]$ with sizes w_i and values v_i. The goal is to select a subset S of items to add to the knapsack such that $\sum_{i\in S} w_i \le \texttt{cap}$ while maximizing the total value $\sum_{i\in S} v_i$ of selected items. We consider a general greedy heuristic to insert items with largest v_i/w_i^ρ first (due to [27]) for $\rho \in [0, 10]$.

The classic greedy heuristic sets $\rho = 1$ and can be used to provide a 2-approximation for the problem. However other values of ρ can improve the knapsack objective on certain problem instances. For example, for the value-weight pairs $\{(0.99, 1), (0.99, 1), (1.01, 1.01)\}$ and capacity $\texttt{cap} = 2$ the classic heuristic $\rho = 1$ gives value 1.01 as the greedy heuristic is maximized for the third item. However, using $\rho = 3$ (or any $\rho > 1 + \log(1/0.99)/\log(1.01) > 2.01$) allows us to pack the two smaller items giving the optimal value 1.98.

Our result (Theorem 3.3) when applied to this problem shows that it is possible to learn the optimal parameter values for the greedy heuristic algorithm family for knapsack from similar tasks.

Theorem C.2. *Consider instances of the knapsack problem given by bounded weights $w_{i,j} \in [1, C]$ and κ-bounded independent values $v_{i,j} \in [0, 1]$ for $i \in [m], j \in [T]$. Then the asymptotic task-averaged regret for learning the algorithm parameter ρ for the greedy heuristic family described above is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

Proof. Lemma 11 of [8] shows that the loss functions form a $\frac{1}{2}$-dispersed sequence. The result follows by applying Theorem 3.3 with $\beta = \frac{1}{2}$. □

C.2 k-center clustering

We consider the α-Lloyd's clustering algorithm family from [15], where the initial k centers in the procedure are set by sampling points with probability proportional to d^α where d is the distance from the centers selected so far for some $\alpha \in [0, D], D \in \mathbb{R}_{\ge 0}$. For example, $\alpha = 0$ corresponds to the vanilla k-means with random initial centers, and $\alpha = 2$ setting is the k-means++ procedure. For this algorithm family, we are able to show the following guarantee. Interestingly, for this family it is sufficient to rely on the internal randomness of the algorithmic procedure and we do not need assumptions on data smoothness.

Theorem C.3. *Consider instances of the k-center clustering problem on n points, with Hamming loss $l_{i,j}$ for $i \in [m], j \in [T]$ against some (unknown) ground truth clustering. Then the asymptotic task-averaged regret for learning the algorithm parameter α for the α-Lloyd's clustering algorithm family of [15] is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

Proof. We start by applying Theorem 4 from [15] to an arbitrary α-interval $[\alpha_0, \alpha_0 + \epsilon] \subseteq [0, D]$ of length ϵ. The expected number of discontinuities (expectation under the internal randomness of the algorithm when sampling successive centers), is at most

$$D(m, \epsilon) = O(nk \log(n) \log(\max\{(\alpha_0 + \epsilon)/\alpha_0), (\alpha_0 + \epsilon) \log R\}),$$

where R is an upper bound on the ratio between any pair of non-zero distances. Considering cases $\alpha_0 \lesseqgtr \frac{1}{\log R}$ and using the inequality $\log(1 + x) \le x$ for $x \ge 0$ we get that there are, in expectation, at most $O(\epsilon nk \log n \log R)$ discontinuities in any interval of length ϵ. Theorem C.1 now implies $\frac{1}{2}$-dispersion using the recipe from [8]. The task-averaged regret bound follows from Theorem 3.3. □

C.3 Maximum weighted independent set (MWIS)

We leverage known results about the MWIS problem to show $\frac{1}{2}$-dispersion, which together with Theorem 3.3 implies that our bound on the task-averaged regret improves with task similarity V.

In MWIS, there is a graph $G = (V, E)$ and a weight $w_v \in \mathbb{R}^+$ for each vertex $v \in V$. The goal is to find a set of non-adjacent vertices with maximum total weight. The problem is NP-hard and in fact does not have any constant factor polynomial time approximation algorithm. [27] propose a greedy heuristic family, which selects vertices greedily based on largest value of $w_v/(1 + \deg(v))^\rho$, where $\deg(v)$ is the degree of vertex v, and removes neighbors of the selected vertex before selecting the next vertex.

For this algorithm family, we can learn the best parameter ρ provided pairs of vertex weights have a joint κ-bounded distribution, and Theorem 3.3 implies regret bounds that improve with task similarity. We use the recipe from [8] to establish dispersion.

Theorem C.4. *Consider instances of MWIS with all vertex weights in $(0, 1]$ and for each instance, every pair of vertex weights has a κ-bounded joint distribution. Then the asymptotic task-averaged regret for learning the algorithm parameter ρ is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

Proof. The loss function is piecewise constant with discontinuities corresponding to ρ such that $w_v/(1 + \deg(v))^\rho = w_u/(1 + \deg(u))^\rho$ for a pair of vertices u, v. [11] show that the discontinuities have $(\kappa \ln n)$-bounded distributions where n is the number of vertices. This implies that in any interval of length ϵ, we have in expectation at most $\epsilon\kappa \ln n$ discontinuities. Using this in dispersion recipe from [8] implies $\frac{1}{2}$-dispersion, which in turn implies the desired regret bound by applying Theorem 3.3. □

C.4 Integer quadratic programming (IQP)

The objective is to maximize a quadratic function $z^T Az$ for A with non-negative diagonal entries, subject to $z \in \{0, 1\}^n$. In the classic Goemans-Williamson algorithm [26] one solves an SDP relaxation $U^T AU$ where columns u_i of U are unit vectors. u_i are then rounded to $\{\pm 1\}$ by projecting on a vector Z drawn according to the standard Gaussian, and using $\texttt{sgn}(\langle u_i, Z\rangle)$. A simple parametric family is s-linear rounding where the rounding is as before if $|\langle u_i, Z\rangle| > s$ but uses probabilistic rounding to round u_i to 1 with probability $\frac{1+(\langle u_i, Z\rangle)/s}{2}$. The dispersion analysis of the problem from [11] and the general recipe from [8] imply that our results yield low task-averaged regret for learning the parameter of the s-linear rounding algorithms.

Theorem C.5. *Consider instances of IQP given by matrices $A_{i,j}$ and rounding vectors $Z_{i,j} \sim \mathcal{N}_n$ for $i \in [m], j \in [T]$. Then the asymptotic task-averaged regret for learning the algorithm parameter s for s-linear rounding is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

Proof. As noted in [11], since $Z_{i,j}$ are normal, the local of discontinuities $s = |\langle u_i, Z\rangle|$ are distributed with a $\sqrt{\frac{2}{\pi}}$-bounded density. Thus in any interval of length ϵ, we have in expectation at most $\epsilon\sqrt{\frac{2}{\pi}}$ discontinuities. Theorem C.1 together with the general recipe from [8] implies $\frac{1}{2}$-dispersion. The task-averaged regret bound is now a simple application of Theorem 3.3. □

Our results are an improvement over prior work which have only considered iid and (single-task) online learning settings. Similar improvements can be obtained for auction design, as described below. We illustrate this using a relatively simple auction, but the same idea applies for an extensive classes of auctions as studied in [13].

C.5 Posted price mechanisms with additive valuations

There are m items and n bidders with valuations $v_j(b_i), j \in [n], i \in [2^m]$ for all 2^m bundles of items. We consider additive valuations which satisfy $v_j(b) = \sum_{i \in b} v_j(\{i\})$. The objective is to maximize the social welfare (sum of buyer valuations). If the item values for each buyer have κ-bounded distributions, then the corresponding social welfare is dispersed and our results apply.

Theorem C.6. *Consider instances of posted price mechanism design problems with additive buyers and κ-bounded marginals of item valuations. Then the asymptotic task-averaged regret for learning the price which maximizes the social welfare is $o_T(1) + 2V\sqrt{m} + O(\sqrt{m})$.*

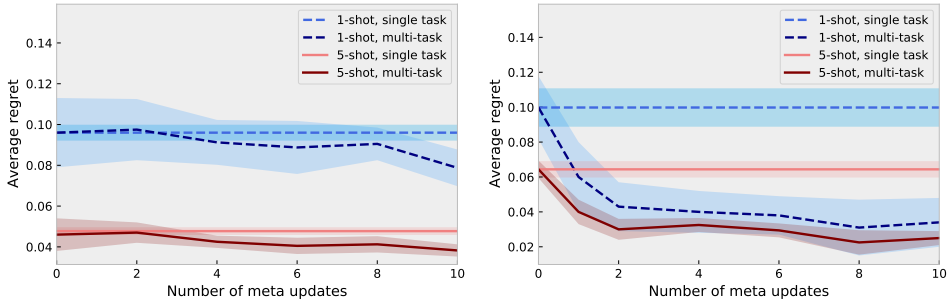
Proof. As noted in [11], the locations of discontinuities are along axis-parallel hyperplanes (buyer j will be willing to buy item i at a price p_i if and only if $v_j(\{i\}) \geq p_i$, each buyer-item pair in each instance corresponds to a hyperplane). Thus in any pair of points p, p' (corresponding to pricing) at distance ϵ, we have in expectation at most $\epsilon\kappa mn$ discontinuities along any axis-aligned path joining p, p', since discontinuities for an item can only occur along axis-aligned segment for the axis corresponding to the item. Theorem C.1 now implies $\frac{1}{2}$-dispersion. The task-averaged regret bound is now a simple application of Theorem 3.3. □

D Additional experiments

We include additional experiments to study how the meta-learning improves with number of tasks, and how the variance of the performance of our approach across different datasets can be understood by examining task similarity and data dispersion.

D.1 Number of training tasks needed for meta-learning

We also examine the number of training tasks that our meta-learning procedure needs to obtain improvements over the single-task baseline. We use a single test task, and a variable number of training tasks (0 through 10) to meta-learn the initialization. We use the same settings as in Section 4.2, except the meta-learning experiments have been averaged over 20 iterations (to average over randomization in the algorithms). In Figure 1, we plot the average regret against number of meta-updates performed before starting the test task, and compare against the single-task baselines. We observe gains with meta-learning with just $T = 10$ tasks for the Omniglot dataset, and with even a single task in the Gaussian mixture dataset. The latter is likely due to a very high degree of task similarity across all the tasks (examined below), so learning on any task transfers very well to another task.



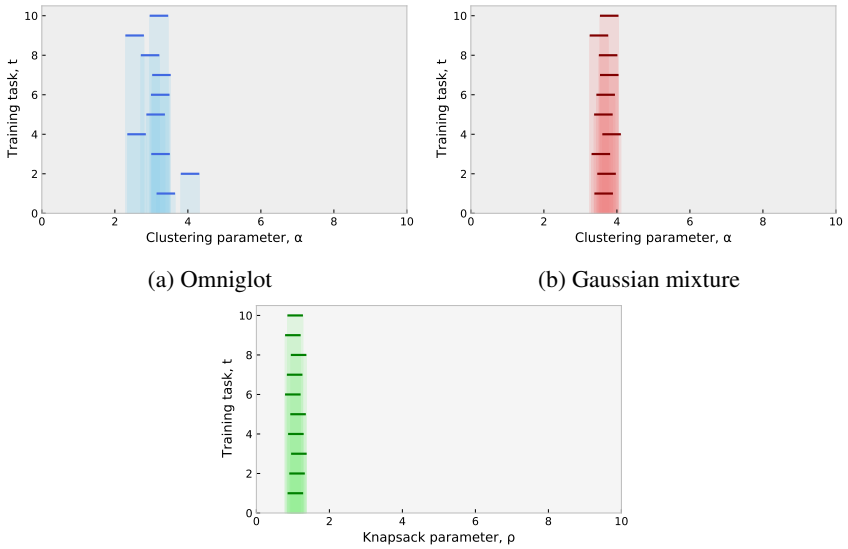
(a) Omniglot (b) Gaussian mixture

Figure 1: Average regret vs. number of training tasks for meta-learning.

D.2 Task similarity and dispersion

We also examine the task similarity of the different tasks by plotting the optimal values α_t^* of the clustering parameter α and the corresponding balls $\mathcal{B}(\alpha_t^*, m^{-\beta})$ used in our definition of task similarity (Figure 2).

The intervals of the parameter induced by these balls correspond to the discretization used by Algorithm 2. We notice a stronger correlation in task similarity for the Gaussian mixture clustering tasks, which implies that meta-learning is more effective here (both in terms of learning test tasks faster, and with lower regret). For knapsack the task similarity is also high, but it turns out that for our dataset there are very 'sharp peaks' at the optima of the total knapsack values as a function of the parameter ρ. So even though meta-learning helps us get within a small ball of the optima, a few steps

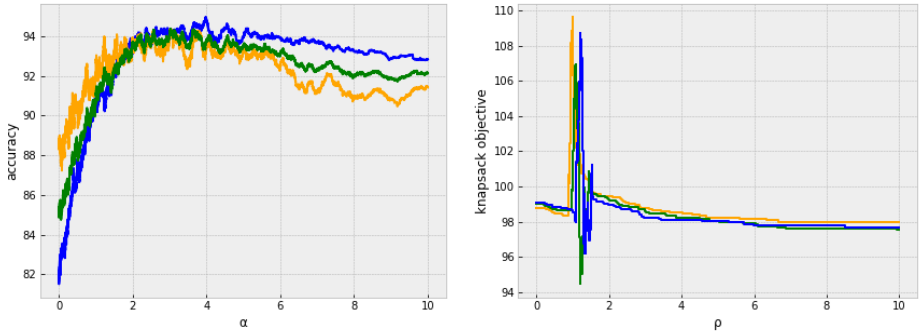


(a) Omniglot (b) Gaussian mixture

(c) Knapsack

Figure 2: Location of optimal parameter values for the training tasks.

are still needed to converge and we do not see the single-shot benefits of meta-learning as we do for the Gaussian clustering experiment.



(a) Clustering (Gaussian mixture dataset) (b) Greedy Knapsack

Figure 3: Average performance (over algorithm randomization) for a few tasks as a function of the configuration parameter. This explains why, despite high task similarity in either case, few-shot meta-learning works better for the Gaussian mixture clustering.