Online Parameter-Free Learning of Multiple Low Variance Tasks

Giulia Denevi[1] giulia.denevi@iit.it
Dimitris Stamos[2] d.stamos.12@ucl.ac.uk
Massimiliano Pontil [1,2] massimiliano.pontil@iit.it

[1] Computational Statistics and Machine Learning, Istituto Italiano di Tecnologia, 16163 Genova, Italy
[2] Computer Science Department, University College of London, WC1E 6BT, London, United Kingdom

Abstract

We propose a method to learn a common bias vector for a growing sequence of low-variance tasks. Unlike state-of-the-art approaches, our method does not require tuning any hyper-parameter. Our approach is presented in the non-statistical setting and can be of two variants. The "aggressive" one updates the bias after each datapoint, the "lazy" one updates the bias only at the end of each task. We derive an across-tasks regret bound for the method. When compared to state-of-the-art approaches, the aggressive variant returns faster rates, the lazy one recovers standard rates, but with no need of tuning hyper-parameters. We then adapt the methods to the statistical setting: the aggressive variant becomes a multi-task learning method, the lazy one a meta-learning method. Experiments confirm the effectiveness of our methods in practice.

1 INTRODUCTION

A long standing problem in machine learning is to develop algorithms that can learn effectively on the basis of only few training examples. To this end, a basic principle that has proven fruitful is to leverage similarities among a set of tasks in order to facilitate their learning process by their corresponding training samples. This basic principle has been studied both from a multi-task learning (MTL) and a meta-learning or learning-to-learn (LTL) perspective. In the first case we wish to perform well on the same tasks used during training, in the second case we aim to extract "knowledge" from the observed tasks that would be useful for solving *new* (possible yet unseen) similar tasks. We refer to [5, 6, 22, 25, 33] and references therein for a detailed discussion on these frameworks.

Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI), PMLR volume 124, 2020.

Both multi-task learning and meta-learning were originally investigated in the setting in which the tasks' data are assumed to be independently sampled from an underlying probability distribution and they are processed in one entire batch, see for instance [5, 16, 24, 25, 29]. Quite recently, significant progress has been made towards the design of more efficient algorithms in which the data are sequentially processed and may even be adversarially generated, see [1, 4, 7, 11, 12, 13, 17, 30]. In this work, we focus on the so-called Online-Within-Online (OWO) setting, in which both the tasks and their samples are observed sequentially.

Nevertheless, the existing multi-task learning or meta-learning methods in the literature require tuning hyper-parameters and their proper choice is necessary in order to demonstrate the advantage of such methods over the baseline algorithm learning the tasks independently. Typically, in practice, this bottleneck is addressed either by an expensive validation procedure in the statistical setting, or by the so-called doubling trick procedures in the adversarial setting. In this work, we wish to design OWO parameter-free methods that are well suited to address an increasing sequence of low-variance tasks. To this end we consider a within-task variant of the online parameter-free algorithm by [10], in which the iterates are translated by a common bias vector. The main goal of this work is to design and analyze a parameter-free procedure to learn a good bias directly from a sequence of observed tasks.

Contributions. We first show that, similarly to what already observed for other families of algorithms requiring tuning of hyper-parameters [4, 11, 14], also for the parameter-free family considered here, setting the "right' bias can be advantageous with respect to (w.r.t.) learning the tasks independently by the unbiased algorithm, when the variance of the target tasks' vectors is sufficiently small. After this, the main contribution of this work is to develop a parameter-free method aiming at inferring a good bias from a sequence of tasks in the OWO

framework. The method is originally presented in a non-statistical setting and it is able to incrementally process a growing sequence of tasks. Our method can be of two variants: an "aggressive" one in which the bias vector is updated after each point, and a "lazy" one, in which the bias' update is performed only at the end of each task training sequence. We then derive an across-tasks regret bound for the proposed method. In the aggressive case the bound enjoys faster rates w.r.t. the state-of-the-art approaches for growing tasks' sequences, while, in the lazy case we recover standard rates, but with no need of hyper-parameters' tuning. Next, we show that both methods and the corresponding bounds can be adapted to the statistical setting. Specifically, the aggressive variant can be converted into a multi-task learning method, whereas the lazy variant can be translated into a meta-learning method, generalizing also to new tasks. Finally, we test the performance of our methods in numerical experiments.

Paper Organization. We start from describing our setting and recalling some basics on parameter-free online learning that will be employed throughout this work in Sec. 2 and Sec. 3, respectively. In Sec. 4 we introduce the biased family of within-task algorithms our method is based on. After this, in Sec. 5, we justify our choice, characterizing the settings in which an appropriate choice of the bias can bring advantages over learning the tasks independently. In Sec. 6 we describe the aggressive variant of our method and we show that it is able to infer a "good" bias vector from a sequence of tasks' datasets providing comparable guarantees to the best bias vector in hindsight. In Sec. 7 we describe how the method can be converted into a multi-task method in the statistical setting. The description and the analysis of the lazy variant of the method are postponed to App. D. Finally, in Sec. 8 we test our method in practice and in Sec. 9 we draw our conclusion. The proofs we skipped in the main body are postponed to the appendix.

Previous Work. The idea of inferring a common bias vector shared among a set of low-variance tasks is a well-established and simple approach. It was originally investigated in the multi-task learning setting for a finite set of tasks [7, 16, 23]. The success of this approach in this setting motivated its application also to meta-learning, both in batch and online [4, 11, 13, 14, 19, 29] fashion. The problem of inferring a good bias shared among a set of tasks is also closely related to the fine tuning problem (see e.g. [17]), where the goal is to find a good starting point for a specific family of learning algorithms over a set of tasks. All the works mentioned above are innovative in their own aspects, however, they require tuning at least one hyper-parameter. Among them, [19] is perhaps the most careful in this aspect, since it develops methods in which the hyper-parameters are adaptively chosen, but in order to reach this target, the authors require to constrain the weight vectors to a bounded set. This does not solve completely the issue above, since in practice one has still to choose an appropriate set. The critical aspect of designing parameter-free online algorithms has been already pointed out and addressed in the single task setting, see e.g. [27, 28, 32] and references therein. In this work we show how ideas developed in those papers for the single-task setting can be applied to design online multi-task learning and meta-learning methods that are well suited to low-variance sequences of tasks and do not require tuning any hyper-parameter.

2 SETTING

In this work, we consider the OWO setting outlined in [4, 14, 19], in which, the learner is asked to tackle a sequence of online supervised tasks.

Each task is associated to an input space $\mathcal{X}$ and an output space $\mathcal{Y}$. The learner incrementally receives a sequence of datapoints $Z = (z_i)_{i=1}^n = (x_i, y_i)_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$ from the task and is asked to make a prediction after each point is observed. Specifically, at each step $i \in \{1, \dots, n\}$: (a) a datapoint $z_i = (x_i, y_i)$ is observed, (b) the learner incurs the error $\ell_i(\hat{y}_i)$, where $\ell_i(\cdot) = \ell(\cdot, y_i)$ for a loss function ℓ and $\hat{y}_i$ is the current outcome (prediction) of the algorithm, (c) the algorithm updates its prediction $\hat{y}_{i+1}$ using the last point it has received. Throughout we let $\mathcal{X} \subseteq \mathbb{R}^d$, $\mathcal{Y} \subseteq \mathbb{R}$ and we consider algorithms that perform linear predictions of the form $\hat{y}_i = \langle x_i, w_i \rangle$, where $(w_i)_{i=1}^n$ is a sequence of weight vectors updated by the algorithm and $\langle \cdot, \cdot \rangle$ denotes the standard inner product in $\mathbb{R}^d$. This assumption can be relaxed by introducing a feature map on the inputs. The performance of the algorithm is evaluated by looking at the regret of its iterates over the dataset Z, i.e.

$$\sum_{i=1}^n \ell_i(\langle x_i, w_i \rangle) - \min_{w \in \mathbb{R}^d} \sum_{i=1}^n \ell_i(\langle x_i, w \rangle). \qquad (1)$$

The algorithm we will use in our framework is identified by a bias (meta-parameter) $\theta \in \mathbb{R}^d$ and the aim is to adapt θ to a sequence of learning tasks. To this end, we introduce one more algorithm (a meta-algorithm) that updates the bias as the tasks are incrementally observed. We consider two variants of such an algorithm. The first one updates the bias after each point is observed and the second variant updates the bias only at the end of each task's training sequence. As we shall see, the main advantage of the first strategy will be to obtain faster learning bounds. However, when we move to the statistical setting, the first variant can be converted into a multi-task

Algorithm 1 One-Dimension Coin Betting Algorithm based on Krichevsky-Trofimov (KT) estimator, [28, Alg. 1]

Input $(g_k)_{k=1}^K$, $g_k \in \mathbb{R}$, $|g_k| \leq C$, $\epsilon > 0$
Initialize $b_1 = 0$, $u_1 = \epsilon$, $p_1 = b_1 u_1$
For $k = 1, \dots, K$
  Receive g_k
  Define $u_{k+1} = u_k - \frac{1}{C} g_k p_k$
  Define $b_{k+1} = \frac{1}{k}\big((k-1)b_k - \frac{1}{C} g_k\big)$
  Update $p_{k+1} = b_{k+1} u_{k+1}$
End
Return $(p_k)_{k=1}^K$

Algorithm 2 Online Projected Subgradient Algorithm, [18, Alg. 6]

Input $\mathcal{B} \subset \mathbb{R}^d$, $(g_k)_{k=1}^K$, $g_k \in \mathbb{R}^d$, $\|g_k\| \leq C$
Initialize $v_1 \in \mathcal{B}$
For $k = 1, \dots, K$
  Receive g_k
  Define $\gamma_k = \frac{\operatorname{diam}(\mathcal{B})}{C\sqrt{2k}}$
  Update $v_{k+1} = \operatorname{proj}_{\mathcal{B}}\big(v_k - \gamma_k g_k\big)$
End
Return $(v_k)_{k=1}^K$

learning method, while, the second into a meta-learning method, able to generalize also across the tasks.

More precisely, denoting by T the number of tasks, for each task $t \in \{1, \dots, T\}$, we let $Z_t = (x_{t,i}, y_{t,i})_{i=1}^n$ be the corresponding data sequence. Throughout this work, we follow the convention adopted in [14] and we use the double subscript notation "$_{t,i}$", to denote the {outer, inner} task index. While, we use $k = k(t,i) = (t-1)n + i \in \{1, \dots, Tn\}$ to denote the index counting the global number of datapoints received by the algorithm. At each time $k = k(t,i)$: (a) the algorithm receives the point $z_{t,i} = (x_{t,i}, y_{t,i})$, (b) the algorithm incurs the error $\ell_{t,i}(\langle x_{t,i}, w_{t,i} \rangle)$, where $\ell_{t,i}(\cdot) = \ell(\cdot, y_{t,i})$ and $w_{t,i}$ is the current within-task iteration, (c) the bias (and consequently, the inner algorithm) is updated in θ_{k+1} for the aggressive variant or it is kept frozen to θ_t for the lazy variant until the entire task's dataset has been observed, (d) the algorithm performs one updating step by the inner algorithm with the current meta-parameter, returning the predictor vector $w_{t,i+1}$. In a very natural way, the performance of the entire procedure above is measured by the regret accumulated across the tasks, i.e.

$$\sum_{t=1}^{T} \left(\sum_{i=1}^{n} \ell_{t,i}(\langle x_{t,i}, w_{t,i} \rangle) - \min_{w_t \in \mathbb{R}^d} \sum_{i=1}^{n} \ell_{t,i}(\langle x_{t,i}, w_t \rangle) \right).$$

We conclude this section by introducing the following standard assumption which will be used in the following.

Assumption 1 (Bounded Inputs and Convex Lipschitz Loss)**.** *Let $\ell(\cdot, y)$ be convex and L-Lipschitz for any $y \in \mathcal{Y}$ and let $\mathcal{X} \subseteq \mathcal{B}(0, R)$, where, for any center $c \in \mathbb{R}^d$ and radius $r > 0$, we have introduced the Euclidean ball*

$$\mathcal{B}(c, r) = \Big\{ v \in \mathbb{R}^d : \|v - c\| \leq r \Big\}. \tag{2}$$

3 PRELIMINARIES

Our method is based on parameter-free online learning. In this section, we briefly recall two well-known parameter-free online algorithms: the one-dimension coin betting algorithm in Alg. 1 and the online projected subgradient algorithm in Alg. 2. In the following, we will use these algorithms to build our framework. We note that Alg. 2 does not require tuning any hyper-parameter and Alg. 1 requires choosing just one hyper-parameter (the initial wealth $\epsilon > 0$). However, as we will see in the following, there is a quite wide range in which the choice of such a hyper-parameter does not affect the overall performance of the algorithm. For this reason, both Alg. 1 and Alg. 2 can be considered parameter-free algorithms. We start from describing Alg. 1.

One-Dimension Coin Betting Algorithm. Alg. 1 coincides with the scalar version of the Krichevsky-Trofimov (KT) algorithm described in [28, Alg. 1]. The algorithm takes in input an initial wealth $\epsilon > 0$. At each iteration k, the algorithm receives a value $g_k \in \mathbb{R}$ with absolute value $|g_k| \leq C$ for some $C > 0$, it updates a betting fraction $b \in \mathbb{R}$ and a wealth $u \in \mathbb{R}$ and, then, it multiplies them together to update the global iteration $p = bu$. The linear regret of Alg. 1 can be bounded as described in the following proposition.

Proposition 1 (Regret Bound for Alg. 1, [28, Cor. 5])**.** *The iterations $(p_k)_{k=1}^K$ returned by Alg. 1 satisfy the following linear regret bound w.r.t. a competitor scalar $p \in \mathbb{R}$*

$$\sum_{k=1}^{K} g_k (p_k - p) \leq C \left[\epsilon + \Phi\Big(\epsilon^{-1} |p| K\Big) |p| \sqrt{K} \right] \tag{3}$$

where, for any $a \in \mathbb{R}$, we have introduced the function $\Phi(a) = \sqrt{\log\big(1 + 24a^2\big)}$.

As we can see from the bound above, the dependency of the bound on the hyper-parameter ϵ is not problematic; any value in $[1, \sqrt{K}]$ does not affect the $\sqrt{K}$ rate. We now recall the main properties of Alg. 2.

Projected Online Subgradient Algorithm. Alg. 2 coincides with [18, Alg. 6]. The algorithm takes in input a

convex, closed and non-empty set $\mathcal{B} \subset \mathbb{R}^d$ with diameter

$$\operatorname{diam}(\mathcal{B}) = \sup_{v,v' \in \mathcal{B}} \|v - v'\|. \tag{4}$$

At each iteration k, the algorithm receives a vector $g_k \in \mathbb{R}^d$ with norm $\|g_k\| \leq C$ for some $C > 0$, it performs a descent step along this vector with an appropriate length and, then, it projects the resulting vector on the set $\mathcal{B}$. The linear regret of Alg. 2 can be bounded as described in the following proposition.

Proposition 2 (Regret Bound for Alg. 2, [18, Thm. 3.1])**.** *The iterations $(v_k)_{k=1}^K$ returned by Alg. 2 satisfy the following linear regret bound w.r.t. a competitor vector $v \in \mathcal{B}$*

$$\sum_{k=1}^{K} \langle g_k, v_k - v \rangle \leq C\sqrt{2}\operatorname{diam}(\mathcal{B})\sqrt{K}. \tag{5}$$

We now have all the ingredients necessary to introduce the family of within-task algorithms.

4 BIASED PARAMETER-FREE ONLINE ALGORITHM

In this section, we consider a family of within-task algorithms parametrized by a bias vector $\theta \in \mathbb{R}^d$. The idea of introducing a bias is a well-established approach in the multi-task learning and meta-learning literature, see e.g. [4, 7, 11, 13, 14, 16, 19, 23, 29]. However, a key novel aspect of our work is to focus on a family of parameter-free algorithms – we are not aware of previous work dealing with a similar framework within the multi-task learning or meta-learning literature. Such a choice allows us to avoid expensive validation procedures which are not even allowed in the so-called 'adversarial setting', where the learner is asked to make predictions on the fly, after observing data only once. Specifically, the algorithm we choose is reported in Alg. 3 and it coincides with a variant of the online parameter-free [10, Alg. 2] in which we add a translation w.r.t. a bias vector $\theta \in \mathbb{R}^d$, which is specified in advanced to the algorithm.

Similarly to the discussion in [10], the motivation behind the algorithm comes from the following simple observation. For a fixed bias vector $\theta \in \mathbb{R}^d$, we can always rewrite any vector $w \in \mathbb{R}^d$ w.r.t. the coordinate system centered in θ:

$$w = pv + \theta \tag{6}$$

where

$$p = \|w - \theta\| \in \mathbb{R} \qquad v = \frac{w - \theta}{\|w - \theta\|} \in \mathcal{B}(0, 1). \tag{7}$$

Alg. 3 receives in input the bias vector $\theta \in \mathbb{R}^d$ and, exploiting the decomposition above, it uses the datapoints

Algorithm 3 Parameter-Free Algorithm with Fixed Bias, Biased Version of [10, Alg. 2]

Input $\theta \in \mathbb{R}^d$, $Z = (z_i)_{i=1}^n = (x_i, y_i)_{i=1}^n$, $e > 0$, L and R as in Asm. 1

Initialize $b_1 = 0$, $u_1 = e$, $p_1 = b_1 u_1$, $v_1 = 0 \in \mathcal{B}(0, 1)$

For $i = 1, \dots, n$

1. Vector update $w_i = p_i v_i + \theta$

2a. Receive the datapoint $z_i = (x_i, y_i)$

2b. Compute $g_i = s_i x_i$, $s_i \in \partial \ell_i(\langle x_i, w_i \rangle) \in \mathbb{R}$

3a. Define $\gamma_i = \frac{1}{LR}\sqrt{\frac{2}{i}}$

3b. Direction update $v_{i+1} = \operatorname{proj}_{\mathcal{B}(0,1)}\big(v_i - \gamma_i g_i\big)$

4a. Define $u_{i+1} = u_i - \frac{1}{RL}\langle g_i, v_i \rangle p_i$

4b. Define $b_{i+1} = \frac{1}{i}\big((i-1)b_i - \frac{1}{RL}\langle g_i, v_i \rangle\big)$

4c. Magnitude update $p_{i+1} = b_{i+1} u_{i+1}$

End

Return $(w_i)_{i=1}^n$

$Z = (z_i)_{i=1}^n = (x_i, y_i)_{i=1}^n$ it receives, in order to incrementally learn

- the direction $v \in \mathcal{B}(0, 1)$ of the vector $w - \theta$ by applying Alg. 2 on the ball $\mathcal{B}(0, 1)$ to the subgradient vectors $(g_i)_{i=1}^n$, where $g_i \in \partial \ell_i(\langle x_i, \cdot \rangle)(w_i)$, with w_i the current global vector returned by the algorithm (steps $3a$-b),

- the magnitude p of the vector $w - \theta$ by applying Alg. 1 to the scalars $(\langle g_i, v_i \rangle)_{i=1}^n$, with v_i the current direction w.r.t. the bias vector θ (steps $4a$-c).

We remark that, in order to update the magnitude p, differently from Alg. 3 in which we use the KT algorithm in Alg. 1, the authors in [10, Alg. 2] use a more sophisticate coin betting algorithm based on online Newton step, see [10, Alg. 1]. This allows them to get more refined regret bounds, which can bring an advantage for instance in the smooth setting. In this work, we employ a simplified version of the algorithm for the theoretical analysis, since the derived regret bounds are simpler and, at the same time, such a simplification does not affect the main message we want to convey.

In the following result we report a regret bound for Alg. 3. We make no claim of originality in the proof of the above result, which is a simple adaptation of the proof technique of [10, Thm. 2], adding the translation w.r.t. the bias and changing the coin betting algorithm to estimate the magnitude, as explained above. We provide below the main ideas used in the proof of the statement because they will be used also in the following. The full proof is reported in App. A for completeness.

Proposition 3 (Single-Task Regret Bound for Alg. 3, Adaptation of [10, Thm. 2]). *Let Asm. 1 hold and let $(w_i)_{i=1}^n$ be the iterates generated by Alg. 3 with bias $\theta \in \mathbb{R}^d$. Then, for any $w \in \mathbb{R}^d$,*

$$\sum_{i=1}^n \ell_i(\langle x_i, w_i\rangle) - \ell_i(\langle x_i, w\rangle) \leq \sum_{i=1}^n \langle g_i, w_i - w\rangle$$
$$\leq RL\left[e + \left(2\sqrt{2} + \Phi\Big(e^{-1}\|w-\theta\|n\Big)\right)\|w-\theta\|\sqrt{n}\right]$$

where, $\Phi(\cdot)$ is defined as in Prop. 1.

Proof Sketch. While the first inequality follows by the convexity of the loss function (see Asm. 1) and the definition of the subgradients $(g_i)_{i=1}^n$, the proof of the second inequality is essentially based on the magnitude-direction decomposition used in the algorithm and explained above. Specifically, by definition of the w_i in Alg. 3 and the rewriting of $w \in \mathbb{R}^d$ as in Eq. (6)–Eq. (7), one can show that the linear regret can be bounded by the sum of two terms,

$$\sum_{i=1}^n \langle g_i, w_i - w\rangle \leq R(p) + pR(v), \tag{8}$$

where, p and v are defined in Eq. (7) and

$$R(p) = \sum_{i=1}^n \langle g_i, v_i\rangle(p_i - p) \quad R(v) = \sum_{i=1}^n \langle g_i, v_i - v\rangle$$

coincide, respectively, with the regret of the magnitudes $(p_i)_{i=1}^n$ generated by Alg. 1 and the regret of the directions $(v_i)_{i=1}^n$ generated by Alg. 2. The statement then follows from exploiting Asm. 1 in order to bound the two terms by Prop. 1 and Prop. 2, respectively. ■

As observed in [10], the leading term in the bound above is equivalent – up to the logarithmic factor contained in the term $\Phi(e^{-1}\|w-\theta\|n)$ – to the optimal bound $\mathcal{O}(\|w-\theta\|\sqrt{n})$ one would get by using a translated version of online subgradient algorithm

$$w_1 = 0 \in \mathbb{R}^d \quad w_i = w_{i-1} - \gamma g_{i-1} + \theta \quad i \geq 2, \tag{9}$$

with oracle-tuning of the step-size $\gamma > 0$ requiring knowledge of the target vector's magnitude $\|w-\theta\|$ in hindsight. Moreover, since the bound matches available lower-bounds, such additional logarithmic terms are unavoidable and they represent the price we pay by estimating the magnitude from the data. We also notice that, by induction argument it is easy to show that the translated iteration in Eq. (9) coincides with standard (untranslated) online subgradient algorithm with initial point θ. As a consequence, Alg. 3 can be also interpreted as a parameter-free variant of the standard family used in fine tuning meta-learning [17].

5 MOTIVATION FOR THE BIAS

In this section, we study the advantage of using an appropriate bias term in Alg. 3. Specifically, we study the performance obtained by applying Alg. 3 with the same bias vector θ over a sequence of T datasets $\mathbf{Z} = (Z_t)_{t=1}^T$, $Z_t = (z_{t,i})_{i=1}^n = (x_{t,i}, y_{t,i})_{i=1}^n$ deriving from T different tasks w.r.t. a sequence of target vectors $(w_t)_{t=1}^T$ associated to the tasks. We are implicitly parametrizing the vector associated to each task as in Eq. (6)–Eq. (7), according to the same bias vector $\theta \in \mathbb{R}^d$:

$$w_t = p_t v_t + \theta \tag{10}$$

$$p_t = \|w_t - \theta\| \in \mathbb{R} \quad v_t = \frac{w_t - \theta}{\|w_t - \theta\|} \in \mathcal{B}(0,1). \tag{11}$$

This situation is analyzed below.

Corollary 4 (Across-Tasks Regret Bound for Alg. 3). *Let Asm. 1 hold. Consider T datasets $\mathbf{Z} = (Z_t)_{t=1}^T$, $Z_t = (z_{t,i})_{i=1}^n = (x_{t,i}, y_{t,i})_{i=1}^n$ deriving from T different tasks. For any task $t = 1, \dots, T$, let $(w_{t,i})_{i=1}^n$ be the iterates generated by Alg. 3 over the dataset Z_t with bias θ. Then, for any sequence $(w_t)_{t=1}^T$, $w_t \in \mathbb{R}^d$,*

$$\sum_{t=1}^T \sum_{i=1}^n \ell_{t,i}(\langle x_{t,i}, w_{t,i}\rangle) - \ell_{t,i}(\langle x_{t,i}, w_t\rangle)$$
$$\leq \sum_{t=1}^T \sum_{i=1}^n \langle g_{t,i}, w_{t,i} - w_t\rangle$$
$$\leq RL\left[eT + \Big(2\sqrt{2}\mathrm{Var}(\theta) + \widehat{\mathrm{Var}}(\theta)\Big)\sqrt{n}T\right] \tag{12}$$

where

$$\mathrm{Var}(\theta) = \frac{1}{T}\sum_{t=1}^T \|w_t - \theta\|, \tag{13}$$

$$\widehat{\mathrm{Var}}(\theta) = \frac{1}{T}\sum_{t=1}^T \Phi\Big(e^{-1}\|w_t-\theta\|n\Big)\|w_t - \theta\| \tag{14}$$

and the function $\Phi(\cdot)$ is defined in Prop. 1.

Proof. The statement directly derives from summing over the datasets the regret bound in Prop. 3. ■

Even though the quantity in Eq. (13) does not coincide with the variance of the target vectors $(w_t)_{t=1}^T$ w.r.t. the bias θ, with some abuse of notation, we are denoting such a quantity by $\mathrm{Var}(\theta)$. To be precise, Eq. (13) represents a lower-bound for the variance, indeed, by Jensen's inequality we have that $\mathrm{Var}(\theta) \leq \sqrt{\frac{1}{T}\sum_{t=1}^T \|w_t - \theta\|^2}$. We observe that, when the logarithmic term is negligible (i.e. $\phi(e^{-1}\|w_t - \theta\|n) \approx 1$) for any task $t \in \{1, \dots, T\}$,

then $\widehat{\mathrm{Var}}(\theta) \approx \mathrm{Var}(\theta)$. As a consequence, in such a case, the leading term in the bound above is proportional to

$$\mathcal{O}\Big(\mathrm{Var}(\theta)\sqrt{nT}\Big). \tag{15}$$

The conclusion we get from the bound above is exactly in line with previous literature addressing the same problem, but by means of methods requiring the tuning of at least one hyper-parameter, see e.g. [4, 11, 14, 19]. Specifically, the bound above suggests that the optimal choice for the bias θ in Alg. 3 is the one minimizing the variance of the target vectors $(w_t)_{t=1}^T$, namely, their empirical average:

$$\operatorname*{argmin}_{\theta\in\mathbb{R}^d} \sqrt{\frac{1}{T}\sum_{t=1}^T \|w_t-\theta\|^2} = \frac{1}{T}\sum_{t=1}^T w_t. \tag{16}$$

Moreover, our analysis confirms the conclusion in [4, 11, 14, 19]: the advantage of using such an optimal bias w.r.t. the unbiased case (corresponding to solving the tasks independently) is significant when the variance of the tasks' target vectors is much smaller than their second moment.

6 LEARNING THE BIAS

Motivated by the conclusion in the previous section, we now propose and analyze a parameter-free method to infer a good bias vector shared across the tasks from an increasing sequence of T datasets $\mathbf{Z} = (Z_t)_{t=1}^T$, $Z_t = (z_{t,i})_{i=1}^n = (x_{t,i}, y_{t,i})_{i=1}^n$. The method is reported in Alg. 4 and it updates the bias vector θ after each point. This characteristic inspires us to refer to Alg. 4 as 'aggressive', in order to distinguish it from the 'lazy' version reported in Alg. 5 in App. D, where, the bias vector is updated only at the end of each task.

The idea motivating the design of our method is similar to the idea in Sec. 5 of parametrizing the vector associated to each task as in Eq. (10)–Eq. (11), according to a common bias vector $\theta \in \mathbb{R}^d$. But, now, we also parametrize the bias vector θ w.r.t. the zero-centered coordinate system:

$$\theta = PV \tag{17}$$

$$P = \|\theta\| \in \mathbb{R} \qquad V = \frac{\theta}{\|\theta\|} \in \mathcal{B}(0,1). \tag{18}$$

Alg. 4 exploits the *joint* parametrization above and it uses the datasets $\mathbf{Z}$ it receives, in order to incrementally learn

- (for any task t) the direction $v_t \in \mathcal{B}(0,1)$ of the vector $w_t - \theta$ by applying Alg. 2 on the ball $\mathcal{B}(0,1)$ to the subgradient vectors $(g_{t,i})_{i=1}^n$, where $g_{t,i} \in \partial\ell_{t,i}(\langle x_{t,i}, \cdot\rangle)(w_{t,i})$, with $w_{t,i}$ the current within-task iteration returned by the algorithm (steps $3a$-b),

Algorithm 4 Parameter-Free Algorithm with Bias Inferred from Data, Aggressive Version

Input $\mathbf{Z} = (Z_t)_{t=1}^T$, $Z_t = (z_{t,i})_{i=1}^n = (x_{t,i}, y_{t,i})_{i=1}^n$, $e > 0$, $E > 0$, L and R as in Asm. 1
Initialize $B_1 = 0, U_1 = E, P_1 = B_1U_1, V_1 = 0 \in \mathcal{B}(0,1)$
For $t = 1, \dots, T$
  Set $b_{t,1} = 0$, $u_{t,1} = e$, $p_{t,1} = b_{t,1}u_{t,1}$, $v_{t,1} = 0 \in \mathcal{B}(0,1)$
  For $i = 1, \dots, n$
    0. Define $k = k(t,i) = (t-1)n + i$
    1. Meta-vector update $\theta_k = P_kV_k$
    1. Within-vector update $w_{t,i} = p_{t,i}v_{t,i} + \theta_k$
    2a. Receive the datapoint $z_{t,i} = (x_{t,i}, y_{t,i})$
    2b. Compute $g_{t,i} = s_{t,i}x_{t,i}$, $s_{t,i} \in \partial\ell_{t,i}(\langle x_{t,i}, w_{t,i}\rangle)$
    3A. Define $\eta_k = \frac{1}{LR}\sqrt{\frac{2}{k}}$
    3B. Define $V_{k+1} = \mathrm{proj}_{\mathcal{B}(0,1)}\big(V_k - \eta_k g_{t,i}\big)$
    3a. Define $\gamma_{t,i} = \frac{1}{LR}\sqrt{\frac{2}{i}}$
    3b. Update $v_{t,i+1} = \mathrm{proj}_{\mathcal{B}(0,1)}\big(v_{t,i} - \gamma_{t,i}g_{t,i}\big)$
    4A. Define $U_{k+1} = U_k - \frac{1}{RL}\langle g_{t,i}, V_k\rangle P_k$
    4B. Define $B_{k+1} = \frac{1}{k}\big((k-1)B_k - \frac{1}{RL}\langle g_{t,i}, V_k\rangle\big)$
    4C. Update $P_{k+1} = B_{k+1}U_{k+1}$
    4a. Define $u_{t,i+1} = u_{t,i} - \frac{1}{RL}\langle g_{t,i}, v_{t,i}\rangle p_{t,i}$
    4b. Define $b_{t,i+1} = \frac{1}{i}\big((i-1)b_{t,i} - \frac{1}{RL}\langle g_{t,i}, v_{t,i}\rangle\big)$
    4c. Update $p_{t,i+1} = b_{t,i+1}u_{t,i+1}$
  End
End
Return $(w_{t,i})_{t=1,i=1}^{T,n}$ and $(\theta_k)_{k=1}^{Tn}$

- (for any task t) the magnitude p_t of the vector $w_t - \theta$, by applying Alg. 1 to the scalars $(\langle g_{t,i}, v_{t,i}\rangle)_{i=1}^n$, with $v_{t,i}$ the current within-task direction w.r.t. the current bias vector $\theta_{k(t,i)}$ estimated by the algorithm and $k(t,i) = (t-1)n + i$ the total number of points seen up to that moment (steps $4a$–c),

- the direction $V \in \mathcal{B}(0,1)$ of the vector θ, by applying Alg. 2 on the ball $\mathcal{B}(0,1)$ to the vectors $(g_{t,i})_{t,i=1}^{T,n}$ (steps $3A$-B),

- the magnitude P of the vector θ, by applying Alg. 1 to the scalars $(\langle g_{t,i}, V_{k(t,i)}\rangle)_{t=1,i=1}^{T,n}$, with $V_{k(t,i)}$ the current meta-direction estimated by the algorithm (steps $4A$-C).

The performance of Alg. 4 is analyzed in the following theorem in which we give an across-tasks regret bound for the method. The complete proof of the statement is reported in App. B.

Theorem 5 (Across-Tasks Regret Bound for Alg. 4)**.** *Let Asm. 1 hold. Consider T datasets $\mathbf{Z} = (Z_t)_{t=1}^T$,*

$Z_t = (z_{t,i})_{i=1}^n = (x_{t,i}, y_{t,i})_{i=1}^n$ deriving from T different tasks. Let $(w_{t,i})_{t=1,i=1}^{T,n}$ be the iterates generated by Alg. 4 over these datasets $\mathbf{Z}$. Then, for any sequence $(w_t)_{t=1}^T$, $w_t \in \mathbb{R}^d$ and any $\theta \in \mathbb{R}^d$,

$$\sum_{t=1}^{T}\sum_{i=1}^{n} \ell_{t,i}(\langle x_{t,i}, w_{t,i}\rangle) - \ell_{t,i}(\langle x_{t,i}, w_t\rangle) \leq \sum_{t=1}^{T}\sum_{i=1}^{n} \langle g_{t,i}, w_{t,i} - w_t\rangle \leq A + B, \tag{19}$$

where, A is the term in Cor. 4 with θ,

$$B = RL\left[E + \left(2\sqrt{2} + \Phi\left(E^{-1}\|\theta\|nT\right)\right)\|\theta\|\sqrt{nT}\right]$$

and $\Phi(\cdot)$ is defined as in Prop. 1.

Proof Sketch. While the first inequality is due to the convexity of the loss function (see Asm. 1) and the definition of the subgradients $(g_{t,i})_{t=1,i=1}^{T,n}$, the proof of the second inequality is based, also in this case, on the joint magnitude-direction decomposition motivating the design of the algorithm and explained above. Specifically, by definition of $w_{t,i}$ and θ_k in Alg. 4 and the rewriting of θ as in Eq. (17)–Eq. (18) and w_t as in Eq. (10)–Eq. (11), one can show that the linear regret can be bounded by four contributions as follows:

$$\sum_{t=1}^{T}\sum_{i=1}^{n} \langle g_{t,i}, w_{t,i} - w_t\rangle \leq \sum_{t=1}^{T}\left(R_t(p_t) + p_t R_t(v_t)\right) + R(P) + PR(V),$$

where, p_t and v_t as in Eq. (11), P and V as in Eq. (18),

$$R_t(p_t) = \sum_{i=1}^{n} \langle g_{t,i}, v_{t,i}\rangle(p_{t,i} - p_t) \tag{20}$$

$$R_t(v_t) = \sum_{i=1}^{n} \langle g_{t,i}, v_{t,i} - v_t\rangle \tag{21}$$

$$R(P) = \sum_{t=1}^{T}\sum_{i=1}^{n} \langle g_{t,i}, V_{k(t,i)}\rangle\left(P_{k(t,i)} - P\right) \tag{22}$$

$$R(V) = \sum_{t=1}^{T}\sum_{i=1}^{n} \langle g_{t,i}, V_{k(t,i)} - V\rangle \tag{23}$$

coincide, respectively, with the within-task regret of the magnitudes $(p_{t,i})_{i=1}^n$ generated by Alg. 1 on the task t, the within-task regret of the directions $(v_{t,i})_{i=1}^n$ generated by Alg. 2 on the task t, the meta-regret of the magnitudes $(P_k)_{k=1}^K$ generated by Alg. 1 and the meta-regret of the directions $(V_k)_{k=1}^K$ generated by Alg. 2. The statement derives from exploiting Asm. 1 in order to bound the four terms by Prop. 1 or Prop. 2, accordingly. ■

We note that the bound in Thm. 5 is composed of two main terms: while the term A coincides with the bound in Cor. 4 for the use of a pre-fixed bias θ across all the tasks, the term B captures the price we pay to estimate the bias from data. Notice that this additional term goes as $\mathcal{O}(\sqrt{nT})$ and, as a consequence, it is negligible when added to the first term going as $\mathcal{O}(\sqrt{n})$. In particular, specifying the bound in Thm. 5 to the bias θ in Eq. (16) (the average of the target tasks' weight vectors), we can conclude that our method is able to match the performance of this best bias in hindsight, when the number of tasks is sufficiently large. On the other hand, by taking $\theta = 0 \in \mathbb{R}^d$ in Thm. 5, we retrieve the bound in Cor. 4 for independent task learning (ITL). Hence, in the worst-case scenario of no low-variance tasks, our method performs, at least, as ITL, without negative transfer effect. We also observe that the additional term due to the estimation of the bias from the data is faster in comparison to the additional term going as $\mathcal{O}(n\sqrt{T})$ paid in benchmark works for growing tasks' sequences requiring hyper-parameter tuning, such as [14]. This is essentially due to the fact that in our method we are updating the bias more frequently: after each point (hence nT updates) instead of only at the end of each task (hence T updates) as done in [14]. We finally notice that the bound in Thm. 5 present a similar rate to the mistakes' bound in [7, Cor. 4] for a Perceptron-based algorithm. However, the method in [7] works only for finite sequences of tasks and, again, it requires hyper-parameter tuning.

7 STATISTICAL MTL SETTING

In this section we show how Alg. 4 can be adapted to a multi-task learning statistical setting. Specifically, following the framework outlined in [6], we assume that, for any $t \in \{1, \ldots, T\}$, the within-task dataset Z_t is an independently identically distributed (i.i.d.) sample from a distribution (task) μ_t.

In this case, for any task $t \in \{1, \ldots, T\}$, we consider the estimator $\bar{w}_t = \frac{1}{n}\sum_{i=1}^n w_{t,i}$ given by the average of the iterations computed by Alg. 4 associated to the task t. We wish to study the performance of such estimators. Formally, for any task μ_t, we require that the corresponding true risk $\mathcal{R}_{\mu_t}(w) = \mathbb{E}_{(x,y)\sim\mu_t}\ell(\langle x, w\rangle, y)$ admits minimizers over the entire space $\mathbb{R}^d$ and we denote by w_{μ_t} the minimum norm one. With these ingredients, we introduce the multi-task oracle $\mathcal{E}^*_{\text{MTL}} = \frac{1}{T}\sum_{t=1}^T \mathcal{R}_{\mu_t}(w_{\mu_t})$, and, introducing the *average multi-task risk* of the estimators $(\bar{w}_t)_{t=1}^T$:

$$\mathcal{E}_{\text{MTL}}\left((\bar{w}_t)_{t=1}^T\right) = \frac{1}{T}\sum_{t=1}^{T} \mathcal{R}_{\mu_t}(\bar{w}_t), \tag{24}$$

we give a bound on it w.r.t. the oracle $\mathcal{E}^*_{\text{MTL}}$. This is

described in the following theorem.

Theorem 6 (Multi-Task Risk Bound for Alg. 4)**.** *Let the same assumptions in Thm. 5 hold in the i.i.d. multi-task statistical setting. Let* $\mathcal{E}_{\mathrm{MTL}}\big((\bar{w}_t)_{t=1}^T\big)$ *be as in Eq. (24), namely, the average multi-task risk of the estimators* $(\bar{w}_t)_{t=1}^T$*, where* $\bar{w}_t$ *is the average of the iterates computed by Alg. 4 associated to the task* t*. Then, for any* $\theta \in \mathbb{R}^d$*, in expectation w.r.t. the sampling of the datasets* $\mathbf{Z} = (Z_t)_{t=1}^T$*,*

$$\mathbb{E}_{\mathbf{Z}}\, \mathcal{E}_{\mathrm{MTL}}\big((\bar{w}_t)_{t=1}^T\big) - \mathcal{E}^*_{\mathrm{MTL}} \leq \frac{1}{nT}(A + B) \qquad (25)$$

where,

$$A = RL\left[eT + \left(2\sqrt{2}\mathrm{Var}_{\mathrm{MTL}}(\theta) + \widehat{\mathrm{Var}}_{\mathrm{MTL}}(\theta)\right)\sqrt{nT}\right]$$

$$\mathrm{Var}_{\mathrm{MTL}}(\theta) = \frac{1}{T}\sum_{t=1}^T \|w_{\mu_t} - \theta\| \qquad (26)$$

$$\widehat{\mathrm{Var}}_{\mathrm{MTL}}(\theta) = \frac{1}{T}\sum_{t=1}^T \Phi\Big(e^{-1}\|\hat{w}_{\mu_t} - \theta\| n\Big)\|w_{\mu_t} - \theta\|$$

$\Phi(\cdot)$ *is defined as in Prop. 1 and* B *is the term in Thm. 5.*

The bound we have obtained above for our parameter-free method is in line with previous batch multi-task learning literature [16, 23] requiring tuning of hyper-parameters. Regarding online benchmarks, it is unclear whether the online method proposed in [7] can be adapted also to a statistical setting. We observe that the bound above is composed by the expectation of the terms comparing in Thm. 5 evaluated at the target vectors $(w_{\mu_t})_{t=1}^T$. This automatically derives from the fact that the proof of the statement exploits the across-tasks regret bound given in Thm. 5 for our meta-learning procedure and, as described in the following proposition, online-to-batch conversion arguments [9, 21]. The statement reported below is used by the authors in [15] in order to address the issue of applying stochastic subgradient descent to a sequence of semy-cyclic datapoints by plurastic (multi-task) point of view. We report the proof in App. C for completeness.

Proposition 7 (Online-To-Batch Conversion for Alg. 4, [15, Thm. 3])**.** *Under the same assumptions in Thm. 6, the following relation holds*

$$\mathbb{E}_{\mathbf{Z}}\, \mathcal{E}_{\mathrm{MTL}}\big((\bar{w}_t)_{t=1}^T\big) - \mathcal{E}^*_{\mathrm{MTL}} \leq$$
$$\mathbb{E}_{\mathbf{Z}}\left[\frac{1}{nT}\sum_{t=1}^T\sum_{i=1}^n \ell_{t,i}(\langle x_{t,i}, w_{t,i}\rangle) - \ell_{t,i}(\langle x_{t,i}, w_{\mu_t}\rangle)\right].$$

We now have all the ingredients necessary to prove Thm. 6.

Proof of Thm. 6. The desired statement derives from applying on the right side of Prop. 7 the across-tasks regret bound in Thm. 5 specified to the sequence of target vectors $(w_{\mu_t})_{t=1}^T$. ■

Looking at the proof in App. C, the reader can notice that the online-to-batch statement in Prop. 7 applies also to the 'lazy' version of our method reported in Alg. 5 in App. D. This allows us to convert also the lazy variant into a statistical (sub-optimal) multi-task learning method with a slower rate. On the other hand, we did not manage to convert the aggressive variant of our method into a statistical meta-learning method. This issue makes us wondering whether faster rates going as $\sqrt{nT}$ as in the multi-task learning setting are achievable also in the meta-learning setting for the second term.

8 EXPERIMENTS

In this section we test the numerical performance of our method[1]. Following the same data-generation procedure described in [11], we generated an environment of $T = 400$ regression tasks with low variance. Specifically, for any task μ, we sampled the corresponding ground truth vector w_μ from a Gaussian distribution with mean given by the vector $\theta^* \in \mathbb{R}^d$ with $d = 10$ and all components equal to 4 and standard deviation 1. After this, we generated the corresponding dataset $(x_i, y_i)_{i=1}^n$, $x_i \in \mathbb{R}^d$ with $n = 25$. We sampled the inputs uniformly on the unit sphere and we generated the labels according to the equation $y = \langle x, w_\mu\rangle + \epsilon$, where the noise ϵ was sampled from a zero-mean Gaussian distribution, with standard deviation chosen in order to have signal-to-noise ratio 1.

In this setting, we compared the performance of independent task learning (ITL) (running the unbiased variant of Alg. 3 over each task), our aggressive method in Alg. 4 (Aggr) and its lazy version in Alg. 5 in App. D (Lazy).

In the experiments below, we noticed that the variants of our methods estimating the magnitude by the refined coin betting algorithm in [10, Alg. 1] returned a more readable plot w.r.t. the variants described in our theory using the KT algorithm in Alg. 1. For this reason, we report below the results obtained using this more refined variant.

In Fig. 1 (top) we report the average across-tasks cumulative error for all the methods w.r.t. to an increasing number of datapoints/iterations. In Fig. 1 (bottom) we report their (statistical) average multi-task test errors for

[1] Code to reproduce the experiments is available at https://github.com/dstamos/Parameter-free-MTL

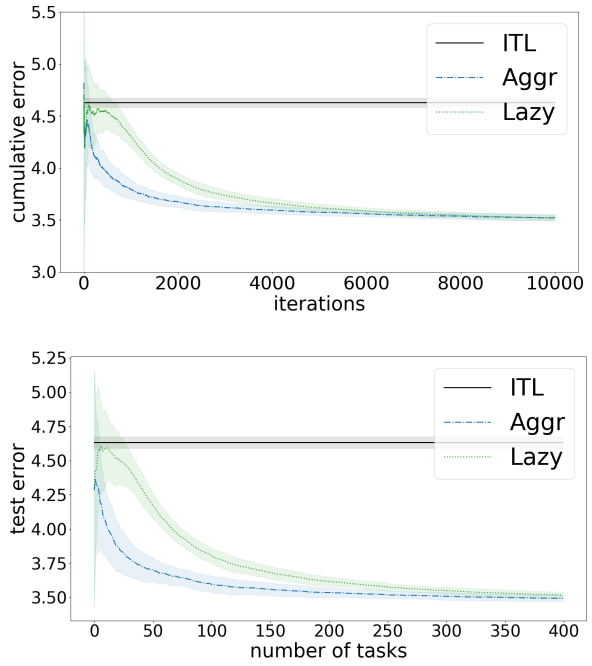

Figure 1: Average performance (over 30 seeds) of different methods w.r.t. an increasing number of iterations or tasks on synthetic data. Average across-tasks cumulative error (top), average multi-task test error (bottom).

an increasing number of tasks. We measured the performance by the absolute loss and we set the initial wealths in our methods equal to 1, for both the within-task and the across-tasks algorithms. The results we got are in agreement with the theory. Our approaches lead to a substantial benefits w.r.t. ITL and they converge to the oracle (the algorithm with the best bias in hindsight) as the number of the observed datapoints/tasks increases. Moreover, coherently with our bounds, we observe that, the aggressive variant of our method presents faster rates w.r.t. its lazy counterpart.

Because of lack of space, in App. E, we report additional experiments investigating the sensitivity of our parameter-free methods w.r.t. to the initialization of the wealths and showing the effectiveness of our methods on two real datasets (the Lenk [20, 26] and the Schools [3] datasets). In such a case, we will report for completeness both the refined and the basic variant of our methods.

9 CONCLUSION

We developed a parameter-free method that learns a common bias shared by a growing sequence of tasks. The advantage of our method in comparison to solving the tasks independently manifests itself when the variance of target tasks' weight vectors is sufficiently small. Our method is originally introduced in the non-statistical setting and it can be applied into an aggressive or lazy version. The aggressive version enjoys faster rates and it can be converted into a statistical multi-task learning method, while, the lazy method recovers standard rates, but it can be converted into a statistical meta-learning method, able to generalize across the tasks.

In the future it would valuable to investigate whether other multi-task learning methods based on different metrics and addressing different types of tasks' relatedness (e.g. those based on a shared low dimensional representation [12, 34] or graph regularization [7, 16]) can be made parameter-free as well. Moreover, it would be interesting to understand if our analysis allows more cycles over the data as in [15] and if this can be beneficial. Finally, we also wonder whether our parameter-free approach can be beneficial to recent meta-learning frameworks [8] dealing with partial feedback scenarios [2].

Acknowledgements

This work was supported in part by SAP SE and by EPSRC Grant N. EP/P009069/1.

References

[1] P. Alquier, T. T. Mai, and M. Pontil. Regret bounds for lifelong learning. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 261–269, 2017.

[2] J. Altschuler and K. Talwar. Online learning over a finite action set with limited switching. In *Conference On Learning Theory*, pages 1569–1573, 2018.

[3] A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[4] M.-F. Balcan, M. Khodak, and A. Talwalkar. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, pages 424–433, 2019.

[5] J. Baxter. A model of inductive bias learning. *J. Artif. Intell. Res.*, 12(149–198):3, 2000.

[6] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[7] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 11:2901–2934, 2010.

[8] L. Cella, A. Lazaric, and M. Pontil. Meta-learning with stochastic linear bandits. *arXiv preprint arXiv:2005.08531*, 2020.

[9] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

[10] A. Cutkosky and F. Orabona. Black-box reductions for parameter-free online learning in banach spaces. In *Conference On Learning Theory*, pages 1493–1529, 2018.

[11] G. Denevi, C. Ciliberto, R. Grazzi, and M. Pontil. Learning-to-learn stochastic gradient descent with biased regularization. In *International Conference on Machine Learning*, pages 1566–1575, 2019.

[12] G. Denevi, C. Ciliberto, D. Stamos, and M. Pontil. Incremental learning-to-learn with statistical guarantees. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, volume 34, pages 457–466. AUAI, 2018.

[13] G. Denevi, C. Ciliberto, D. Stamos, and M. Pontil. Learning to learn around a common mean. In *Advances in Neural Information Processing Systems*, pages 10190–10200, 2018.

[14] G. Denevi, D. Stamos, C. Ciliberto, and M. Pontil. Online-within-online meta-learning. In *Advances in Neural Information Processing Systems*, pages 13089–13099, 2019.

[15] H. Eichner, T. Koren, B. Mcmahan, N. Srebro, and K. Talwar. Semi-cyclic stochastic gradient descent. In *International Conference on Machine Learning*, pages 1764–1773, 2019.

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

[17] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine. Online meta-learning. In *International Conference on Machine Learning*, pages 1920–1930, 2019.

[18] E. Hazan. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3-4):157–325, 2016.

[19] M. Khodak, M.-F. F. Balcan, and A. S. Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, pages 5915–5926, 2019.

[20] P. J. Lenk, W. S. DeSarbo, P. E. Green, and M. R. Young. Hierarchical bayes conjoint analysis: Recovery of partworth heterogeneity from reduced experimental designs. *Marketing Science*, 15(2):173–191, 1996.

[21] N. Littlestone. From on-line to batch learning. In *Proceedings of the second annual workshop on Computational learning theory*, pages 269–284, 1989.

[22] A. Maurer. Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 6:967–994, 2005.

[23] A. Maurer. The rademacher complexity of linear transformation classes. In *International Conference on Computational Learning Theory*, pages 65–78, 2006.

[24] A. Maurer, M. Pontil, and B. Romera-Paredes. Sparse coding for multitask and transfer learning. In *International Conference on Machine Learning*, pages 343–351, 2013.

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

[26] A. M. McDonald, M. Pontil, and D. Stamos. New perspectives on k-support and cluster norms. *Journal of Machine Learning Research*, 17(155):1–38, 2016.

[27] F. Orabona. Simultaneous model selection and optimization through parameter-free stochastic learning. In *Advances in Neural Information Processing Systems*, pages 1116–1124, 2014.

[28] F. Orabona and D. Pál. Coin betting and parameter-free online learning. In *Advances in Neural Information Processing Systems*, pages 577–585, 2016.

[29] A. Pentina and C. Lampert. A PAC-Bayesian bound for lifelong learning. In *International Conference on Machine Learning*, pages 991–999, 2014.

[30] A. Pentina and R. Urner. Lifelong learning with weighted majority votes. In *Advances in Neural Information Processing Systems*, pages 3612–3620, 2016.

[31] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[32] M. Streeter and H. B. McMahan. No-regret algorithms for unconstrained online convex optimization. In *Proceedings of the 25th International Conference on Neural Information Processing Systems-Volume 2*, pages 2402–2410, 2012.

[33] S. Thrun and L. Pratt. *Learning to Learn*. Springer, 1998.

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