Incremental Learning-to-Learn with Statistical Guarantees

Giulia Denevi[1,2] **Carlo Ciliberto**[3] **Dimitris Stamos**[3] **Massimiliano Pontil** [1,3]
giulia.denevi@iit.it c.ciliberto@ucl.ac.uk d.stamos.12@ucl.ac.uk massimiliano.pontil@iit.it

[1] Computational Statistics and Machine Learning, Istituto Italiano di Tecnologia, 16163 Genova, Italy
[2] Department of Mathematics, University of Genova, 16146 Genova, Italy
[3] Department of Computer Science, University College of London, WC1E 6BT, London, United Kingdom

Abstract

In learning-to-learn the goal is to infer a learning algorithm that works well on a class of tasks sampled from an unknown meta-distribution. In contrast to previous work on batch learning-to-learn, we consider a scenario where tasks are presented sequentially and the algorithm needs to adapt incrementally to improve its performance on future tasks. Key to this setting is for the algorithm to rapidly incorporate new observations into the model as they arrive, without keeping them in memory. We focus on the case where the underlying algorithm is Ridge Regression parametrised by a symmetric positive semidefinite matrix. We propose to learn this matrix by applying a stochastic strategy to minimize the empirical error incurred by Ridge Regression on future tasks sampled from the meta-distribution. We study the statistical properties of the proposed algorithm and prove non-asymptotic bounds on its excess transfer risk, that is, the generalization performance on new tasks from the same meta-distribution. We compare our online learning-to-learn approach with a state-of-the-art batch method, both theoretically and empirically.

1 INTRODUCTION

Learning-to-learn (LTL) or meta-learning aims at finding an algorithm that is best suited to address a class of learning problems (tasks). These tasks are sampled from an unknown meta-distribution and are only partially observed via a finite collection of training examples, see (Baxter, 2000; Maurer, 2005; Thrun & Pratt, 1998) and references therein. This problem plays a large role in artificial intelligence in that it can improve the efficiency of learning from human supervision. In particular, substantial improvement over "learning in isolation" (also known as independent task learning, ITL) is to be expected when the sample size per task is small, a setting which naturally arises in many applications, see e.g. (Camoriano et al., 2017; Rebuffi et al., 2017; Rohrbach et al., 2013).

LTL is particularly appealing when considered from an online or incremental perspective. In this setting, which is sometimes referred to as lifelong learning, see e.g. (Ruvolo & Eaton, 2013), the tasks are observed sequentially – via corresponding sets of training examples – from a common environment and we aim to improve the learning ability of the underlying algorithm on future yet-to-be-seen tasks from the same environment. Practical scenarios of lifelong learning are wide ranging, including computer vision (Rebuffi et al., 2017), robotics (Camoriano et al., 2017), user modelling and many more.

Although LTL is naturally suited for the incremental setting, surprisingly, theoretical investigations are lacking. Previous studies, starting from the seminal paper (Baxter, 2000) and (Maurer, 2009; Maurer et al., 2013; 2016; Pentina & Lampert, 2014), have almost exclusively considered the setting in which the tasks are given in one batch, that is, the meta-algorithm processes multiple datasets from the environment jointly and only once as opposed to sequentially and indefinitely.

The papers (Balcan et al., 2015; Herbster et al., 2016) present results in an online framework which applies to a finite number of tasks using different performance measures. Perhaps most related to our work is (Alquier et al., 2017), where the authors consider a general PAC-Bayesian approach to lifelong learning based on the exponentially weighted aggregation procedure. Unfortunately, this approach is not efficient for large scale applications as it entails storing the entire sequence of datasets during the meta-learning process.

LTL also bears strong similarity to multi-task learning

(MTL), see e.g. (Caruana, 1997), and much work has been done on the theoretical study of both batch (Ando & Zhang, 2005; Maurer et al., 2013) and online (Cavallanti et al., 2010) multi-task learning algorithms. However multi-task learning aims to solve the different problem of learning well on a prescribed set of tasks (the learned model is tested on the same tasks used during training) whereas LTL aims to extrapolate to new tasks.

The principal contribution of this paper is to propose an incremental approach to learning-to-learn and to analyse its statistical guarantees. This incremental approach is appealing in that it efficiently processes one dataset at the time, without the need to store previously encountered datasets. We study in detail the case of linear representation learning, in which an underlying learning algorithm receives in input a sequence of datasets and incrementally updates the data representation so as to better learn future tasks. Following previous work on LTL, e.g. (Baxter, 2000; Maurer, 2009), we measure the performance of the incremental meta-algorithm by the *transfer risk*, namely the average error obtained by running the underlying algorithm with the learned representation, over tasks sampled from the meta-distribution.

Specifically, in this work we choose the underlying algorithm to be Ridge Regression parametrised by a symmetric positive semidefinite matrix. The incremental LTL approach we propose aims at optimizing the *future empirical error* (Maurer, 2009; Maurer et al., 2016) incurred by Ridge Regression over a class of linear representations. For this purpose, we propose to apply Projected Stochastic Subgradient Algorithm (PSSA). We show that the objective function of the resulting meta-algorithm is convex and we give a non-asymptotic convergence rate for the algorithm in high probability. A remarkable feature of our learning bound is that it is comparable to previous bounds for batch LTL. Our proof technique leverages previous work on learning-to-learn (Maurer, 2009) with tools from online convex optimization, see (Cesa-Bianchi et al., 2004; Hazan, 2016) and references therein.

The paper is organized as follows. In Sec. 2, we review the LTL problem and describe in detail the case of linear feature learning with Ridge Regression. In Sec. 3, we present our incremental meta-algorithm for linear feature learning. Sec. 4 contains our bound on the excess transfer risk for the proposed algorithm and in Sec. 5 we compare the bound to a previous bound for the batch setting. In Sec. 6, we report preliminary numerical experiments for the proposed algorithm and, finally, Sec. 7 summarizes the paper and highlight directions of future research. The detailed proofs of the statements in the paper are reported in the appendix.

2 PROBLEM FORMULATION

In the standard independent task learning setting the goal is to learn a functional relation between an input space $\mathcal{X}$ and an output space $\mathcal{Y}$ from a finite number of training examples. More precisely, given a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ measuring prediction errors and given a distribution μ on the joint data space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, the goal is to find a function $f : \mathcal{X} \to \mathcal{Y}$ minimizing the *expected risk*

$$\mathcal{R}_\mu(f) = \mathbb{E}_{z\sim\mu}\, \ell(f, z) \tag{1}$$

where, with some abuse of notation, for any $z = (x, y) \in \mathcal{Z}$ we denoted $\ell(f, z) = \ell(f(x), y)$. In most practical situations the underlying distribution is *unknown* and the learner is only provided with a finite set $Z = (z_i)_{i=1}^n \in \mathcal{Z}^n$ of observations independently sampled from μ. The goal of a learning algorithm is therefore, given such a *training* dataset Z to return a "good" estimator $A(Z) = f_Z$ whose expected risk is small and tends to the minimum of Eq. (1) as n increases.

A well-established approach to tackle the learning problem is offered by *regularized empirical risk minimization*. This corresponds to the family of algorithms A_ϕ such that, for any $Z \in \mathcal{Z}^n$,

$$A_\phi(Z) = \operatorname*{argmin}_{f\in\mathcal{F}_\phi} \mathcal{R}_Z(f) + \lambda\|f\|^2_{\mathcal{F}_\phi} \tag{2}$$

where $\phi : \mathcal{X} \to \mathcal{F}_\phi$ is a feature map, $\mathcal{F}_\phi$ is the Hilbert space of functions $f : \mathcal{X} \to \mathcal{Y}$ such that $f(x) = \langle f, \phi(x)\rangle_{\mathcal{F}_\phi}$ for any $x \in \mathcal{X}$ and

$$\mathcal{R}_Z(f) = \frac{1}{n}\sum_{i=1}^n \ell(f, z_i)$$

denotes the *empirical risk* of function f on the set Z.

2.1 LINEAR FEATURE LEARNING

In this work we will focus on the case that $\mathcal{Y} \subseteq \mathbb{R}$, $\mathcal{X} \subseteq \mathbb{R}^d$, ℓ is the square loss and $\phi : \mathbb{R}^d \to \mathbb{R}^m$ is a *linear* feature map (also known as a representation), corresponding to the action $\phi(x) = \Phi x$ of a matrix $\Phi \in \mathbb{R}^{m\times d}$ on the input space. It is well known, see e.g. (Argyriou et al., 2008), that, setting $D = \frac{1}{\lambda}\Phi^\top\Phi \in \mathbb{R}^{d\times d}$, any problem of the form in Eq. (2) can be equivalently formulated as

$$A_D(Z) = \operatorname*{argmin}_{w\in\operatorname{Ran}(D)} \mathcal{R}_Z(w) + w^\top D^\dagger w \tag{3}$$

where, with some abuse of notation, we denoted with $\mathcal{R}_Z(w)$ the empirical risk of the linear function $x \mapsto w^\top x$, for any $x \in \mathcal{X}$. Here, $D^\dagger$ denotes the pseudoinverse of D, which is symmetric positive semidefinite (PSD) but not necessarily invertible; when it is not invertible the constraint requiring w to be in the range $\mathrm{Ran}(D) \subseteq \mathbb{R}^d$ of D is needed to grant the equivalence with Eq. (2). Since for any linear feature map ϕ there exists a symmetric PSD matrix D such that Eq. (2) and Eq. (3) are equivalent, in the following we will refer to D as the *representation* used by algorithm A_D.

2.2 LEARNING TO LEARN D

A natural question is how to choose a good representation D for a given family of related learning problems. In this work we consider the approach of *learning* it from data. In particular, following the seminal work of (Baxter, 2000), we consider a setting where we are provided with an increasing number of tasks and our goal is to find a joint representation D such that the corresponding algorithm A_D is suited to address all such learning problems. The underlying assumption is that all the tasks that we observe *share a common structure* that algorithm A_D can leverage in order to achieve better prediction performance.

More formally, we assume that the tasks we observe are independently sampled from a meta-distribution ρ on the set of probability measures on $\mathcal{Z}$. According to the literature on the topic, see e.g. (Baxter, 2000; Maurer, 2005), we refer to the meta-distribution ρ as the *environment* and we identify each task sampled from ρ by its corresponding distribution μ, from which we are provided with a *training* dataset $Z \sim \mu^n$ of n points sampled independently from μ. While it is possible to consider a more general setting, for simplicity in this work we study the case where for each task we sample the same fixed number n of training points. In line with the independent task learning setting, the goal of a "learning-to-learn" algorithm is therefore to find the best parameter D minimizing the so-called *transfer risk*

$$\mathcal{E}(D) = \mathbb{E}_{\mu\sim\rho}\mathbb{E}_{Z\sim\mu^n}\ \mathcal{R}_\mu\big(A_D(Z)\big) \tag{4}$$

over a set $\mathfrak{D}$ of candidate representations. The term $\mathcal{E}(D)$ is the expected risk that the corresponding algorithm A_D, when trained on the dataset Z, would incur *on average with respect to the distribution of tasks μ induced by ρ*. That is, to compute the transfer risk, we first draw a task $\mu \sim \rho$ and a corresponding n-sample $Z \in \mathcal{Z}^n$ from μ^n, we then apply the learning algorithm to obtain an estimator $A_D(Z)$ and finally we measure the risk of this estimator on the distribution μ.

The problem of minimizing the transfer risk in Eq. (4) given a finite number T of training datasets $Z_1, \dots, Z_T$ sampled from the corresponding tasks $\mu_1, \dots, \mu_T$, has been subject of thorough analysis in literature, see e.g. (Baxter, 2000; Maurer, 2005; Maurer et al., 2016). Most work has been focused on the so-called "batch" setting, where all such training datasets are provided at once. However, by its nature, LTL is an ongoing (possibly never ending) process, with training datasets observed a few at the time. In such a scenario the meta-algorithm should allow for an evolving representation D, which improves over time as new datasets are observed. In the following we propose a meta-algorithm to learn D *online* with respect to the tasks, allowing us to transfer past experience about the environment in an efficient manner, *without requiring the memorization of training data*, which could be prohibitive in large scale applications. We will study the statistical guarantees of the proposed algorithm and compare it to its batch counterpart in terms of both theoretical and empirical performance.

2.3 CONNECTION WITH MULTI-TASK LEARNING

LTL is strongly related to *multi-task learning* (MTL) and in fact, as we will see later for the algorithm in Eq. (3), approaches developed for MTL can be used as inspiration to design algorithms for LTL. In multi-task learning a fixed number of tasks $\mu_1, \dots, \mu_T$ is provided up front and, given T datasets $Z_1, \dots, Z_T$, each sampled from its corresponding distribution, the goal is to find a joint representation D incurring a small *average expected risk* $\frac{1}{T}\sum_{t=1}^T \mathcal{R}_{\mu_t}(A_D(Z_t))$. In this sense, the main difference between LTL and MTL is that the former aims to guarantee good prediction performance on *future tasks*, while the latter aims to guarantee good prediction performance on the same tasks used to train D.

A well-established approach to MTL is *multi-task feature learning* (Argyriou et al., 2008). This method consists in solving the optimization problem

$$\min_{D\in\mathfrak{D}_\lambda} \frac{1}{T}\sum_{t=1}^T \min_{w\in\mathrm{Ran}(D)} \mathcal{R}_{Z_t}(w_t) + w_t^\top D^\dagger w_t$$

over the set

$$\mathfrak{D}_\lambda = \big\{D \in \mathbb{S}_+^d \mid \mathrm{tr(D)} \le 1/\lambda\big\} \tag{5}$$

where $\mathbb{S}_+^d$ denotes the set of $d \times d$ symmetric PSD matrices, $\mathrm{tr}(D)$ is the trace of D and λ is a positive parameter which controls the degree of regularization. In the subsequent analysis the parameter λ must be intended as a fixed hyper-parameter, which will be chosen by cross-validation in the experiments. This choice for $\mathfrak{D}_\lambda$ is motivated by the following variational form, see e.g. (Argyriou et al., 2008, Prop. 4.2), of the square trace norm of $W = [w_1, \dots, w_T] \in \mathbb{R}^{d\times T}$

$$\|W\|_1^2 = \frac{1}{\lambda} \inf_{D\in\mathrm{Int}(\mathfrak{D}_\lambda)} \sum_{t=1}^T w_t^\top D^{-1} w_t$$

where $\mathrm{Int}(\mathfrak{D}_\lambda)$ is the interior of $\mathfrak{D}_\lambda$, namely the set of the symmetric PSD invertible matrices with trace strictly smaller than $1/\lambda$. This leads to the equivalent problem

$$\min_{W \in \mathbb{R}^{d \times T}} \frac{1}{T} \sum_{t=1}^{T} \mathcal{R}_{Z_t}(w_t) + \gamma \|W\|_1^2 \qquad (6)$$

with $\gamma = \lambda/T$. The trace norm of a matrix is defined as the sum (ℓ_1-norm) of its singular values, and it is known to induce low-rank solutions for Problem (6). Intuitively, this means that tasks are encouraged to *share a common set of features (or representation)*. In this paper, we adopt this perspective to design our online LTL approach for linear feature learning.

3 ONLINE LEARNING-TO-LEARN

Motivated by the above connection with multi-task learning, we propose an online LTL approach to approximate the solution of the learning problem

$$\min_{D \in \mathfrak{D}_\lambda} \mathcal{E}(D)$$

over the set $\mathfrak{D}_\lambda$ introduced in Eq. (5). We consider the setting in which we are provided with a stream of independent datasets $Z_1, \dots, Z_T, \dots$, each sampled from an individual task distribution $\mu_1, \dots, \mu_T, \dots$ coming from the environment ρ and our goal is to find an estimator in $\mathfrak{D}_\lambda$ that improves *incrementally* as the number of observed tasks T increases.

3.1 MINIMIZING THE EMPIRICAL TRANSFER RISK

A key observation motivating the online procedure proposed in this work, is that in the *independent task learning* setting, standard results from learning theory, see e.g. (Shalev-Shwartz & Ben-David, 2014), allow one to control the statistical performance of regularized empirical risk minimization, providing bounds on the *generalization error* of A_D as

$$\mathbb{E}_{Z \sim \mu^n} |\mathcal{R}_\mu\big(A_D(Z)\big) - \mathcal{R}_Z\big(A_D(Z)\big)| \le G(D, n) \quad (7)$$

where $G(\cdot, n)$ is a decreasing function converging to 0 as $n \to +\infty$, while $G(D, \cdot)$ is a measure of complexity of D, which is large for more "expressive" representations and smaller otherwise.

Eq. (7) suggests us to use the empirical risk $\mathcal{R}_Z$ as a proxy for the expected risk $\mathcal{R}_\mu$. Therefore, we introduce the so-called *future empirical risk* (Maurer, 2009; Maurer et al., 2016),

$$\hat{\mathcal{E}}(D) = \mathbb{E}_{\mu \sim \rho} \mathbb{E}_{Z \sim \mu^n} \mathcal{R}_Z\big(A_D(Z)\big)$$

Algorithm 1 PSSA applied to $\hat{\mathcal{E}}$

```
Input: $T$ number of tasks, $\lambda > 0$ hyper-parameter, $\{\gamma_t\}_{t \in \mathbb{N}}$ step sizes.
Initialization: $D^{(1)} \in \mathfrak{D}_\lambda$
For $t = 1$ to $T$:
    Sample   $\mu_t \sim \rho$, $Z_t \sim \mu_t^n$.
    Choose   $U_t \in \partial \mathcal{L}_{Z_t}(D^{(t)})$
    Update   $D^{(t+1)} = \mathrm{proj}_{\mathfrak{D}_\lambda}(D^{(t)} - \gamma_t U_t)$
Return $\bar{D}_T = \frac{1}{T} \sum_{t=1}^{T} D^{(t)}$
```

and consider the related problem

$$\min_{D \in \mathfrak{D}_\lambda} \hat{\mathcal{E}}(D), \qquad (8)$$

which in the sequel, introducing the shorthand notation $\mathcal{L}_Z(D) = \mathcal{R}_Z(A_D(Z))$ for any $D \in \mathbb{S}_+^d$, will be rewritten as

$$\min_{D \in \mathfrak{D}_\lambda} \mathbb{E}_{\mu \sim \rho} \mathbb{E}_{Z \sim \mu^n} \mathcal{L}_Z(D) \qquad (9)$$

to highlight the dependency on Z.

Problem (9) can be approached with stochastic optimization strategies. Such methods proceed by sequentially sampling a point (dataset in this case) Z and performing an update step. In recent years, stochastic optimization, finding its origin in the Stochastic Approximation method by (Robbins & Monro, 1951), has been effectively used to deal with large scale applications. We refer to (Nemirovski et al., 2009) for a more comprehensive discussion about this topic. We therefore propose to apply Projected Stochastic Subgradient Algorithm (PSSA) (Shamir & Zhang, 2013), to solve the optimization problem in Eq. (9). The candidate representation coincides in this case with the mean after T iterations $\bar{D}_T$ and it is known as Polyak-Ruppert averaging scheme (Nemirovskii & Yudin, 1985; Polyak & Juditsky, 1992) in the optimization literature. Alg. 1 reports the application of PSSA to $\hat{\mathcal{E}}$ when $\mathcal{L}_Z$ is convex on the set $\mathbb{S}_+^d$. It requires iteratively: i) sampling a dataset Z, ii) performing a step in the direction of a subgradient of $\mathcal{L}_Z$ at the current point, and iii) projecting onto the set $\mathfrak{D}_\lambda$ (which can be done in a finite number of iterations, see Lemma 16 in App. E). Note that in this case, since the function $\mathcal{L}_Z$ is convex, there is no ambiguity in the definition of the subdifferential $\partial \mathcal{L}_Z$, see e.g. (Bertsekas et al., 2003), and we can rely on the convergence of Alg. 1 to a global minimum of $\hat{\mathcal{E}}$ over $\mathfrak{D}_\lambda$ for a suitable choice of step-sizes, as discussed in Sec. 4.

3.2 LTL WITH RIDGE REGRESSION

In this work, we focus on the case that the loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ corresponds to the square loss, namely $\ell(y, y') = (y - y')^2$ for any $y, y' \in \mathcal{Y} \subseteq \mathbb{R}$. In this setting, given a dataset $Z \in \mathcal{Z}^n$, algorithm A_D is equivalent to perform the following variant to *Ridge Regression*

$$\min_{w \in \mathrm{Ran}(D)} \frac{1}{n}\|\mathbf{y} - Xw\|^2 + w^\top D^\dagger w \tag{10}$$

where $X \in \mathbb{R}^{n\times d}$ is the matrix with rows corresponding to the input points $x_i \in \mathbb{R}^d$ in the dataset Z and $\mathbf{y} \in \mathbb{R}^n$ the vector with entries equal to the corresponding output points $y_i \in \mathbb{R}$. The solution to Eq. (10) can be obtained in closed form, in particular, see e.g. (Argyriou et al., 2008; Maurer, 2009),

$$A_D(Z) = DX^\top\big(XDX^\top + nI\big)^{-1}\mathbf{y}. \tag{11}$$

Plugging this solution in the definition of $\mathcal{L}_Z(D)$, a direct computation yields that

$$\mathcal{L}_Z(D) = n\big\|(XDX^\top + nI)^{-1}\mathbf{y}\big\|^{\,2}. \tag{12}$$

The following result characterizes some key properties of the function $\mathcal{L}_Z$ in Eq. (12), which will be useful in our subsequent analysis. We denote by $\mathcal{B}_r \subseteq \mathbb{R}^d$ the ball of radius $r > 0$ centered at 0.

Proposition 1 (Properties of $\mathcal{L}_Z$ for the Square Loss)**.** *Let $\mathcal{X} \subseteq \mathcal{B}_1$, $\mathcal{Y} \subseteq [0,1]$ and ℓ be the square loss. Then, for any dataset $Z \in \mathcal{Z}^n$ the following properties hold:*

1. *$\mathcal{L}_Z$ is convex on the set $\mathbb{S}^d_+$.*
2. *$\mathcal{L}_Z$ is $\mathcal{C}^\infty$ and, for every $D \in \mathbb{S}^d_+$,*

$$\nabla\mathcal{L}_Z(D) = -nX^\top M(D)^{-1}S(D)M(D)^{-1}X$$

where

$$M(D) = XDX^\top + nI$$
$$S(D) = \mathbf{y}\mathbf{y}^\top M(D)^{-1} + M(D)^{-1}\mathbf{y}\mathbf{y}^\top.$$

3. *$\mathcal{L}_Z$ is 2-Lipschitz w.r.t. the Frobenius norm.*
4. *$\nabla\mathcal{L}_Z$ is 6-Lipschitz w.r.t. the Frobenius norm.*
5. *$\mathcal{L}_Z(D) \in [0,1]$, for any $D \in \mathbb{S}^d_+$.*

The proposition above establishes the convexity of Problem (8) for the case of the square loss. This fact is important in that it guarantees no ambiguity in applying Alg. 1 to our setting and moreover, since $\mathcal{L}_Z$ is differentiable, Alg. 1 becomes a *Projected Stochastic Gradient Algorithm.*

4 THEORETICAL ANALYSIS

In this section, we study the statistical properties of Alg. 1 for the case of the square loss. Below we report the main result of this work, which characterizes the non-asymptotic behavior of the estimator $\bar{D}_T$ produced by Alg. 1 with respect to a minimizer $D_* \in \mathrm{argmin}_{D\in\mathfrak{D}_\lambda}\mathcal{E}(D)$. To present our results we introduce the $d \times d$ matrix $C_\rho = \mathbb{E}_{\mu\sim\rho}\mathbb{E}_{(x,y)\sim\mu}[xx^\top]$ denoting the covariance of the input data, obtained by averaging over all input marginals sampled from ρ. We also denote with $\|C_\rho\|_\infty$ the operator norm of C_ρ, which corresponds to the largest eigen-value.

Theorem 2 (Online LTL Bound)**.** *Let $\mathcal{X} \subseteq \mathcal{B}_1$, $\mathcal{Y} \subseteq [0,1]$ and ℓ be the square loss. Let $\mu_1, \dots, \mu_T$ be independently sampled from ρ and Z_t sampled from μ_t^n for $t \in \{1, \dots, T\}$. Let $\bar{D}_T$ be the output of Alg. 1 with step sizes $\gamma_t = (\lambda\sqrt{2t})^{-1}$. Then, for any $\delta \in (0,1]$*

$$\mathcal{E}(\bar{D}_T) - \mathcal{E}(D_*) \le \frac{4\sqrt{2\pi}\|C_\rho\|_\infty^{1/2}}{\sqrt{n}}\,\frac{1+\sqrt{\lambda}}{\lambda} + \frac{4\sqrt{2}}{\lambda\sqrt{T}} + \sqrt{\frac{8\log(2/\delta)}{T}}$$

with probability at least $1-\delta$ with respect to the independent sampling of the tasks $\mu_t \sim \rho$ and training sets $Z_t \sim \mu_t^n$ for any $t \in \{1, \dots, T\}$.

In Sec. 5, we will compare Thm. 2 with the statistical bound available for a state-of-the-art LTL batch procedure. We will see that the statistical behaviour of these two approaches is essentially equivalent, with the online LTL approach being more appealing given the lower requirements in terms of both number of computations and memory. In the rest of this section we give a sketch of the proof for Thm. 2. Proofs of intermediate results are reported in the appendix.

4.1 ERROR DECOMPOSITION

The statistical analysis of Alg. 1 hinges upon the following decomposition for the excess transfer risk of the estimator $\bar{D}_T$:

$$\begin{aligned}
&\mathcal{E}(\bar{D}_T) - \mathcal{E}(D_*) && (13)\\
&\quad = \mathcal{E}(\bar{D}_T) \pm \hat{\mathcal{E}}(\bar{D}_T) \pm \hat{\mathcal{E}}(D_*) - \mathcal{E}(D_*)\\
&\quad \le 2\sup_{D\in\mathfrak{D}_\lambda}|\mathcal{E}(D) - \hat{\mathcal{E}}(D)| + \hat{\mathcal{E}}(\bar{D}_T) - \hat{\mathcal{E}}(D_*)\\
&\quad \le \underbrace{2\sup_{D\in\mathfrak{D}_\lambda}|\mathcal{E}(D) - \hat{\mathcal{E}}(D)|}_{\text{Uniform generalization error}} + \underbrace{\hat{\mathcal{E}}(\bar{D}_T) - \hat{\mathcal{E}}(\hat{D}_*)}_{\text{Excess future empirical risk}}
\end{aligned}$$

where the matrix $\hat{D}_*$ denotes a minimizer of the future transfer risk over $\mathfrak{D}_\lambda$, that is, $\hat{D}_* \in \mathrm{argmin}_{D\in\mathfrak{D}_\lambda}\hat{\mathcal{E}}(D)$.

Eq. (13) decomposes $\mathcal{E}(\bar{D}_T) - \mathcal{E}(D_*)$ in a *uniform generalization error*, implicitly encoding the complexity of the class of algorithms parametrised by D and an *excess future empirical risk*, measuring the discrepancy between the estimator $\bar{D}_T$ and the minimizer $\hat{D}_*$ of $\hat{\mathcal{E}}$. In the following we describe how to bound these two terms.

4.2 BOUNDING THE UNIFORM GENERALIZATION ERROR

Results providing generalization bounds for the class of regularized empirical risk minimization algorithms A_D considered in this work are well known. The following result, which is taken from (Maurer, 2009), leverages an explicit estimate of the generalization bound $G(D, n)$ introduced in Sec. 3.1 for independent task learning, see Eq. (7), to obtain a uniform bound over the class of algorithms parametrized by $\mathfrak{D}_\lambda$.

Proposition 3 (Uniform Generalization Error Bound)**.** *Let $\mathcal{X} \subseteq \mathcal{B}_1$, $\mathcal{Y} \subseteq [0,1]$ and let ℓ be the square loss, then*

$$\sup_{D \in \mathfrak{D}_\lambda} |\mathcal{E}(D) - \hat{\mathcal{E}}(D)| \leq \frac{2\sqrt{2\pi}\|C_\rho\|_\infty^{1/2}}{\sqrt{n}} \frac{1+\sqrt{\lambda}}{\lambda}.$$

For completeness, we report the proof of this proposition in App. B.3.

4.3 BOUNDING THE EXCESS FUTURE EMPIRICAL RISK

Providing bounds for the excess future empirical risk introduced in Eq. (13) consists in studying the convergence rates of Alg. 1 to the minimum of $\hat{\mathcal{E}}$ over $\mathfrak{D}_\lambda$ *in high probability with respect to the sample of T tasks μ_t from ρ and datasets Z_t from μ_t^n for any $t \in \{1, \dots, T\}$.*

To this end, we leverage classical results from the online learning literature (Hazan, 2016). In online learning, the performance of an online algorithm returning a sequence $\{D^{(t)}\}_{t=1}^T$ over T trials is measured in terms of its *regret*, which in the context of this work corresponds to

$$R_T = \frac{1}{T}\sum_{t=1}^T \mathcal{L}_{Z_t}(D^{(t)}) - \min_{D \in \mathfrak{D}_\lambda} \frac{1}{T}\sum_{t=1}^T \mathcal{L}_{Z_t}(D).$$

Differently from the statistical setting considered in this work, in the online setting no assumption is made about the data generation process of $Z_1, \dots, Z_T$, which could be even adversely generated. Therefore, an algorithm that is able to solve the online problem (i.e. if its regret vanishes as $T \to \infty$) can be also expected to solve the corresponding problem in the statistical setting. This is indeed the case for Alg. 1, for which the following lemma provides a non-asymptotic regret bound.

Lemma 4 (Regret Bound for Alg. 1)**.** *Let $\mathcal{X} \subseteq \mathcal{B}_1$, $\mathcal{Y} \subseteq [0,1]$ and ℓ be the square loss. Then the regret of Alg. 1 with step-sizes $\gamma_t = (\lambda\sqrt{2t})^{-1}$ is such that*

$$R_T \leq \frac{4\sqrt{2}}{\lambda\sqrt{T}}.$$

The above lemma is a corollary of Prop. 1 combined with classical results on regret bounds for Projected Online Subgradient Algorithm (Hazan, 2016). We refer the reader to App. D.1 for a more in-depth discussion and for a detailed proof.

In our setting, the datasets $Z_1, \dots, Z_T$ are assumed to be independently sampled from the underlying environment. Combining this assumption with the regret bound in Lemma 4, we can control the excess future empirical risk by means of so-called *online-to-batch conversion* results (Cesa-Bianchi et al., 2004; Hazan, 2016), leading to the following proposition.

Proposition 5 (Excess Future Empirical Risk Bound for Alg. 1)**.** *Let $\mathcal{X} \subseteq \mathcal{B}_1$, $\mathcal{Y} \subseteq [0,1]$ and let ℓ be the square loss. Let $\mu_1, \dots, \mu_T$ be independently sampled from ρ and Z_t sampled from μ_t^n for $t \in \{1, \dots, T\}$. Let $\bar{D}_T$ be the output of Alg. 1 with step sizes $\gamma_t = (\lambda\sqrt{2t})^{-1}$. Then, for any $\delta \in (0,1]$*

$$\hat{\mathcal{E}}(\bar{D}_T) - \hat{\mathcal{E}}(\hat{D}_*) \leq \frac{4\sqrt{2}}{\lambda\sqrt{T}} + \sqrt{\frac{8\log(2/\delta)}{T}}$$

with probability at least $1-\delta$ with respect to the independent sampling of the tasks $\mu_t \sim \rho$ and training sets $Z_t \sim \mu_t^n$ for any $t \in \{1, \dots, T\}$.

The result above follows by combining Prop. 1 with online-to-batch results, see e.g. (Hazan, 2016, Thm. 9.3) and (Cesa-Bianchi et al., 2004). In App. D.2 we provide the complete proof of this statement together with a more detailed discussion about this topic. At this point we are ready to give the proof of Thm. 2.

Proof of Thm. 2. The claim follows by combining Prop. 3 and Prop. 5 in the decomposition of the error $\mathcal{E}(\bar{D}_T) - \mathcal{E}(D_*)$ given in Eq. (13). ■

5 ONLINE LTL VERSUS BATCH LTL

In this section, we compare the statistical guarantees obtained for our online meta-algorithm with a state-of-the-art batch LTL method for linear feature learning. We also comment on the computational cost of both procedures.

5.1 STATISTICAL COMPARISON

Given a finite collection $\mathbf{Z} = \{Z_1, \dots, Z_T\}$ of datasets, a standard approach to approximate a minimizer of the future empirical risk $\hat{\mathcal{E}}$ is to take a representation $\hat{D}_T$ minimizing the multi-task empirical risk

$$\hat{\mathcal{E}}_{\mathbf{Z}}(D) = \frac{1}{T}\sum_{t=1}^{T}\mathcal{R}_{Z_t}(A_D(Z_t)) \tag{14}$$

over the set $\mathfrak{D}_\lambda$. Such a choice has been extensively studied in the LTL literature (Baxter, 2000; Maurer, 2009; Maurer et al., 2013; 2016). Here we report a result analogous to Thm. 2, characterizing the discrepancy between the transfer risks of $\hat{D}_T$ and D_*.

Theorem 6 (Batch LTL Bound). *Let $\mathcal{X} \subseteq \mathcal{B}_1$, $\mathcal{Y} \subseteq [0,1]$ and let ℓ be the square loss. Let tasks $\mu_1, \dots, \mu_T$ be independently sampled from ρ and Z_t sampled from μ_t^n for $t \in \{1, \dots, T\}$. Let $\hat{D}_T$ be a minimizer of the multi-task empirical risk in Eq. (14) over the set $\mathfrak{D}_\lambda$. Then, for any $\delta \in (0,1]$*

$$\mathcal{E}(\hat{D}_T) - \mathcal{E}(D_*) \leq \frac{4\sqrt{2\pi}\|C_\rho\|_\infty^{1/2}}{\sqrt{n}}\,\frac{1+\sqrt{\lambda}}{\lambda} + \frac{2\sqrt{2\pi}}{\lambda\sqrt{T}} + \sqrt{\frac{2\log(2/\delta)}{T}}$$

with probability at least $1-\delta$ with respect to the independent sampling of the tasks $\mu_t \sim \rho$ and training sets $Z_t \sim \mu_t^n$ for any $t \in \{1, \dots, T\}$.

The result above is obtained by further decomposing the error $\mathcal{E}(\hat{D}_T) - \mathcal{E}(D_*)$ as done in Eq. (13). In particular, since the multi-task empirical error provides an estimate for the future empirical risk, it is possible to control the overall error by further bounding the term $|\hat{\mathcal{E}}(D) - \hat{\mathcal{E}}_{\mathbf{Z}}(D)|$ uniformly with respect to $D \in \mathfrak{D}_\lambda$. This last result was originally presented in (Maurer, 2009); in App. C we report the complete analysis of such decomposition, leading to the bound in Thm. 6.

5.2 STATISTICAL CONSIDERATIONS

For a fixed value of λ, we can now compare the bounds on the excess transfer risk for the representations resulting from the application of the online procedure (see Thm. 2) and the batch one (see Thm. 6). Since the approximation error due to the choice of λ will be the same for both approaches, this comparison provides a first indication of their statistical behavior. However, it should be kept in mind that we are comparing upper bounds, hence our considerations are not conclusive and further analysis by means of lower bounds for both algorithms would be valuable.

Thm. 2 and Thm. 6 are both composed of three terms. The first term is exactly the same for both procedures and this is obvious looking at the decompositions used to deduce both results. This term can be interpreted as a within-task-estimation error, that depends on the number of points n used to train the underlying learning algorithm (in our case Ridge Regression with a linear feature map). This term, similarly to the MTL setting, highlights the advantage of exploiting the relatedness of the tasks in the learning process in comparison to independent task learning (ITL). Indeed, if the inputs are distributed on a high dimensional manifold, then $\|C_\rho\|_\infty \ll 1$, while upper bounds for ITL have a leading constant of 1. In particular, $\|C_\rho\|_\infty = 1/d$ if the marginal distributions of the tasks are uniform on the $d-1$ dimensional unit sphere; see (Maurer, 2009; Maurer et al., 2016) for a more detailed discussion about this point. The last term in the bounds expresses the dependency on the confidence parameter δ and it is again approximately the same for the batch and the online case. It follows that the main role in the comparison between the online and batch bounds is driven by the middle term, which expresses the dependency of the bound on the number of tasks T. This term originates in different ways: in the batch approach it is derived from the application of uniform bounds and it can be interpreted as an inter-task estimation error, while in the online approach, it plays the role of an optimization error. Despite the different derivations, we can ascertain from the explicit formula of the bounds that this term is approximately the same for both procedures. This is remarkable since it implies that the representation resulting from our online procedure enjoys the same statistical guarantees than the batch one, despite its more parsimonious memory and computational requirements.

5.3 COMPUTATIONAL CONSIDERATIONS

After discussing the theoretical comparison between the online and the batch LTL approach, in this section we point out some key aspects regarding the computational costs of both procedures.

Memory. The batch LTL estimator corresponds to a minimizer of the multi-task empirical risk in Eq. (14) over *all tasks observed so far*. The corresponding approach therefore requires storing in memory all training datasets as they arrive in order to perform the optimization. This is clearly not sustainable in the incremental setting, since tasks are observed sequentially and, possibly indefinitely, inevitably leading to a memory overflow. On the contrary, in line with stochastic methods, online LTL has a small memory footprint, since it requires to store only one dataset at the time, allowing to “forget” it as soon as one gradient step is performed.

Time. Online LTL is also advantageous in terms of the number of iterations performed whenever a new task is observed. Indeed, for every new task, online LTL performs *only one step* of gradient descent for a total of T steps after T tasks. On the contrary, batch LTL requires finding a minimizer for Eq. (14), which cannot be obtained in closed form but requires adopting an iterative method such as Projected Gradient Descent, see e.g. (Combettes & Wajs, 2005). These methods typically require k iterations to achieve an error of the order of $O(1/k)$ from the optimum (better rates are possible adopting accelerated schemes). However, since for any new task batch LTL needs to find a minimizer for the multi-task empirical error from scratch, this leads to a total of Tk iterations after T tasks. Noting that every such iteration requires to compute T gradients of $\mathcal{L}_Z$ in contrast to the single one of PSSA, this shows that online LTL requires much less operations. In the batch case, a "warm-restart" strategy can be adopted to initialize the Projected Gradient Descent with the representation learned during the previous step, however, as we empirically observed in Sec. 6, online LTL is still significantly faster than batch.

6 EXPERIMENTS

In this section, we report preliminary empirical evaluations of the online LTL strategy proposed in this work; the Python implementation of our algorithm is available at *https://github.com/dstamos*. In particular we compare our method with its batch (or offline) counterpart and independent task learning (ITL), i.e. standard Ridge Regression, which does not leverage any shared structure among the tasks.

In all experiments, we obtain the online and batch estimators $\bar{D}_{\lambda,T_{\rm tr}}$ and $\hat{D}_{\lambda,T_{\rm tr}}$ by learning them on a dataset $\mathbf{Z}_{\rm tr}$ of $T_{\rm tr}$ *training* tasks, each comprising n input-output pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. Below to simplify our notation we omit the subscript $T_{\rm tr}$ in these estimators. We perform this training for different values of $\lambda \in \{\lambda_1, \dots, \lambda_p\}$ and select the best estimator based on the prediction error measured on a separate set $\mathbf{Z}_{\rm va}$ of $T_{\rm va}$ *validation* tasks. Once such optimal λ value has been selected, we report the generalization performance of the corresponding estimator on a set $\mathbf{Z}_{\rm te}$ of $T_{\rm te}$ *test* tasks. Note that the tasks in the test and validation sets $\mathbf{Z}_{\rm te}$ and $\mathbf{Z}_{\rm va}$ are all provided with both a training and test datasets $Z, Z' \in \mathcal{Z}^n$. Indeed, in order to evaluate the performance of a representation D, we need to first train the corresponding algorithm A_D on Z, and then test its performance on Z' (sampled from the same distribution), by computing the empirical risk $\mathcal{R}_{Z'}(A_D(Z))$. For all methods considered in this setting, we perform parameter selection over $p = 30$ candidate values of λ over the range $[10^{-6}, 10^3]$ with logarithmic spacing. In the online setting the training datasets arrive one at the time, therefore model selection is performed *online*: the system keeps track of all candidate representation matrices $\bar{D}_{\lambda_1}, \dots, \bar{D}_{\lambda_m}$ and whenever a new training task is presented, these matrices are all updated by incorporating the corresponding new observations. The best representation is then returned at each iteration, based on its performance on the validation set $\mathbf{Z}_{\rm va}$. Finally, in the subsequent experiments, we set the step sizes of the online LTL method in Alg. 1 equal to $\gamma_t = c/\sqrt{t}$, for some constant $c > 0$ chosen by model selection. Moreover, we computed the batch LTL estimator by classical Projected Gradient Descent method up to convergence, within 10^{-6} relative descent of the objective function.

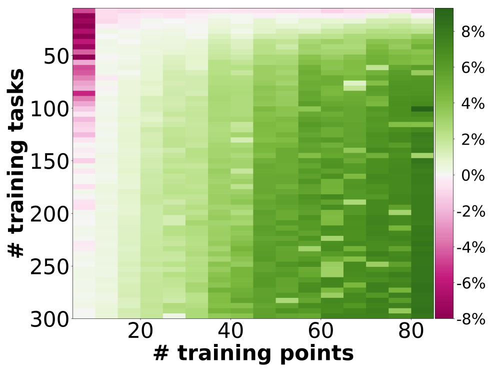

Figure 1: Relative improvement (in %) of our online LTL algorithm over the ITL baseline for a varying range of training tasks T_{tr} and number of samples n per task, during 30 trials.

Synthetic Data. We considered a regression problem on $\mathcal{X} \subseteq \mathbb{R}^d$ with $d = 50$ and a variable number of training tasks $T_{\rm tr}$ and training points n. We also generated $T_{\rm te} = 300$ test tasks and we sampled a number $T_{\rm va}$ of validation tasks equal to 50% of $T_{\rm tr}$. For each task, the corresponding dataset $(x_i, y_i)_{i=1}^n$ was generated according to the linear regression equation $y = w^\top x + \epsilon$, with x sampled uniformly on the unit sphere in $\mathbb{R}^d$ and ϵ sampled from a Normal distribution, $\epsilon \sim \mathcal{N}(0, 0.2)$. The tasks predictors w were generated as $P\tilde{w}$ with the components of $\tilde{w} \in \mathbb{R}^{d/2}$ sampled from $\mathcal{N}(0, 1)$ and then $\tilde{w}$ normalized to have unit norm, with $P \in \mathbb{R}^{d \times d/2}$ a matrix with orthonormal rows. In this way, the tasks reflect the assumption of sharing a low dimensional representation, which needs to be inferred by the LTL algorithm.

Fig. 1 reports the comparison between the baseline ITL and the proposed online LTL approach in terms of the relative difference of the prediction error on test tasks for the two methods. More precisely, given the mean squared errors (MSE) $R_{\rm oLTL}$ of online LTL and $R_{\rm ITL}$ of ITL averaged across the test tasks, we report the ratio $(R_{\rm ITL} - R_{\rm oLTL})/R_{\rm ITL}$ as a percentage improvement. Results are reported across a range of $T_{\rm tr}$ and n. We note that the regime considered for these experiments is par-

Table 1: Time (in seconds) for computing online and batch LTL for T_{tr} training tasks and n of samples per task.

T_{tr}	50		100		150	
n	20	50	20	50	20	50
Batch	85	227	246	617	428	2003
Online	36	86	108	273	227	776

ticularly favorable to LTL, almost always outperforming ITL. However, when the number of training points per task is small, the LTL algorithm, as expected, is unable to capture the underlying representation, unless several tasks are used in training.

To provide further evidence of the performance of online LTL, Fig. 2 (Top) compares the prediction error of online LTL, batch LTL, and ITL as the number of training tasks T_{tr} increases one at the time and the different methods update their corresponding representation accordingly. In this case, the number of samples per task is fixed to $n = 40$. We also added to the comparison the multi-task algorithm (MTL) described in Sec. 2.3, performing trace norm regularization *on the test set*. As expected, the performance of both ITL and MTL does not depend on the number of training tasks. Consistently to what observed before, ITL is outperformed by both LTL methods, which tend to converge to the MTL method as more training tasks are provided. In general, when, as in this case, the number of test tasks is large enough, the MTL method is expected to outperform LTL, since MTL optimizes the representation directly on the test tasks. Concerning the LTL methods, consistently with the theory presented in Sec. 4, the performance of the online method is equivalent to that of its batch counterpart, which is, as already stressed in Sec. 5.3, less appealing from the computational point of view. To confirm this aspect, we report in Tab. 1 the computational times required on average by online LTL and batch LTL as T_{tr} and n vary. Online LTL is faster than batch LTL.

Schools Dataset. We evaluated online LTL on the Schools dataset, consisting of examination records from 139 schools, see (Argyriou et al., 2008). Each school is associated to a regression task, individual students correspond to the input and their exam score to the output. In this case, the sample size n varies across the tasks and the features belong to an input space $\mathcal{X} \subseteq \mathbb{R}^d$, with $d = 26$. We randomly sampled 25% and 50% of the 139 tasks for LTL training and validation respectively and the remaining tasks were used as test set. Fig. 2 (Bottom) reports the performance of online LTL, batch LTL, ITL and MTL. Performance is reported in terms of the Explained Variance on the tasks (Argyriou et al., 2008), higher values correspond to better performance. Results are consistent with synthetic experiments; in particular, online

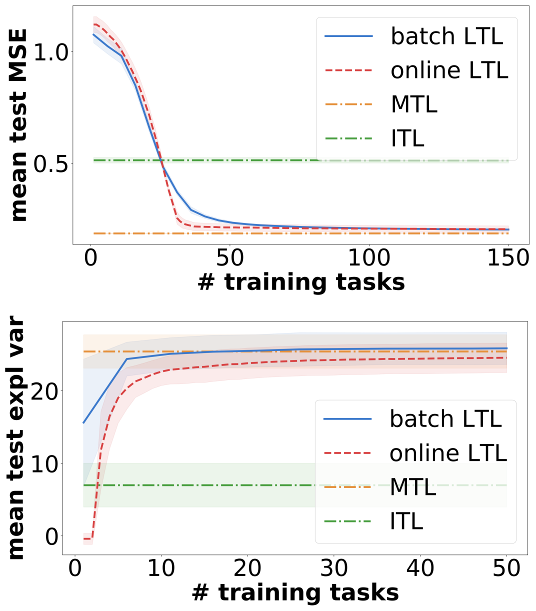

Figure 2: Performance of online LTL, batch LTL, ITL and MTL (on the test set) during 30 trials on the synthetic dataset (**Top**) and the Schools dataset (**Bottom**) as the number of training tasks increases incrementally.

and batch LTL are comparable.

7 CONCLUSION AND FUTURE WORK

We proposed an on-line (incremental) approach to LTL for linear data representation learning. Compared with its batch counterpart, this approach is computationally more efficient both in terms of memory and number of operations, while enjoying the same generalization properties. Preliminary experiments have highlighted the favorable learning capability of the proposed LTL strategy. Our analysis opens several future research directions. First, it would be valuable to investigate whether the same statistical guarantees hold for a projection-free meta-algorithm which does not require the computation of the entire SVD (e.g. certain variants of Frank Wolfe algorithm (Hazan & Kale, 2012), which do not require memorizing the sequence of datasets). Second, from a modeling perspective, we could take inspiration from the vast MTL literature to design new LTL methods in order to deal with tasks that are not necessarily spanning a low-rank subspace but are for instance organized into clusters (Jacob et al., 2009) or share a sparse set of relations (Ciliberto et al., 2015a;b). Finally, extending our analysis to non-convex settings would allow one to tackle more general families of learning algorithms as well as recent empirical meta-learning approaches (e.g. Franceschi et al., 2018) which implicitly attempt to directly minimize the transfer risk.

References

Alquier, Pierre, Mai, The Tien, and Pontil, Massimiliano. Regret bounds for lifelong learning. In *International Conference on Artificial Intelligence and Statistics*, 2017.

Ando, Rie Kubota and Zhang, Tong. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 2005.

Argyriou, Andreas, Evgeniou, Theodoros, and Pontil, Massimiliano. Convex multi-task feature learning. *Machine Learning*, 2008.

Balcan, Maria-Florina, Blum, Avrim, and Vempala, Santosh. Efficient representations for lifelong learning and autoencoding. In *Conference on Learning Theory*, 2015.

Bartlett, Peter L and Mendelson, Shahar. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3:463–482, 2002.

Bauschke, Heinz H, Combettes, Patrick L, et al. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer.

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

Bertsekas, Dimitri P, Nedi, Angelia, and Ozdaglar, Asuman. *Convex analysis and optimization*. Athena Scientific, 2003.

Bhatia, Rajendra. Matrix analysis, volume 169 of graduate texts in mathematics, 1997.

Boucheron, Stéphane, Lugosi, Gábor, and Bousquet, Olivier. Concentration inequalities. In *Advanced Lectures on Machine Learning*, pp. 208–240. Springer, 2004.

Camoriano, Raffaello, Pasquale, Giulia, Ciliberto, Carlo, Natale, Lorenzo, Rosasco, Lorenzo, and Metta, Giorgio. Incremental robot learning of new objects with fixed update time. In *ICRA*, 2017.

Caruana, Rich. Multitask learning. *Machine Learning*, 1997.

Cavallanti, Giovanni, Cesa-Bianchi, Nicolo, and Gentile, Claudio. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 2010.

Cesa-Bianchi, Nicolo, Conconi, Alex, and Gentile, Claudio. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 2004.

Ciliberto, Carlo, Mroueh, Youssef, Poggio, Tomaso, and Rosasco, Lorenzo. Convex learning of multiple tasks and their structure. In *ICML*, 2015a.

Ciliberto, Carlo, Rosasco, Lorenzo, and Villa, Silvia. Learning multiple visual tasks while discovering their structure. In *CVPR*, 2015b.

Combettes, Patrick L and Wajs, Valérie R. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

Franceschi, Luca, Frasconi, Paolo, Grazzi, Riccardo, Salzo, Saverio, and Pontil, Massi. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, 2018.

Grimmett, Geoffrey and Stirzaker, David. *Probability and random processes*. Oxford university press, 2001.

Hazan, Elad. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2016.

Hazan, Elad and Kale, Satyen. Projection-free online learning. *arXiv preprint arXiv:1206.4657*, 2012.

Herbster, Mark, Pasteris, Stephen, and Pontil, Massimiliano. Mistake bounds for binary matrix completion. In *Advances in Neural Information Processing Systems*, 2016.

Jacob, Laurent, Vert, Jean-philippe, and Bach, Francis R. Clustered multi-task learning: A convex formulation. In *Advances in neural information processing systems*, 2009.

Kollo, Tonu and von Rosen, Dietrich. *Advanced Multivariate Statistics with Matrices*, volume 579. Springer Science & Business Media, 2006.

Maurer, Andreas. Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 2005.

Maurer, Andreas. Transfer bounds for linear feature learning. *Machine learning*, 75(3):327–350, 2009.

Maurer, Andreas, Pontil, Massi, and Romera-Paredes, Bernardino. Sparse coding for multitask and transfer learning. In *ICML*, 2013.

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

McDonald, Andrew M, Pontil, Massimiliano, and Stamos, Dimitris. New perspectives on k-support and cluster norms. *Journal of Machine Learning Research*, 2016.

Nemirovski, Arkadi, Juditsky, Anatoli, Lan, Guanghui, and Shapiro, Alexander. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 2009.

Nemirovskii, A. and Yudin, D. B. Problem complexity and method efficiency in optimization. *SIAM Review*, 1985.

Pentina, Anastasia and Lampert, Christoph. A PAC-Bayesian bound for lifelong learning. In *International Conference on Machine Learning*, pp. 991–999, 2014.

Petersen, Kaare Brandt and Pedersen, Michael Syskind. The matrix cookbook. *Technical University of Denmark*, 2008.

Polyak, Boris T and Juditsky, Anatoli B. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Rebuffi, Sylvestre-Alvise, Kolesnikov, Alexander, and Lampert, Christoph H. iCaRL: Incremental classifier and representation learning. In *Proc. CVPR*, 2017.

Robbins, Herbert and Monro, Sutton. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951.

Rohrbach, Marcus, Ebert, Sandra, and Schiele, Bernt. Transfer learning in a transductive setting. In *Advances in Neural Information Processing Systems*, pp. 46–54, 2013.

Ruvolo, Paul and Eaton, Eric. Ella: An efficient lifelong learning algorithm. In *ICML*, 2013.

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

Shamir, Ohad and Zhang, Tong. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *ICML*, 2013.

Thrun, Sebastian and Pratt, Lorien. *Learning to Learn*. Springer, 1998.