Online Parameter-Free Fine Tuning for Heterogeneous Sequences of Tasks

Youning Xia[1]

MSc Computational Statistics and Machine Learning

Supervisor: Giulia Denevi
Dimitris Stamos
Massimiliano Pontil

Submission date: August 2020

Abstract

Knowledge transfer plays an essential part in increasing the capability of an intelligent system to perform tasks more proficiently by leveraging shared knowledge among multiple related tasks. Recently, to cater for the practical need of performing tasks in a continuous manner, knowledge-transfer methods under online setting have been actively studied. Yet popular online knowledge-transfer methods have proved fruitful results in their own way, a majority of them either require tuning hyper-parameters or guarantee performance only on homogeneous tasks. Such aspects could become bottlenecks in practice as hyper-parameter tuning is usually difficult to conduct in the online setting and an system does not always receive similar tasks to process. In this thesis, we develop an online parameter-free knowledge-transfer method using the fine tuning framework to learn a good conditioning function for a heterogeneous sequence. The proposed method does not require tuning hyper-parameters and is able to learn non-homogeneous tasks well. Our method is theoretically grounded and can be of two variants - the "aggressive" one and the "lazy" one. The "aggressive" one updates the conditioning function after each datapoint and the "lazy" one updates the conditioning function only at the end of each task. We present our methods primarily in the non-statistical setting but also show that they can be adapted to the statistical setting as well. More specifically, the "aggressive" variant can be converted into a statistical multi-task learning method and the "lazy" variant can be converted into a statistical meta-learning method. Empirical results justify the efficacy of our methods in practice.

Acknowledgements

First and foremost, I would like to express my greatest gratitude to my amazing advisors, Giulia Denevi and Dimitris Stamos for their support throughout the dissertation work. Both Giulia and Dimitris have guided me in developing the idea to work on and have always answered my doubts with great patience to execute the work along the way. Without their sincere help, I would have been nowhere in the dissertation. I also want to thank my PI Massimiliano Pontil who gave me this wonderful opportunity to work with them. Last but most importantly, I want to thank my parents and my younger sister for always being there for me.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Machine learning (ML) is a realm that studies how a system can perform well on tasks through learning. In the classic ML setup, learning is conducted with supervision individually for each task. For example, when given with data on personal preference about PC products of a student (task), a system can learn to predict this student's preference over a new PC product. However, learning a single task can sometimes require a large amount of data in order to perform well, which is not always practical due to limited computational resources and lack of high-quality labeled data in reality. Therefore, we need methods that could transfer knowledge among tasks to facilitate and enhance the learning procedure.

Such methods are generally studied under the umbrella of knowledge-transfer problems across tasks, which encapsulates *multi-task learning* (MTL) and *meta-learning*. Simply put, multi-task learning aims to improve the performance of some predefined tasks by exploiting their relatedness during learning. For example, in product recommendation, the system's goal is to predict product-likeness of a certain group of students (multiple tasks) more proficiently by learning them jointly than individually. On the other hand, meta-learning aims to perform well on a not-yet-seen new task by assimilating and utilizing knowledge learned from previous related tasks. For instance, in the same example, the system's goal is to learn product preference of a new student customer effectively by leveraging knowledge accumulated from previous student customers.

There are various frameworks that have been studied to tackle such knowledge transfer problems. Successful modern approaches range from bias-regularization [1, 7, 5] to gradient-based fine tuning [8, 14, 13]. Typically a multi-task learning or meta-learning method consists of a *transfer-learner* which is used to learn task relatedness and a *within-task learner* which is used to exploit such relatedness and make prediction. Whilst different approaches have their own designs of transfer-learners and within-task learners, they could be identified by the following two dichotomies. We remark that the categorizations here are made primarily for contextualising our work in this thesis and are not limited to the listed aspects.

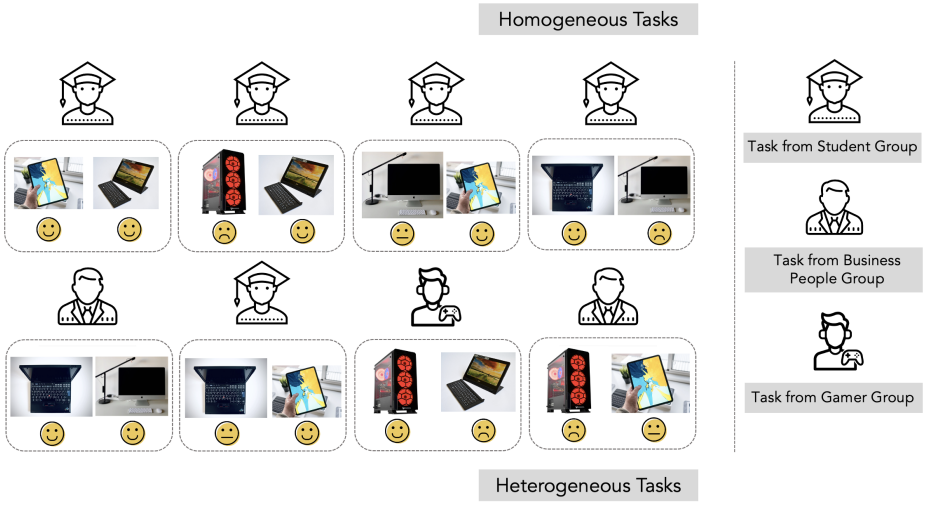

Figure 1.1: Illustrative comparison of homogeneous tasks and heterogeneous tasks. Each dot-line box represents the dataset (in this case, preference over PC products) partially observed from a task (in this case, customer). Each face icon represents the task. In the homogeneous sequence (top), four tasks are all derived from student group. In the heterogeneous sequence (bottom), two tasks are derived from business people group, one task is from student group and one task is from gamer group. The diversity in customers results in heterogeneity of tasks.

Homogeneous Tasks vs Heterogeneous Tasks The first distinction stems from the nature of the tasks that the method tackles. In the former, all tasks in hand are derived from a particular pool/group of tasks (in ML parlance, also known as environment). In the latter, tasks received are taken from more than one environment. For example, in the aforementioned product recommendation context, a sequence of homogeneous tasks could be different student customers whilst a heterogeneous sequence of tasks could be mixed customers ranging from students, business people to professional gamers. Figure 1.1 gives an illustrative comparison between the two settings.

Parameter-Free vs Hyper-Parameter Tuning The second distinction distinguishes between methods which do not require tuning any hyper-parameters and methods which require hyper-parameter tuning. As we will see in Chapter 2, tuning hyper-parameter optimally is particularly difficult in the online setting.

A majority of multi-task learning and meta-learning literature has been mainly focusing on methods for homogeneous tasks, which at the same time require hyper-parameter tuning (see for instance [8, 14, 13, 5, 6, 9]). It is only until recently that some research has developed effective knowledge-transfer methods that is parameter-free [10, 4] and techniques which deal with heterogeneity in tasks [19, 21, 20, 3].

1.2 Related Work & Contributions

We have seen that previous work on multi-task learning and meta-learning methods have been largely focused on homogeneous tasks and require tuning hyper-parameters. Whilst these approaches have proved to be successful in some aspects of their own practice, one crucial drawback could be that when used in real-world application, where tasks often arrive one after another instead of coming in one go, it could be expensive and impractical to identify the optimal hyper-parameter through validation process. Such case where data is available in a sequential order is referred to as *online* setting. Another shortage of these methods comes from the fact that knowledge transfer problems in real practice are complicated and often comprise diverse tasks instead of just similar tasks. Therefore performing well only on homogeneous tasks might not be applicable enough.

These bottlenecks have motivated the recent study of parameter-free online knowledge-transfer methods [10, 4] and conditioning techniques to handle heterogeneity among tasks [19, 20, 21, 3]. However, such work only partially deals with the aforementioned problems and do not provide comprehensive solutions. In this thesis, we aim to propose an online parameter-free method to learn a good conditioning function for heterogeneous tasks by combining the idea in [4, 3], using the framework of fine tuning methods [8]. Our method can be of two variants: the "aggressive" one and the "lazy" one. The "aggressive" one updates the conditioning function upon receiving a new datapoint and the "lazy" one updates the conditioning function only at the end of each task. We present the method primarily in the non-statistical setting, where no assumptions are made on the received data, and provide theoretical learning guarantees for the two variants. We also show that the two variants can be adapted to the statistical setting, where data is assumed to come from certain statistical distributions. More specifically, the "aggressive" variant becomes a multi-task learning method and the "lazy" variant becomes a meta-learning method. The empirical performance of our method is confirmed using synthetic data.

1.3 Outline

This thesis is organized as follows. In Chapter 2, we cover the background on fine tuning framework which requires tuning of hyper-parameters. We start by introducing the oracle hyper-parameter fine tuning method for learning homogeneous tasks and briefly argue how to find theoretically optimal hyper-parameter in the method (which we refer to as oracle hyper-parameter tuning) so as to show why it is difficult to tune the hyper-parameter in the non-statistical setting. Next, we present the conditioning technique and analyse its advantage in learning heterogeneous tasks. This approach leads the basis for the formulation of our proposed methods.

In Chapter 3, we present two well-established parameter-free online learning algorithms which are used in designing our methods and give their regret bound to illustrate how well these online algorithms

perform.

In Chapter 4, we present our proposed online parameter-free fine tuning method for heterogeneous tasks, demonstrating its "aggressive" variant and "lazy" variant. We then provide regret bound for the two variants to show the performance guarantees. After presenting the methods in the non-statistical setting, we briefly recall two paradigms of statistical setting - the statistical multi-task learning (MTL) and statistical meta-learning setting and analyse how our two variants can be adapted to these two statistical setups by giving their (statistical) performance guarantees accordingly.

In Chapter 5, we test the empirical performance of our methods using two types of synthetic data and show the results are in line with our theoretical findings. Finally, in Chapter 6 we draw our conclusions.

The proofs and supplementary materials omitted from the main body are postponed to the appendices.

Part I

Background

Chapter 2

Fine Tuning Framework

In this chapter, we briefly present the fine tuning framework on which the work in this thesis builds. The fine tuning formulation adopted in this chapter is an adaptation of [4]. We begin by introducing the oracle hyper-parameter fine tuning framework for homogeneous tasks and then extend the discussion to the conditional technique proposed in [3] which tackles heterogeneous tasks.

2.1 Preliminaries

First of all, we list some notations and remarks used throughout this work. We define each task on the data space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ where $\mathcal{X} \subseteq \mathbb{R}^d$ is the input space and $\mathcal{Y} \subseteq \mathbb{R}$ is the output space. For all methods introduced, we consider the Online-Within-Online (OWO) setting outlined in [6], where the transfer-learner incrementally receives datasets $Z_t = (z_{t,i})_{t=1,i=1}^{T,n} = (x_{t,i}, y_{t,i})_{t=1,i=1}^{T,n} \in \mathcal{Z}^n$ and the within-task learner is required to make a prediction upon a new observation $(x_{t,i}, y_{t,i})$. To simplify presentation, in this work we restrict the within-task learner to perform linear prediction of the form $\hat{y} = \langle x, w \rangle$, where $w \in \mathbb{R}^d$ is the weight vector and $\langle \cdot, \cdot \rangle$ denotes the standard inner product in $\mathbb{R}^d$. The performance of the prediction is measured by error $\ell(\cdot, y_i) = \ell_i(\langle x_i, w_i \rangle)$ where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a convex and closed loss function. We denote the set of subgradients of $\ell_i(\langle x_i, \cdot \rangle)$ at w_i by $\partial \ell_i(\langle x_i, \cdot \rangle)(w_i)$. We use $\|\cdot\|$ to denote the Euclidean norm and specifically use $\|\cdot\|_F$ to denote the Frobenius norm (Euclidean norm used for matrix). We also use $\langle \cdot, \cdot \rangle_F$ to denote the Frobenius inner product in $\mathbb{R}^{d \times k}$. Finally, in the presentation of the algorithm, we denote $\mathrm{Proj}_{\mathcal{B}}(\cdot)$ as the projection operation, mapping a vector into a set $\mathcal{B}$. For the rest of the work, we make the following assumption on the loss function and inputs.

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

$$\mathcal{B}(c, r) = \Big\{ v \in \mathbb{R}^d \ : \ \|v - c\| \leq r \Big\}. \tag{2.1}$$

2.2 Oracle Hyper-Parameter Fine Tuning for a Homogeneous Sequence

Fine tuning is a recently developed technique that transfers knowledge across a sequence of tasks through a common parameter. Particularly, the fine tuning framework adopts online gradient descent approach as the within-task algorithm and the common parameter (also called transfer-parameter in ML parlance) is the associated starting point. The goal of a fine tuning method is to learn an ideal initialization of the transfer-parameter for the within-task learner so that a few gradient steps on training datapoints from a new task by the within-task learner suffice to produce good task-specific predictions.

2.2.1 Within-Task Algorithm

In this section, we introduce and analyse the online gradient descent algorithm for a single task $Z = (x_i, y_i)_{i=1}^n$, which could be the simplest form of the gradient-based within-task learner in OWO fine tuning method. We present the algorithm in the non-statistical setting where no assumptions are made on the dataset Z. Unlike the standard formulation used in [8], here we employs a translated version taken from [4] in which the transfer-parameter θ acts as the starting point. The procedure is outlined in Algorithm 1.

Algorithm 1 Online Gradient Descent (translated version)

Input $\tilde{w}_1 = 0$, transfer-parameter θ, step sizes $\{\eta_i\}$, non-empty closed convex set $\mathcal{C}$
For $i = 1, \ldots, n$
  Predict $\langle x_i, w_i \rangle$ and observe loss $\ell_i(\langle x_i, w_i \rangle)$
  Update $\tilde{w}_{i+1} = \tilde{w}_i - \eta_i g_i$, $g_i \in \partial \ell_i(\langle x_i, \cdot \rangle)(w_i)$
  Translate $w_{i+1} = \tilde{w}_{i+1} + \theta$
  Project $w_{i+1} = \mathrm{Proj}_{\mathcal{C}}(w_{i+1})$
End
Return $(w_i)_{i=1}^n$

By setting $\tilde{w}_1 = \theta$, it can be shown that Algorithm 1 is equivalent to the standard (untranslated) online gradient descent algorithm with initial weight vector θ.

The performance of the algorithm is evaluated by the regret of its iterates $(w_i)_{i=1}^n$ generated over the task Z, i.e.

$$\sum_{i=1}^{n} \ell_i \left(\langle x_i, w_i \rangle \right) - \min_{w \in \mathbb{R}^d} \sum_{i=1}^{n} \ell_i \left(\langle x_i, w \rangle \right). \tag{2.2}$$

The first term of (2.2) indicates the cumulative error incurred by Algorithm 1 and the second term

indicates the attainable minimum cumulative error in hindsight, which is thus treated as the benchmark performance of the algorithm. Intuitively, for a 'good' algorithm, we expect the cumulative error to approach the benchmark error with the increase of datapoints, i.e. $n \to \infty$. Such behaviour can be analysed by providing a bound on the regret. In the following, we provide the upper bound on the regret (2.2).

Proposition 2.1 (Regret Bound for Algorithm 1, [**Theorem 2.13**, 15])**.** *Let Assumption 1 hold and let $\mathcal{C}$ be a non-empty closed convex set. Assume the learning rate is constant $\eta_i = \eta$, we have the following regret bound hold for any $w \in \mathcal{C}$,*

$$\begin{aligned}\sum_{i=1}^{n} \ell_i\left(\langle x_i, w_i\rangle\right) - \sum_{i=1}^{n} \ell_i\left(\langle x_i, w\rangle\right) &\leq \sum_{i=1}^{n} \langle g_i, w_i - w\rangle \\ &\leq \frac{\|w-\theta\|^2}{2\eta} + \frac{\eta}{2}\sum_{i=1}^{n}\|g_t\|^2 \end{aligned} \tag{2.3}$$

As can be seen from (2.3), the exact regret bound of Algorithm 1 depends on the choice of hyper-parameter η and hence we need to tune it properly so as to obtain optimal performance of the algorithm. A simple attempt is to set the learning rate η as the one that minimizes the regret bound for a fixed number of iterations, a technique called *oracle tuning*. It can be easily shown that such optimal η is $\frac{\|w-\theta\|}{\sqrt{\sum_{i=1}^{n}\|g_i\|^2}}$ by arithmetic. Regarding the norm of the subgradients $(g_i)_{i=1}^n$ in this quantity, following the L-Lipschitzness of loss ℓ_i and bounded inputs assumption in Assumption 1, g_i can be bounded by RL, i.e. $\|g_i\| \leq RL$ for $i \in \{1, \ldots, n\}$. Using this fact, we can further reduce the bound to

$$\|w-\theta\|\, RL\sqrt{n}. \tag{2.4}$$

From (2.4) we see that the regret grows sublineraly with the number of datapoints, implying that the incurred cumulative errors is able to approach the optimal benchmark in the long run. However, the oracle tuning of learning rates requires knowledge of all future gradients $(g_i)_{i=1}^n$ and the distance between the optimal weight vector and the initial point, i.e. $\|w-\theta\|$, in hindsight. This is impossible in the online non-statistical setting because the inputs given by an invisible adversary can always been adapted to the learner's behaviour. Such shortage of oracle hyper-parameter fine tuning motivates the study of parameter-free fine tuning method in [4], which forms the fundamental concepts of our proposed methods in this work. We will not go into this method here and refer the readers to [4] for more details.

2.2.2 The advantage of using the right starting point

In the previous section, we present Algorithm 1 using an arbitrary fixed transfer-parameter θ to learn a single task. In this section, we study the advantage of using an appropriate starting transfer-parameter in Algorithm 1 when learning multiple tasks. Specifically, we analyse the performance of applying Algorithm 1 with the same starting point θ to a sequence of homogeneous tasks $\mathbf{Z} = (Z_t)_{t=1}^{T} = (z_{t,i})_{t=1,i=1}^{T,n} = (x_{t,i}, y_{t,i})_{t=1,i=1}^{T,n}$. The performance of this procedure is measured by the regret accumulated across tasks, i.e

$$\sum_{t=1}^{T} \left(\sum_{i=1}^{n} \ell_{t,i} \left(\langle x_{t,i}, w_{t,i} \rangle \right) - \min_{w_t \in \mathbb{R}^d} \sum_{i=1}^{n} \ell_{t,i} \left(\langle x_{t,i}, w_t \rangle \right) \right). \tag{2.5}$$

The first term in (2.5) indicates the across-tasks error incurred by Algorithm 1 and the second term indicates the minimum across-tasks error in hindsight, where we assume the minimum targets w_t w.r.t. each task are attained. By construction, the target weight vectors for each task is expressed as

$$w_t = \tilde{w}_t + \theta. \tag{2.6}$$

As a direct adaptation from Proposition 2.1, we get the across-tasks regret bound for Algorithm 1 over the tasks as follows.

Proposition 2.2 (Adaptation of [4])**.** *For any task $t = 1, \ldots, T$, let $(w_{t,i})_{i=1}^{n}$ be the iterates generated by Algorithm 1 over the dataset Z_t with starting point θ. Let Assumption 1 hold and let the learning rate η be the oracle hyper-parameter, then for any sequence $(w_t)_{t=1}^{T}, w_t \in \mathcal{C}$,*

$$\begin{aligned} \sum_{t=1}^{T} \sum_{i=1}^{n} \ell_{t,i} \left(\langle x_{t,i}, w_{t,i} \rangle \right) - \ell_{t,i} \left(\langle x_{t,i}, w_t \rangle \right) &\leq \sum_{t=1}^{T} \sum_{i=1}^{n} \langle g_{t,i}, w_{t,i} - w_t \rangle \\ &\leq TRL \operatorname{Var}(\theta) \sqrt{n} \end{aligned} \tag{2.7}$$

where

$$\operatorname{Var}(\theta) = \frac{1}{T} \sum_{t=1}^{T} \left\| w_t - \theta \right\|. \tag{2.8}$$

Proof Sketch. The statement in 2.7 directly follows by summing the regret obtained in Proposition 2.1 over the datasets. □

Although quantity (2.8) is not strictly the variance of target vectors $(w_t)_{t=1}^{T}$ w.r.t. initialization θ,

with a slight abuse of notation, we write $\mathrm{Var}(\theta)$ for this quantity. In fact, by Jensen's inequality, we have $\frac{1}{T}\sum_{t=1}^{T}\|w_t-\theta\| \leq \sqrt{\frac{1}{T}\sum_{t=1}^{T}\|w_t-\theta\|^2}$ and therefore $\mathrm{Var}(\theta)$ is the lower bound of the variance.

Proposition 2.2 implies that the across-tasks regret bound in (2.7) grows in the order of $\mathcal{O}(\mathrm{Var}(\theta)\sqrt{n}T)$. Therefore, to achieve the minimal bound, the optimal choice of the starting point θ is the one that minimizes the variance of the target vectors $(w_t)_{t=1}^T$, namely

$$\hat{\theta} = \underset{\theta\in\mathbb{R}}{\operatorname{argmin}} \sqrt{\frac{1}{T}\sum_{t=1}^{T}\|w_t-\theta\|^2} = \frac{1}{T}\sum_{t=1}^{T} w_t \tag{2.9}$$

The analysis above suggests that the advantage of using a common optimal starting point in Algorithm 1 when learning multiple tasks over learning each task independently is more significant when the tasks have a low variance of target vector.

2.2.3 Estimating the starting point from data

Now that we have shown that a good initialization is vital in learning multiple low-variance tasks proficiently, a natural question is how we can set the initial point appropriately. To address this problem, we introduce the simple but successful oracle hyper-parameter fine tuning approach here. In this approach, a gradient-based transfer-learner is used to estimate a good starting point θ shared across multiple low-variance homogeneous tasks from a sequence of datasets $\mathbf{Z} = (Z_t)_{t=1}^T$, $Z_t = (x_{t,i}, y_{t,i})_{i=1}^n$. As mentioned in Chapter 1, there can be two variants of this approach: the "aggressive" variant where θ is updated after each datapoint, and the "lazy" variant where θ is updated only at the end of each task. Below we describe the two variants and analyse their performance respectively.

Aggressive Variant

The aggressive variant of the oracle hyper-parameter fine tuning method is outlined in Algorithm 2. The performance of Algorithm 2 is analysed in the following proposition which gives the across-tasks regret bound for the method.

Algorithm 2 Oracle Hyper-Parameter Fine Tuning Method, Aggressive Version

Input $\theta_1 = 0$, step sizes $\{\gamma_k\}_{k=1}^{nT}$, non-empty closed convex set Θ
For $t = 1, \ldots, T$
  For $i = 1, \ldots, n$
    Run a within-task online gradient descent algorithm on loss $\ell_{t,i}(\langle x_{t,i}, w_{t,i}\rangle)$ (refer to Algorithm 1)
    Update $\tilde{\theta}_{k(t,i+1)} = \theta_{k(t,i)} - \gamma_k g_k,\ g_k \in \partial\ell_{t,i}(\langle x_{t,i}, \cdot\rangle)(w_{t,i})$
    Project $\theta_{k(t,i+1)} = \mathrm{Proj}_{\Theta}(\tilde{\theta}_{k(t,i+1)})$
End
Return $(w_{t,i})_{t=1,i=1}^{T,n}$, $(\theta_k)_{k=1}^{nT}$

Proposition 2.3 (Across-Tasks Regret Bound for Algorithm 2)**.** *Consider T datasets $\mathbf{Z} = (Z_t)_{t=1}^T$, $Z_t = (x_{t,i}, y_{t,i})_{i=1}^n$ and let $(w_{t,i})_{t=1,i=1}^{T,n}$ be the iterates generated by Algorithm 2 over $\mathbf{Z}$. Let Assumption 1 hold and let step-sizes $(\gamma_k)_{k=1}^{nT}$ be tuned oracle. Then for any sequence $(w_t)_{t=1}^T, w_t \in \mathcal{C}$ and any $\theta \in \Theta$,*

$$\begin{aligned}\sum_{t=1}^{T}\sum_{i=1}^{n} \ell_{t,i}\left(\langle x_{t,i}, w_{t,i}\rangle\right) - \ell_{t,i}\left(\langle x_{t,i}, w_t\rangle\right) &\leq \sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, w_{t,i} - w_t\rangle \\ &\leq TRL\mathrm{Var}(\theta)\sqrt{n} + RL\left\|\theta\right\|\sqrt{nT} \end{aligned} \tag{2.10}$$

where

$$\mathrm{Var}(\theta) = \frac{1}{T}\sum_{t=1}^{T}\left\|w_t - \theta\right\|. \tag{2.11}$$

Proof Sketch. The idea of the proof is as follows. Since Algorithm 2 involves using the online gradient descent algorithm twice for updating weight vector $(\tilde{w}_{t,i})_{t=1,i=1}^{T,n}$ and transfer-parameter $(\theta_k)_{k=1}^{nT}$, we attempt to decompose the regret of $w_{t,i}$ into the sum of the regret of $\tilde{w}_{t,i}$ and the regret of θ_k across tasks. This sum can be written as

$$\sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, w_{t,i} - w_t\rangle = \sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, \tilde{w}_{t,i} - (w_t - \theta)\rangle + \langle g_{t,i}, \theta_{k(t,i)} - \theta\rangle.$$

To bound the above sum, we only need to bound the two regret terms in the sum respectively by applying the oracle regret bound result of the online gradient descent algorithm obtained in (2.4). □

Lazy Variant

The lazy variant of the oracle hyper-parameter fine tuning method is outlined in Algorithm 3 and its performance is analysed in the following proposition.

Algorithm 3 Oracle Hyper-Parameter Fine Tuning Method, Lazy Version

Input $\theta_1 = 0$, step sizes $\{\gamma_t\}_{t=1}^T$, non-empty closed convex set Θ
For $t = 1, \ldots, T$
  Run a within-task online gradient descent algorithm on losses $\{\ell_{t,i}(\langle x_{t,i}, w_{\theta_t,i}\rangle)\}_{i=1}^n$
  Update $\tilde{\theta}_{t+1} = \theta_t - \gamma_t \sum_{i=1}^n g_{t,i},\ g_{t,i} \in \partial\ell_{t,i}(\langle x_{t,i}, \cdot\rangle)(w_{\theta_t,i})$
  Project $\theta_{t+1} = \mathrm{Proj}_\Theta(\tilde{\theta}_{t+1})$
End
Return $(w_{\theta_t,i})_{t=1,i=1}^{T,n}$, $(\theta_t)_{t=1}^T$

Proposition 2.4 (Across-Tasks Regret Bound for Algorithm 3)**.** *Let Assumption 1 hold and consider the same setup as in Proposition 2.3 for Algorithm 3, we have*

$$\sum_{t=1}^{T}\sum_{i=1}^{n}\ell_{t,i}\left(\langle x_{t,i},w_{t,i}\rangle\right)-\ell_{t,i}\left(\langle x_{t,i},w_{t}\rangle\right)\leq\sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i},w_{t,i}-w_{t}\rangle$$
$$\leq TRL\mathrm{Var}(\theta)\sqrt{n}+\|\theta\|\,RLn\sqrt{T} \tag{2.12}$$

where

$$\mathrm{Var}(\theta)=\frac{1}{T}\sum_{t=1}^{T}\|w_t-\theta\|\,. \tag{2.13}$$

Proof Sketch. The proof of Proposition 2.4 adopts the same decomposition strategy as is used in the one of Proposition 2.3. □

2.2.4 Statistical Setting

So far, we have presented the oracle hyper-parameter fine tuning approach under the non-statistical setting where we assume no prior information on the data. In this section, we put forward arguments that we can leverage the theoretical online-to-batch conversion results to adapt the oracle hyper-parameter fine tuning methods to two kinds of statistical setting. More specifically, the aggressive variant can be converted into a statistical multi-task learning method which is able to tackle deterministic tasks, where all the tasks are predefined but the datapoints within each task come random; the lazy variant can be converted into a statistical meta-learning method which can learn random tasks, where both tasks and datapoints within each task are generated randomly.

In this section, we first briefly review the setup of statistical multi-task learning (MTL) problem and the statistical meta-learning problem. To give an overview, in the statistical MTL setup, the algorithm is trained on a sequence of tasks and is required to perform well only on these tasks, so the tasks are all determined in advance. In the statistical meta-learning setup, the method is trained on a sequence of tasks similarly but instead aims to perform well on yet-not-seen new random tasks derived from an environment. After introducing the setups, we then shortly analyse how Algorithm 2 and 3 can be adapted to these settings.

Deterministic Tasks (Statistical MTL)

Let $(\mu_t)_{t=1}^T$ be the set of tasks and $\mathbf{Z}=(Z_t)_{t=1}^T$ be the corresponding partially observed datasets of the tasks. For any $t\in\{1,\ldots,T\}$, dataset Z_t is assumed to be sampled i.i.d. from the corresponding distribution (task) μ_t, i.e., $Z_t\sim\mu_t$. A statistical MTL method aims to perform well on (test) data

generated from the same set of tasks $(\mu_t)_{t=1}^T$. The performance on each task is measured through the expected risk (or true risk) associated to each task. i.e.

$$\mathcal{R}_{\mu_t}(w) = \mathbb{E}_{(x,y)\sim\mu_t}\ell(\langle x, w\rangle, y). \tag{2.14}$$

We assume the minimizer of (2.14) is attained over $\mathbb{R}^d$ and denote the one with minimum norm as w_{μ_t}. We consider the estimator $\bar{w}_t = \frac{1}{n}\sum_{i=1}^n w_{t,i}$, namely the average of the iterates generated by Algorithm 2 associated to each task μ_t. The performance of Algorithm 2 in this case is evaluated by the multi-task risk, i.e.

$$\frac{1}{T}\sum_{t=1}^T \mathcal{R}_{\mu_t}(\bar{w}_t) - \frac{1}{T}\sum_{t=1}^T \mathcal{R}_{\mu_t}(w_{\mu_t}) \tag{2.15}$$

where the first term in (2.15) denotes the average multi-task risk of the estimators $(\bar{w}_t)_{t=1}^T$ and the second term is the benchmark multi-task oracle (the best possible performance). Ideally, for a 'good' statistical MTL method, we expect the average multi-task risk to approach the multi-task oracle with the number of tasks increases. A bound on (2.15) in expectation w.r.t. dataset $\mathbf{Z}$ is given as follows. In fact, the across-tasks regret bound for Algorithm 2 we obtained in the non-statistical setting can be automatically passed to the statistical setting with the use of the online-to-batch conversion results.

Proposition 2.5 (Multi-Task Risk Bound for Algorithm 2)**.** *Let the same assumptions in proposition 2.3 hold in the i.i.d. multi-task statistical setting. For any $\theta \in \mathbb{R}^d$, in expectation w.r.t. the sampling of the datasets $\mathbf{Z} = (Z_t)_{t=1}^T$,*

$$\mathbb{E}_{\mathbf{Z}}\left[\frac{1}{T}\sum_{t=1}^T \mathcal{R}_{\mu_t}(\bar{w}_t)\right] - \frac{1}{T}\sum_{t=1}^T \mathcal{R}_{\mu_t}(w_{\mu_t}) \leq \frac{1}{nT}(A+B) \tag{2.16}$$

where

$$A = \sum_{t=1}^T \|w_{\mu_t} - \theta\| RL\sqrt{n}, \quad B = \|\theta\| RL\sqrt{nT}. \tag{2.17}$$

Proof Sketch. The multi-task regret bound is derived by applying the across-tasks regret bound obtained in Proposition 2.3 on the right side of the following Online-To-Batch Conversion results presented in Proposition 2.6, specifying the sequence of target vectors to be $(w_{\mu_t})_{t=1}^T$. □

Proposition 2.6 (Online-To-Batch Conversion, [**Proposition 7**, 4])**.** *Consider an online algorithm that, when applied to a sequence Z, returns weight vectors $(w_i)_{i=1}^n$. Suppose we run the algorithm over*

datasets $\mathbf{Z} = (Z_t)_{t=1}^T$ *and if the datapoints are i.i.d. sampled from distributions, we have*

$$\mathbb{E}_{\mathbf{Z}}\left[\frac{1}{T}\sum_{t=1}^{T}\mathcal{R}_{\mu_t}(\bar{w}_t)\right] - \frac{1}{T}\sum_{t=1}^{T}\mathcal{R}_{\mu_t}(w_{\mu_t}) \leq \mathbb{E}_{\mathbf{Z}}\left[\frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\ell_{t,i}\left(\langle x_{t,i}, w_{t,i}\rangle\right) - \ell_{t,i}\left(\langle x_{t,i}, w_{\mu_t}\rangle\right)\right]. \tag{2.18}$$

Random Tasks (Statistical Meta-Learning)

Let $(\mu_t)_{t=1}^T$ be a sequence of tasks which are independently sampled from an environment (or meta-distribution) ρ and let $\mathbf{Z} = (Z_t)_{t=1}^T$ be the corresponding partially observed datasets of the tasks formed by n datapoints. A statistical meta-learning method aims to learn good transferable shared knowledge from these datasets and exploit it so as to perform well on a new test dataset sampled from a newly sampled task from the environment. The performance of the method on a new task is measured by the true risk w.r.t. the new task in expectation, i.e.

$$\mathcal{R}_\mu(w) = \mathbb{E}_{(x,y)\sim\mu}\ell(\langle x, w\rangle, y) \tag{2.19}$$

and we assume (2.19) admits minimizer with minimum norm w_μ over $\mathbb{R}^d$ for any $\mu \sim \rho$. The estimator we consider in this case is $\bar{w}_{\theta_{\hat{t}}}(Z) = \frac{1}{n}\sum_{i=1}^n w_{\theta_{\hat{t}},i}(Z)$ which is the average of iterates returned by Algorithm 3 when applied to a new test dataset Z with transfer-parameter $\theta_{\hat{t}}$. Here $\theta_{\hat{t}}$ is a vector uniformly sampled among the transfer-parameters inferred by Algorithm 3 applied to training datasets $\mathbf{Z} = (Z_t)_{t=1}^T$. The performance of estimator $\bar{w}_{\theta_{\hat{t}}}(Z)$ is evaluated by

$$\mathbb{E}_{\mu\sim\rho}\mathbb{E}_{Z\sim\mu^n}\mathcal{R}_\mu(\bar{w}_{\theta_{\hat{t}}}(Z)) - \mathbb{E}_{\mu\sim\rho}\mathcal{R}_\mu(w_\mu) \tag{2.20}$$

where the first term in (2.20) denotes the transfer risk of the estimator $\bar{w}_{\theta_{\hat{t}}}(Z)$ and the second term is the benchmark meta-learning oracle (the best possible performance for a new task in expectation). Similarly, we wish the transfer risk of the estimator could approach the meta-learning oracle with the increase of number of tasks and datapoints. A bound on (2.20) in expectation w.r.t. the sampling of datasets $\mathbf{Z}$ and the uniform sampling of $\hat{t}$ is provided in the following proposition.

Proposition 2.7 (Transfer Risk Bound for Algorithm 3, [4])**.** *Let the same assumptions in proposition 2.3 hold in the i.i.d. multi-task statistical setting. For any* $\theta \in \Theta$*, in expectation w.r.t. the sampling of the datasets* $\mathbf{Z} = (Z_t)_{t=1}^T$ *and the uniform sampling of* $\hat{t} \sim \mathcal{U}(T)$

$$\mathbb{E}_{\hat{t}\sim\mathcal{U}(T)}\mathbb{E}_{\mathbf{Z}}\left[\mathbb{E}_{\mu\sim\rho}\mathbb{E}_{Z\sim\mu^n}\mathcal{R}_\mu(\bar{w}_{\theta_{\hat{t}}}(Z))\right] - \mathbb{E}_{\mu\sim\rho}\mathcal{R}_\mu(w_\mu) \leq \frac{1}{nT}(A+B) \tag{2.21}$$

where

$$A = \mathbb{E}_{\mu\sim\rho} \|w_\mu - \theta\| \, L\sqrt{n}T, \quad B = \|\theta\| \, Ln\sqrt{T}. \tag{2.22}$$

Proof Sketch. The proof of Proposition 2.7 uses a similar online-to-batch conversion theoretical results to Proposition 2.6 (see details in [**Proposition 10**, 4]) to pass the results obtained in Proposition 2.4 to the statistical meta-learning setting. □

2.3 From Homogeneous Sequence to Heterogeneous Sequence

In this section, we study the conditional approach for tackling heterogeneous tasks. As is already mentioned in Chapter 1, we describe the conditional framework proposed in [3] and give a rough idea on how could such approach handle heterogeneity among tasks.

2.3.1 Starting point as conditioning function

Recall that in the homogeneous tasks setting, knowledge cross tasks is captured by a common parameter θ (i.e. the transfer-parameter in the translated fine tuning method). However, there could be situations where we receive a sequence of heterogeneous tasks and a single parameter is not sufficient to identify the relatedness among these tasks. This limitation can be addressed by a recently proposed conditional approach in [3], as described below.

First of all, the intuition behind the conditioning technique is: when we have diversified tasks, it is natural to think that each group/type of tasks has its own common parameter representing relatedness among the tasks within the same group. To cope with the diversity of task groups, we need a mechanism for our method to be able to identify which common parameter should the algorithm assign when receiving a new task. To execute this idea, we require additional information that can help us do such identification. Therefore, in addition to a sequence of datasets $\mathbf{Z} = (Z_t)_{t=1}^T$, we now assume that a collection of side information $\mathbf{S} = (s_t)_{t=1}^T \in \mathcal{S}^T$ associated with each task is also available. For instance, in the previous PC recommendation example, this side information could be the characteristics of customers - whether he/she is a student, a business person or else. To leverage such side information when identifying the heterogeneity among tasks, the transfer-parameter $\theta \in \Theta$ is now conditioned on side information $s \in \mathcal{S}$ and is represented by a function τ mapping s to θ, i.e. $\tau : \mathcal{S} \to \Theta$.

We give an illustration of this technique by adapting the oracle hyper-parameter within-task learner (see Algorithm 1) to the environment of heterogeneous tasks using the conditioning function. In this case, we do not know which group the received task Z comes from but are given the side information s associated to the task. To do the adaptation, we only need to replace the fixed transfer-parameter θ in

Algorithm 1 with the value returned by a conditioning function $\tau(\cdot)$ taking s as input. Below we list the adapted algorithm using conditioning function.

Algorithm 4 Online Gradient Descent with conditioning function

Input $\tilde{w}_1 = 0$, conditioning function τ, side information s, step sizes $\{\eta_i\}$, non-empty closed convex set $\mathcal{C}$
For $i = 1, \ldots, n$
Predict $\langle x_i, w_i \rangle$ and observe loss $\ell_i(\langle x_i, w_i \rangle)$
Update $\tilde{w}_{i+1} = \tilde{w}_i - \eta_i g_i$, $g_i \in \partial \ell_i(\langle x_i, \cdot \rangle)(w_i)$
Translate $w_{i+1} = \tilde{w}_{i+1} + \tau(s)$
Project $w_{i+1} = \mathrm{Proj}_{\mathcal{C}}(w_{i+1})$
End
Return $(w_i)_{i=1}^n$

2.3.2 The advantage of using the right conditioning function

In this section, we study the advantage of using an appropriate conditioning function through the example of Algorithm 4. Similar to the discussion in Section 2.2.2, we analyse the performance of Algorithm 4 with the same conditioning function $\tau(\cdot)$ against a sequence of target vectors $(w_t)_{t=1}^T$ when applied to a sequence of heterogeneous tasks $\mathbf{Z} = (Z_t)_{t=1}^T$ with associated side information $\mathbf{S} = (s_t)_{t=1}^T$. By construction, the target vectors are expressed as

$$w_t = \tilde{w}_t + \tau(s). \tag{2.23}$$

We measure the performance by the across-tasks regret and we analyse its bound as follows.

Proposition 2.8 (Across-Tasks Regret Bound for Algorithm 4)**.** *For any task $t = 1, \ldots, T$, let $(w_{t,i})_{i=1}^n$ be the iterates generated by Algorithm 4 over the dataset Z_t and side information s_t with starting point computed by conditioning function $\tau(s_t)$. Let Assumption 1 hold and let the learning rate η be the oracle hyper-parameter, then for any sequence $(w_t)_{t=1}^T$, $w_t \in \mathcal{C}$,*

$$\begin{aligned}\sum_{t=1}^{T}\sum_{i=1}^{n} \ell_{t,i}\left(\langle x_{t,i}, w_{t,i}\rangle\right) - \ell_{t,i}\left(\langle x_{t,i}, w_t\rangle\right) &\leq \sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, w_{t,i} - w_t\rangle \\ &\leq TRL\,\mathrm{Var}(\tau)\sqrt{n}\end{aligned} \tag{2.24}$$

where

$$\mathrm{Var}(\tau) = \frac{1}{T}\sum_{t=1}^{T} \|w_t - \tau(s_t)\| \,. \tag{2.25}$$

Proof Sketch. We first derive the regret for Algorithm 4 similarly in Proposition 2.2 by replacing the transfer-parameter θ with the value $\tau(s)$ returned by the conditioning function, i.e. $\|w - \tau(s)\| \, RL\sqrt{n}$. We then sum the regret over all datasets and obtain the desired statement. □

We write $\mathrm{Var}(\tau)$ for the quantity (2.25) with slight abuse of notation in that $\mathrm{Var}(\tau)$ is in fact the lower bound of the variance of target vectors w_t w.r.t. $\tau(s_t)$ according to Jensen's inequality. We see Proposition 2.8 implies that the across-tasks regret bound in (2.24) grows in the order of $\mathcal{O}(\mathrm{Var}(\tau)\sqrt{n}T)$, suggesting the optimal choice for the conditioning function in Algorithm 4 is the one that minimizes the variance of the target vectors $(w_t)_{t=1}^T$. Hence, the advantage of using a good conditioning function in Algorithm 4 is more significant for low variance tasks.

Chapter 3

Parameter-Free Online Learning

In the previous chapter, we have introduced the fine tuning framework which requires oracle hyper-parameter tuning to achieve optimal performance. Whilst it is common for a machine learning algorithm to have some hyper-parameters to be tuned, we have seen in Chapter 2 that oracle tuning is impractical in the non-statistical setting. Fortunately, recent studies have shown that it is possible to achieve similar rates of regret bound attained in the oracle tuning case without tuning any hyper-parameters in the non-statistical setting. This line of work is referred to as parameter-free online algorithms. In this chapter, we briefly recall two well-known parameter-free online algorithms and show their comparable performance.

3.1 Basic Algorithms

3.1.1 Online Projected Subgradient Algorithm

The online projected subgradient algorithm is a generic online algorithm to solve online learning problems with convex losses. We refer readers to [11, 15] for more detailed formulations and analyses. The algorithm in stated in Algorithm 5 and we observe that there is no requirement to tune any hyper-parameters in the method, hence making it parameter-free.

Algorithm 5 Online Projected Subgradient Algorithm, [11]

Input $\mathcal{C} \subset \mathbb{R}^d$, $(g_i)_{i=1}^n$, $g_i \in \mathbb{R}^d$, $\|g_i\| \leq C$
Initialize $v_1 \in \mathcal{C}$
For $i = 1, \ldots, n$
  Receive g_i
  Define $\gamma_i = \frac{\text{diam}(\mathcal{C})}{C\sqrt{2i}}$
  Update $v_{i+1} = \text{proj}_{\mathcal{C}}\left(v_i - \gamma_i g_i\right) = \text{argmin}_{y \in \mathcal{C}} \ \|v_i - \gamma_i g_i - y\|$
End
Return $(v_i)_{i=1}^n$

The algorithm takes input a convex, closed and non-empty set $\mathcal{C} \subset \mathbb{R}^d$ with diameter

$$\operatorname{diam}(\mathcal{C}) = \sup_{v,v' \in \mathcal{C}} \|v - v'\| . \tag{3.1}$$

At each iteration i, the algorithm receives a bounded vector $g_i \in \mathbb{R}^d$ with norm $\|g_i\| \leq C$ for some $C > 0$ that determines the direction towards which we would like to minimize the target function. By the online learning problem convention, the target is to minimize the loss incurred in predictions and g_i is usually computed as the subgradient of the convex loss ℓ_i with respect to the predictions. Since the algorithm restricts the search of target vector v_i to the set $\mathcal{C}$, it performs a descent step along the vector g_i with an appropriate length and then projects the resultant vector back on the restriction set $\mathcal{C}$.

To demonstrate the performance of the algorithm, we describe the regret bound for Algorithm 5 with respect to the convex losses ℓ_i in the following proposition. By the convexity of the loss function, we could reduce the regret w.r.t. losses to regret w.r.t. linear losses.

Proposition 3.1 (Linear Regret Bound for Algorithm 5, [**Theorem 3.1**, 11])**.** *The iterations* $(v_i)_{i=1}^n$ *returned by Algorithm 5 satisfy the following linear regret bound w.r.t. a competitor vector* $v \in \mathcal{C}$

$$\sum_{i=1}^{n} \ell_i(v_i) - \ell_i(v) \leq \sum_{i=1}^{n} \langle g_i, v_i - v \rangle \leq C\sqrt{2} \operatorname{diam}(\mathcal{C}) \sqrt{n}. \tag{3.2}$$

It is important to note that although Algorithm 5 is presented using vector $v \in \mathcal{C}$, the algorithm can be adapted to matrix form also with the same regret bound in Proposition 3.1 holds. The matrix form of Algorithm 5 will be outlined in Appendix A.

3.1.2 Coin Betting Algorithm

Another recently developed parameter-free online algorithm is the one-dimension coin betting algorithm. The concept of this coin betting approach originates from a known problem in economics and hence the algorithm we show below looks very different from the usual online learning algorithms.

To give a picture of the idea behind the algorithm, consider the following game: Let us assume that we are given initial wealth $\epsilon > 0$ and, at each round i, are asked to bet p_i money, which is computed as proportion b_i of the current wealth u_i we hold, i.e. $p_i = b_i u_i$ on a coin. We then receive the coin outcome $c_i \in [-1, 1]$ and get $c_i p_i$ money. Our wealth thus becomes $u_i + c_i p_i$. The goal of this game is to design a betting strategy that can increase the wealth as much as possible. The protocol of a winning betting strategy is demonstrated in Algorithm 6.

In Algorithm 6, the outcome of the coins is set as $c_i = -\frac{g_i}{C} \in [-1, 1]$ and g_i is usually set as the subgradient of the losses on the current prediction in online learning convention. At this point,

Algorithm 6 1-dim Coin Betting Algorithm based on Krichevsky-Trofimov (KT) estimator, [16]

Input $(g_i)_{i=1}^n$, $g_i \in \mathbb{R}$, $|g_i| \leq C$, $\epsilon > 0$
Initialize $b_1 = 0, u_1 = \epsilon, p_1 = b_1 u_1$
For $i = 1, \ldots, n$
  Receive g_i
  Define $u_{i+1} = u_i - \frac{1}{C} g_i p_i$
  Define $b_{i+1} = \frac{1}{i}\left((i-1)b_i - \frac{1}{C}g_i\right)$
  Update $p_{i+1} = b_{i+1} u_{i+1}$
End
Return $(p_i)_{i=1}^n$

readers may wonder why Algorithm 6 is considered parameter-free even though it has one tunable hyper-parameter, which is the initial wealth ϵ. We will show later that in fact there is a wide range in which the choice of the initial wealth does not affect the overall performance of the algorithm.

As explained previously in Section 3.1.1, through reduction, we only need to consider the linear regret of Algorithm 6 to evaluate its performance. The bound on the linear regret is given in the following proposition.

Proposition 3.2 (Linear Regret Bound for Algorithm 5, [**Corollary 5**, 16])**.** *The iterations* $(p_i)_{i=1}^n$ *returned by Algorithm 6 satisfy the following linear regret bound w.r.t. a competitor scalar* $p \in \mathbb{R}$

$$\sum_{i=1}^{n} g_i(p_i - p) \leq C\left[\epsilon + \Psi\left(\epsilon^{-1}|p|n\right)|p|\sqrt{n}\right] \tag{3.3}$$

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

We can see that any value of ϵ in $[1, \sqrt{n}]$ would not change the asymptotic $\sqrt{n}$ rate of the bound above, which makes Algorithm 6 parameter-free. The logarithmic factor in the regret bound stems from the procedure of converting coin betting problem to online linear optimization problem and hence is unavoidable.

Part II

Our Methods

Chapter 4

Parameter-Free Fine Tuning Framework for a Heterogeneous Sequence

In this chapter, we propose an online parameter-free fine tuning method to learn a good conditioning function for a possibly non-finite and non-heterogeneous sequence of tasks. We first present the parameter-free within-task learner and then demonstrate the aggressive and lazy version of our proposed fine tuning method. After that, we also show how the two variants can be converted into a statistical MTL method and a statistical meta-learning method respectively.

4.1 Within-Task Algorithm

Given a task dataset $Z = (x_i, y_i)_{i=1}^n$ and its side information $s \in \mathcal{S}$, the parameter-free within-task algorithm we propose is outlined in Algorithm 7. The design of the algorithm is motivated by [2, 4], using a decomposition technique to estimate a vector in a parameter-free manner by using two parameter-free algorithms to learn its magnitude and direction respectively and then combining together to obtain the final estimate.

Inspired by [4], we use the coin betting algorithm (see Algorithm 6) to learn the magnitude of weight vectors and the online subgradient descent algorithm (see Algorithm 5) to learn the direction of weight vectors. Different from [4], we introduce the conditional transfer-parameter $\tau(s)$ computed by a conditioning function $\tau(\cdot) \in \mathcal{T}$ with side information s as input in the magnitude-direction decomposition. To use a good conditioning function in practice, it is reasonable to restrict the class of functions rather than using the entire space $\mathcal{T}$ of measurable functions. In the remaining of this work, we consider a set of linear functions for the class of conditioning functions. For a given feature map $\Phi : \mathcal{S} \to \mathbb{R}^k$ on the side information space, we define the associated space of linear functions as

$$\mathcal{T}_\Phi = \left\{\tau : \mathcal{S} \to \mathbb{R}^d \,|\, \tau(\cdot) = M\Phi(\cdot) + b, \text{ for some } M \in \mathbb{R}^{d\times k}, b \in \mathbb{R}^d\right\}. \tag{4.1}$$

To highlight the dependency of the conditioning function $\tau \in \mathcal{T}_\Phi$ w.r.t. its parameter M and b, we will use the notation $\tau_{M,b}$. Regarding the feature map $\Phi(\cdot)$, we introduce the following assumption which will allow the design our methods.

Assumption 2. *There exists $K > 0$ such that $\|\Phi(s)\| \leq K$ for any $s \in \mathcal{S}$.*

To illustrate the aforementioned decomposition, we show that given any vector $w \in \mathbb{R}^d$ and side information $s \in \mathcal{S}$, the target weight vector w can always be written w.r.t. the coordinate system centered in $\tau_{M,b}(s) = M\Phi(s) + b$ as follows.

$$w = pv + M\Phi(s) + b \tag{4.2}$$

where

$$p = \|w - M\Phi(s) - b\| \in \mathbb{R} \qquad v = \frac{w - M\Phi(s) - b}{\|w - M\Phi(s) - b\|} \in \mathcal{B}(0,1). \tag{4.3}$$

Algorithm 7 Parameter-Free Within-Task Algorithm with Fixed Conditional Starting Point

Input $Z = (z_i)_{i=1}^n$, $s \in \mathcal{S}$, $e > 0$, $M \in \mathbb{R}^{d\times k}$, $b \in \mathbb{R}^d$, feature map $\Phi(\cdot)$, L and R in Assumption 1.
Initialize $b_1 = 0, u_1 = e, p_1 = b_1 u_1, v_1 = 0 \in \mathcal{B}(0,1)$
For $i = 1, \ldots, n$
  Vector update $w_i = p_i v_i + M\Phi(s) + b$
  Receive the datapoint $z_i = (x_i, y_i)$
  Compute $g_i = h_i x_i, h_i \in \partial\ell_i(\langle x_i, w_i\rangle) \in \mathbb{R}$
  Define $\gamma_i = \frac{1}{LR}\sqrt{\frac{2}{i}}$
  Direction update $v_{i+1} = \mathrm{proj}_{\mathcal{B}(0,1)}(v_i - \gamma_i g_i)$
  Define $u_{i+1} = u_i - \frac{1}{RL}\langle g_i, v_i\rangle p_i$
  Define $b_{i+1} = \frac{1}{i}\big((i-1)b_i - \frac{1}{RL}\langle g_i, v_i\rangle\big)$
  Magnitude update $p_{i+1} = b_{i+1}u_{i+1}$
End
Return $(w_i)_{i=1}^n$

In Algorithm 7, we are given the datapoints $Z = (z_i)_{i=1}^n = (x_i, y_i)_{i=1}^n$, side information s associated with Z and a linear conditioning function $M\Phi(\cdot) + b$ to incrementally learn

- the direction $v \in \mathcal{B}(0,1)$ of the vector $w - M\Phi(s) - b$ by applying Algorithm 5 on the ball $\mathcal{B}(0,1)$ to the subgradient vectors $(g_i)_{i=1}^n$, where $g_i \in \partial\ell_i(\langle x_i, \cdot\rangle)(w_i)$.

- the magnitude p of the vector $w - M\Phi(s) - b$ by applying Algorithm 6 to the scalars $(\langle g_i, v_i \rangle)_{i=1}^n$ with v_i being the current direction w.r.t. the conditional transfer-parameter $M\Phi(s) + b$ returned by the conditioning function.

We provide the regret bound of Algorithm 7 in the following proposition. The proof of this result is simply an adaptation of the proof technique used in [4], changing the fixed transfer-parameter to a conditional one. We provide a sketch of the proof below since the main idea behind the proof is useful when deriving the main results of our proposed methods in later sections. The full proof can be found in Appendix B.1 for completeness.

Proposition 4.1 (Single-Task Regret Bound for Algorithm 7, Adaptation of [**Proposition 3**, 4])**.** *Let Assumption 1 hold and let $(w_i)_{i=1}^n$ be the iterates generated by Algorithm 7 with conditional transfer-parameter $\tau(s)$. Then, for any $w \in \mathbb{R}^d$,*

$$\begin{aligned}
&\sum_{i=1}^{n} \ell_i(\langle x_i, w_i \rangle) - \ell_i(\langle x_i, w \rangle) \leq \sum_{i=1}^{n} \langle g_i, w_i - w \rangle \\
&\leq RL\left[e + \left(2\sqrt{2} + \Psi\left(e^{-1} \left\| w - M\Phi(s) - b \right\| n\right)\right) \left\| w - M\Phi(s) - b \right\| \sqrt{n}\right] \qquad (4.4)
\end{aligned}$$

where $\Psi(a) = \sqrt{\log(1 + 24a^2)}$.

Proof Sketch. The first step is to show the regret w.r.t. losses can be bounded by linear regret of $(w_i)_{i=1}^n$ by using the convexity of loss function (see Assumption 1) and the definition of subgradients $(g_i)_{i=1}^n$. The next step is to decompose the linear regret in the first inequality into the sum of linear regret of magnitudes $(p_i)_{i=1}^n$ and the linear regret of directions $(v_i)_{i=1}^n$, i.e.

$$\sum_{i=1}^{n} \langle g_i, w_i - w \rangle = \sum_{i=1}^{n} \langle g_i, v_i \rangle (p_i - p) + p \sum_{i=1}^{n} \langle g_i, v_i - v \rangle.$$

Because $(p_i)_{i=1}^n$ are returned by the coin-betting algorithm and $(v_i)_{i=1}^n$ are returned by the online subgradient descent algorithm, we bound the decomposed sum by applying the linear regret bounds obtained for the two base online algorithms to derive the desired results in the statement. □

4.2 The advantage of using the right conditioning function

In this section, we recall the advantage of using an appropriate conditioning function for a heterogeneous sequence and specify the analysis to Algorithm 7. When Algorithm 7 is applied to a sequence of heterogeneous tasks $\mathbf{Z} = (Z_t)_{t=1}^T$ with associated side information $\mathbf{S} = (s_t)_{t=1}^T$, the across-tasks regret is

$$\sum_{t=1}^{T}\left(\sum_{i=1}^{n}\ell_{t,i}\left(\langle x_{t,i},w_{t,i}\rangle\right)-\min_{w_t\in\mathbb{R}^d}\sum_{i=1}^{n}\ell_{t,i}\left(\langle x_{t,i},w_t\rangle\right)\right), \tag{2.5}$$

where the target weight vectors are represented as

$$w_t = p_t v_t + M\Phi(s_t) + b. \tag{4.5}$$

The across-tasks regret can be bounded by the following quantity in proposition 4.2.

Proposition 4.2 (Across-Tasks Regret Bound for Algorithm 7)**.** *Let Assumption 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 Algorithm 7 over the dataset Z_t with associated side information s_t using the linear conditioning function $M\Phi(\cdot)+b$. Then for any sequence $(w_t)_{t=1}^T, w_t \in \mathbb{R}^d$,*

$$\begin{aligned}\sum_{t=1}^{T}\sum_{i=1}^{n}\ell_{t,i}\left(\langle x_{t,i},w_{t,i}\rangle\right)-\ell_{t,i}\left(\langle x_{t,i},w_t\rangle\right) &\le \sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, w_{t,i}-w_t\rangle \\ &\le RL\left[eT+\left(2\sqrt{2}\mathrm{Var}(\tau_{M,b})+\widehat{\mathrm{Var}}(\tau_{M,b})\right)\sqrt{n}T\right]\end{aligned} \tag{4.6}$$

where

$$\begin{aligned}\mathrm{Var}(\tau_{M,b}) &= \frac{1}{T}\sum_{t=1}^{T}\left\|w_t - M\Phi(s_t) - b\right\|, \\ \widehat{\mathrm{Var}}(\tau_{M,b}) &= \frac{1}{T}\sum_{t=1}^{T}\Psi\left(e^{-1}\left\|w_t - M\Phi(s) - b\right\| n\right)\left\|w_t - M\Phi(s) - b\right\|\end{aligned} \tag{4.7}$$

where $\Psi(a) = \sqrt{\log(1+24a^2)}$.

Proof Sketch. The statement can be derived by summing the regret obtained for the within-task Algorithm 7 in Proposition 4.1 over all datasets $(Z_t)_{t=1}^T$. □

As discussed in Chapter 2, we use a slight abuse of notation regarding the term $\mathrm{Var}(\tau_{M,b})$. We see that when the logarithmic factor is negligible for each task $t \in \{1,\dots,T\}$, namely

$$\Psi\left(e^{-1}\left\|w_t - M\Phi(s) - b\right\| n\right) \approx 1,$$

we have $\widehat{\mathrm{Var}}(\tau_{M,b}) \approx \mathrm{Var}(\tau_{M,b})$. Therefore, in this case, the leading term in the across-tasks regret

bound in (4.6) grows in the order of $\mathcal{O}(\mathrm{Var}(\tau_{M,b})\sqrt{n}T)$. This suggests us that the optimal choice of the linear conditioning function $\tau_{M,b}$ in Algorithm 7 is the one that minimizes the variance of the target vectors $(w_t)_{t=1}^T$, i.e.

$$\arg \min_{M \in \mathbb{R}^{d \times k}, b \in \mathbb{R}^d} \sqrt{\frac{1}{T} \sum_{t=1}^{T} \|w_t - M\Phi(s_t) - b\|^2}, \tag{4.8}$$

which is

$$\begin{aligned} b &= \frac{1}{T} \sum_{t=1}^{T} w_t - M\Phi(s_t) \\ M &= \left(\sum_{t=1}^{T} (w_t - b)\Phi(s_t)^\top \right) \left(\sum_{t=1}^{T} \Phi(s_t)\Phi(s_t)^\top \right)^{-1}. \end{aligned} \tag{4.9}$$

4.3 Estimating the conditioning function from data

We now propose the online parameter-free fine tunning method that can learn a good linear conditioning function shared across heterogeneous tasks from an increasing sequence of T datasets $\mathbf{Z} = (Z_t)_{t=1}^T$, $Z_t = (z_{t,i})_{i=1}^n = (x_{t,i}, y_{t,i})_{i=1}^n$ with their associated side information $\mathbf{S} = (s_t)_{t=1}^T$.

We observe that the linear function class $\mathcal{T}_\Phi$ we adopt in our method also contains the space of all unconditional starting points (i.e. by setting M as zero matrix), so that the conditioning function is able to capture the homogeneity of tasks as well. We also see that $\tau_{M,b} = 0$ when $M = 0_{d \times k}$ and $b = 0$, implying that the class of conditioning functions also contain the solution which corresponds to solve each task independently. This guarantees that our method is able to perform independent task learning (ITL) and hence its performance should be no worse than ITL, avoiding the so-called *negative transfer* effect. Our proposed method can be of the "aggressive" variant and the "lazy" variant. Below we outline the two variant algorithms and analyse their performance accordingly.

Aggressive Variant

We describe the aggressive variant of our method in Algorithm 8 where the conditioning function is updated after receiving each new observation. The concept behind Algorithm 8 is similar to the idea of parameterizing the weight vector associated to each task with a conditioning function $\tau(s) \in \mathbb{R}^d$, as shown in Equation (2.23). But here we represent the conditioning transfer-parameter using the specific linear function form we introduced previously, i.e.

$$w_t = p_t v_t + M\Phi(s_t) + b \tag{4.10}$$

$$p_t = \|w_t - M\Phi(s_t) - b\| \in \mathbb{R} \qquad v_t = \frac{w_t - M\Phi(s_t) - b}{\|w_t - M\Phi(s_t) - b\|} \in \mathcal{B}(0,1). \tag{4.11}$$

and we further decompose the parameters $M \in \mathbb{R}^{d\times k}$ and $b \in \mathbb{R}^d$ into directions and magnitudes w.r.t. the zero-centered coordinate system respectively as:

$$M = \mathscr{P}\mathscr{V} \tag{4.12}$$

$$\mathscr{P} = \|M\|_F \in \mathbb{R} \qquad \mathscr{V} = \frac{M}{\|M\|_F} \in \mathcal{B}(0,1) \tag{4.13}$$

and

$$b = PV \tag{4.14}$$

$$P = \|b\| \in \mathbb{R} \qquad V = \frac{b}{\|b\|} \in \mathcal{B}(0,1). \tag{4.15}$$

Algorithm 8 exploits the triple parameterization described above and uses the datasets $\mathbf{Z}$ and side information $\mathbf{S}$ it receives to incrementally learn

- (for any task t) the direction $v_t \in \mathcal{B}(0,1)$ of the weight vector $w_t - M\Phi(s_t) - b$ by applying Algorithm 5 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 $w_{t,i}$ being the current within-task iterate returned by Algorithm 8,
- (for any task t) the magnitude p_t of the weight vector $w_t - M\Phi(s_t) - b$ by applying Algorithm 6 to the scalars $(\langle g_{t,i}, v_{t,i}\rangle)_{i=1}^n$, with $v_{t,i}$ being the current within-task direction w.r.t. the current conditional transfer-parameter $\theta_{k(t,i)}$ estimated by Algorithm 8,
- the direction $\mathscr{V} \in \mathcal{B}(0,1)$ of the transition matrix M by applying Algorithm 5 on the ball $\mathcal{B}(0,1)$ to the vectors $(g_{t,i}\Phi(s_t)^\top)_{t=1,i=1}^{T,n}$,
- the magnitude $\mathscr{P}$ of the transition matrix M by applying Algorithm 6 to the scalars $(\langle g_{t,i}\Phi(s_t)^\top, \mathscr{V}_{k(t,i)}\rangle_F)_{t=1,i=1}^{T,n}$, with $\mathscr{V}_{k(t,i)}$ being the current meta-direction of the transition matrix estimated by Algorithm 8,
- the direction $V \in \mathcal{B}(0,1)$ of the shift vector b by applying Algorithm 5 on the ball $\mathcal{B}(0,1)$ to the vectors $(g_{t,i})_{t=1,i=1}^{T,n}$,

- the magnitude P of the shift vector b by applying Algorithm 6 to the scalars $\left(\left\langle g_{t,i}, V_{k(t,i)}\right\rangle\right)_{t=1,i=1}^{T,n}$, with $V_{k(t,i)}$ being the current meta-direction of the shift vector estimated by by Algorithm 8.

We now evaluate the performance of Algorithm 8 by giving an across-tasks regret bound for the method. The result is presented in the following theorem and a complete proof is provided in Appendix B.2.

Theorem 4.3 (Across-Tasks Regret Bound for Algorithm 8)**.** *Let Assumption 1 and 2 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 with associated T datasets of side information $\mathbf{S} = (s_t)_{t=1}^T$ w.r.t. each task. Let $(w_{t,i})_{t=1,i=1}^{T,n}$ be the iterates generated by Algorithm 8 over datasets $\mathbf{Z}$ and $\mathbf{S}$. Then, for any sequence $(w_t)_{t=1}^T$, $w_t \in \mathbb{R}^d$ and any $M \in \mathbb{R}^{d\times k}$, $b \in \mathbb{R}^d$,*

$$\sum_{t=1}^{T}\sum_{i=1}^{n} \ell_{t,i}\left(\langle x_{t,i}, w_{t,i}\rangle\right) - \ell_{t,i}\left(\langle x_{t,i}, w_t\rangle\right) \leq \sum_{t=1}^{T}\sum_{i=1}^{n} \left\langle g_{t,i}, w_{t,i} - w_t\right\rangle \leq A + B + C \tag{4.16}$$

where

$$\begin{aligned} A &= RL\left[eT + \left(2\sqrt{2}\mathrm{Var}(\tau_{M,b}) + \widehat{\mathrm{Var}}(\tau_{M,b})\right)\sqrt{n}T\right] \\ B &= RLK\left[\mathscr{E} + \left(2\sqrt{2} + \Psi\left(\mathscr{E}^{-1}\left\|M\right\|_F nT\right)\right)\left\|M\right\|_F\sqrt{nT}\right] \\ C &= RL\left[E + \left(2\sqrt{2} + \Psi\left(E^{-1}\left\|b\right\| nT\right)\right)\left\|b\right\|\sqrt{nT}\right] \end{aligned} \tag{4.17}$$

with

$$\begin{aligned} \mathrm{Var}(\tau_{M,b}) &= \frac{1}{T}\sum_{t=1}^{T}\left\|w_t - M\Phi(s_t) - b\right\|, \\ \widehat{\mathrm{Var}}(\tau_{M,b}) &= \frac{1}{T}\sum_{t=1}^{T}\Psi\left(e^{-1}\left\|w_t - M\Phi(s_t) - b\right\| n\right)\left\|w_t - M\Phi(s_t) - b\right\|. \end{aligned} \tag{4.18}$$

and $\Psi(a) = \sqrt{\log(1+24a^2)}$.

Proof Sketch. The first inequality is derived from the convexity of the loss function (see Assumption 1) and the definition of the subgradients $(g_{t,i})_{t=1,i=1}^{T,n}$. To obtain the second inequality, namely the bound on the linear regret, we first decompose the linear regret $\sum_{t=1}^{T}\sum_{i=1}^{n}\left\langle g_{t,i}, w_{t,i} - w_t\right\rangle$ into the sum of the linear regret of zero-centered weight vector $w_t - M\Phi(s_t) - b$, transition matrix M and shift vector b. Since the three quantity: $w_t - M\Phi(s_t) - b$, M and b are estimated through magnitude-direction decomposition using the coin betting algorithm and the online subgradient algorithm, we can further

Algorithm 8 Parameter-Free Fine Tuning Method with Conditional Function, Aggressive Version

Input $\mathbf{Z} = (Z_t)_{t=1}^T, Z_t = (z_i)_{i=1}^n = (x_{t,i}, y_{t,i})_{i=1}^n$, $\mathbf{S} = (s_t)_{t=1}^T$, $\Phi(\cdot)$, $e > 0, E > 0, \mathscr{E} > 0$, L, R and K as in Assumption 1, 2

Initialize $\mathscr{B}_1 = 0, \mathscr{U}_1 = \mathscr{E}, \mathscr{P}_1 = \mathscr{B}_1\mathscr{U}_1, \mathscr{V}_1 = O_{d\times k} \in \mathcal{B}(0,1)$, $B_1 = 0, U_1 = E, P_1 = B_1U_1, V_1 = 0 \in \mathcal{B}(0,1)$

For $t = 1, \ldots, T$

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 $i = 1, \ldots, n$

Define $k = k(t,i) = (t-1)n + i$

Meta-transition matrix update $M_k = \mathscr{P}_k\mathscr{V}_k$

Meta-shift update $b_k = P_kV_k$

Conditional meta-vector update $\theta_k = M_k\Phi(s_t) + b_k$

Within-vector update $w_{t,i} = p_{t,i}v_{t,i} + \theta_k$

Receive the datapoint $z_{t,i} = (x_{t,i}, y_{t,i})$

Compute $g_{t,i} = h_{t,i}x_{t,i}, h_{t,i} \in \partial\ell_{t,i}(\langle x_{t,i}, w_{t,i}\rangle)$

Receive $g_{t,i}\Phi(s_t)^\top$

Define $\zeta_k = \frac{1}{LRK}\sqrt{\frac{2}{k}}$

Direction update $\mathscr{V}_{k+1} = \mathrm{proj}_{\mathcal{B}(0,1)}(\mathscr{V}_k - \zeta_k g_{t,i}\Phi(s_t)^\top)$

Define $\eta_k = \frac{1}{LR}\sqrt{\frac{2}{k}}$

Direction update $V_{k+1} = \mathrm{proj}_{\mathcal{B}(0,1)}(V_k - \eta_k g_{t,i})$

Define $\gamma_{t,i} = \frac{1}{LR}\sqrt{\frac{2}{i}}$

Direction update $v_{t,i+1} = \mathrm{proj}_{\mathcal{B}(0,1)}(v_{t,i} - \gamma_{t,i}g_{t,i})$

Define $\mathscr{U}_{k+1} = \mathscr{U}_k - \frac{1}{RLK}\big\langle g_{t,i}\Phi(s_t)^\top, \mathscr{V}_k\big\rangle_F\mathscr{P}_k$

Define $\mathscr{B}_{k+1} = \frac{1}{k}\big((k-1)\mathscr{B}_k - \frac{1}{RLK}\big\langle g_{t,i}\Phi(s_t)^\top, \mathscr{V}_k\big\rangle_F\big)$

Magnitude update $\mathscr{P}_{k+1} = \mathscr{B}_{k+1}\mathscr{U}_{k+1}$

Define $U_{k+1} = U_k - \frac{1}{RL}\big\langle g_{t,i}, V_k\big\rangle P_k$

Define $B_{k+1} = \frac{1}{k}\big((k-1)B_k - \frac{1}{RL}\big\langle g_{t,i}, V_k\big\rangle\big)$

Magnitude update $P_{k+1} = B_{k+1}U_{k+1}$

Define $u_{t,i+1} = u_{t,i} - \frac{1}{RL}\langle g_{t,i}, v_{t,i}\rangle p_{t,i}$

Define $b_{t,i+1} = \frac{1}{i}\big((i-1)b_{t,i} - \frac{1}{RL}\langle g_{t,i}, v_{t,i}\rangle\big)$

Magnitude update $p_{t,i+1} = b_{t,i+1}u_{t,i+1}$

End

End

Return $(w_{t,i})_{t=1,i=1}^{T,n}$, $(M_k)_{k=1}^{Tn}$ and $(b_k)_{k=1}^{Tn}$

break down the linear regret of each quantity into a sum of the linear regret of the coin betting algorithm and the online subgradient algorithm. Hence we finally have the sum of six degraded linear regret terms to deal with. To obtain the desired bound in the statement, we only need to bound these six linear regret terms separately by using the regret bound results we introduced in Chapter 3 and add them up. In Appendix B.2, we will show how such decomposition is conducted formally. □

We observe that the bound obtained in Theorem 4.3 is primarily comprised of three terms. We note that term A coincides with the bound we obtained in Proposition 4.2 for using a fixed conditioning function over all tasks and thus term B and term C reflect the price we pay for estimating the parameter of our linear conditioning function from data. We see that term B and C grow with $\mathcal{O}(\sqrt{nT})$ which is negligible when added to $\mathcal{O}(\sqrt{n})$ in term A. Particularly, setting the parameter M and b as the optimal one we computed in (4.9), we can conclude that our method is able to match the performance of using the best conditioning function in hindsight when the number of tasks is sufficiently large.

Lazy Variant

Next, we present the lazy variant of our method in Algorithm 9 which updates the parameters of the conditioning function only at the end of each task. The across-tasks regret bound for Algorithm 9 is given in the following theorem. See Appendix B.4 for the full proof.

Theorem 4.4. *Let Assumption 1 and 2 hold. Consider* $\mathbf{Z} = (Z_t)_{t=1}^T$, $Z_t = (x_{t,i}, y_{t,i})_{i=1}^n$ *deriving from* T *different tasks with associated side information* $\mathbf{S} = (s_t)_{t=1}^T$. *Let* $(w_{\theta_t,i})_{t=1,i=1}^{T,n}$ *be the iterates generated by Algorithm 9 over* $\mathbf{Z}$ *and* $\mathbf{S}$. *Then for any* $w_t \in \mathbb{R}^d$, $M \in \mathbb{R}^{d\times k}$ *and* $b \in \mathbb{R}^d$,

$$\sum_{t=1}^{T}\sum_{i=1}^{n} \ell_{t,i}\left(\langle x_{t,i}, w_{\theta_t,i}\rangle\right) - \ell_{t,i}\left(\langle x_{t,i}, w_t\rangle\right) \leq \sum_{t=1}^{T}\sum_{i=1}^{n} \left\langle g_{t,i}, w_{\theta_t,i} - w_t\right\rangle \leq A + B + C \tag{4.19}$$

where

$$\begin{aligned} A &= RL\left[eT + \left(2\sqrt{2}\mathrm{Var}(\tau_{M,b}) + \widehat{\mathrm{Var}}(\tau_{M,b})\right)\sqrt{n}T\right] \\ B &= RLK\left[\mathscr{E}n + \left(2\sqrt{2} + \Psi\left(\mathscr{E}^{-1}\left\|M\right\|_F T\right)\right)\left\|M\right\|_F n\sqrt{T}\right] \\ C &= RL\left[En + \left(2\sqrt{2} + \Psi\left(E^{-1}\left\|b\right\| T\right)\right)\left\|b\right\| n\sqrt{T}\right] \end{aligned} \tag{4.20}$$

and $\Psi(a) = \sqrt{\log(1+24a^2)}$.

Proof Sketch. The proof of the statement follows the same idea we use in the one of Proposition 4.3, which is to decompose the target linear regret into a sum of six linear regret terms w.r.t the coin betting

Algorithm 9 Parameter-Free Fine Tuning Method with Conditional Function, Lazy Version

Input $\mathbf{Z} = (Z_t)_{t=1}^T, Z_t = (z_i)_{i=1}^n = (x_{t,i}, y_{t,i})_{i=1}^n$, $\mathbf{S} = (S_t)_{t=1}^T$, $\Phi(\cdot)$, $e > 0, E > 0, \mathscr{E} > 0$, L, R and K as in Assumption 1, 2.

Initialize $\mathscr{B}_1 = 0, \mathscr{U}_1 = \mathscr{E}, \mathscr{P}_1 = \mathscr{B}_1\mathscr{U}_1, \mathscr{V}_1 = O_{d\times k} \in \mathcal{B}(0,1)$, $B_1 = 0, U_1 = E, P_1 = B_1U_1, V_1 = \mathbf{0} \in \mathcal{B}(0,1)$

For $t = 1, \ldots, T$

Meta-transition matrix update $M_t = \mathscr{P}_t\mathscr{V}_t$

Meta-shift update $b_t = P_tV_t$

Conditional meta-vector update $\theta_t = M_t\Phi(S_t) + b_t$

Set $b_{\theta_t,1} = 0, u_{\theta_t,1} = e, p_{\theta_t,1} = b_{\theta_t,1}u_{\theta_t,1}, v_{\theta_t,1} = 0 \in \mathcal{B}(0,1)$

For $i = 1, \ldots, n$

Define $k = k(t,i) = (t-1)n + i$

Within-vector update $w_{\theta_t,i} = p_{\theta_t,i}v_{\theta_t,i} + \theta_t$

Receive the datapoint $z_{t,i} = (x_{t,i}, y_{t,i})$

Compute $g_{t,i} = h_{t,i}x_{t,i}, h_{t,i} \in \partial\ell_{t,i}(\langle x_{t,i}, w_{\theta_t,i}\rangle)$

Define $\gamma_{t,i} = \frac{1}{LR}\sqrt{\frac{2}{i}}$

Direction update $v_{\theta_t,i+1} = \mathrm{proj}_{\mathcal{B}(0,1)}(v_{t,i} - \gamma_{t,i}g_{t,i})$

Define $u_{\theta_t,i+1} = u_{\theta_t,i} - \frac{1}{RL}\langle g_{t,i}, v_{\theta_t,i}\rangle p_{\theta_t,i}$

Define $b_{\theta_t,i+1} = \frac{1}{i}\big((i-1)b_{\theta_t,i} - \frac{1}{RL}\langle g_{t,i}, v_{\theta_t,i}\rangle\big)$

Magnitude update $p_{\theta_t,i+1} = b_{\theta_t,i+1}u_{\theta_t,i+1}$

End

Define $\zeta_t = \frac{1}{LRKn}\sqrt{\frac{2}{t}}$ and $G_t = \sum_{i=1}^n g_{t,i}$

Receive $G_t\Phi(S_t)^\top$

Direction update $\mathscr{V}_{t+1} = \mathrm{proj}_{\mathcal{B}(0,1)}(\mathscr{V}_t - \zeta_tG_t\Phi(S_t)^\top)$

Define $\eta_t = \frac{1}{LRn}\sqrt{\frac{2}{t}}$

Direction update $V_{t+1} = \mathrm{proj}_{\mathcal{B}(0,1)}(V_t - \eta_tG_t)$

Define $\mathscr{U}_{t+1} = \mathscr{U}_t - \frac{1}{RLKn}\big\langle G_t\Phi(S_t)^\top, \mathscr{V}_t\big\rangle_F\mathscr{P}_t$

Define $\mathscr{B}_{t+1} = \frac{1}{t}\big((t-1)\mathscr{B}_t - \frac{1}{RLKn}\big\langle G_t\Phi(S_t)^\top, \mathscr{V}_t\big\rangle_F\big)$

Magnitude update $\mathscr{P}_{t+1} = \mathscr{B}_{t+1}\mathscr{U}_{t+1}$

Define $U_{t+1} = U_t - \frac{1}{RLn}\big\langle G_t, V_t\big\rangle P_t$

Define $B_{t+1} = \frac{1}{t}\big((t-1)B_t - \frac{1}{RLn}\big\langle G_t, V_t\big\rangle\big)$

Magnitude update $P_{t+1} = B_{t+1}U_{t+1}$

End

Return $(w_{\theta_t,i})_{t=1,i=1}^{T,n}$, $(M_t)_{t=1}^T$ and $(b_t)_{t=1}^T$

algorithm and the online subgradient algorithm and bound these terms respectively. Finally, we combine the derived bounds together to obtain the desired statement. □

4.4 Statistical Setting

In the previous section, we have presented our method under the non-statistical setting where no assumptions are made on data. In this section, we show how our proposed online parameter-free fine tuning method can be adapted to the statistical setting. More specifically, we demonstrate how the aggressive variant can be applied to the statistical multi-task learning (MTL) setting and the lazy variant can be applied to the statistical meta-learning setting.

Deterministic Task (Statistical MTL Setting)

Below we analyse how Algorithm 8 can be adapted to learning deterministic heterogeneous sequence of tasks.

First of all, we briefly recall the deterministic task setting introduced in Chapter 2. Suppose we have within-task datasets $\mathbf{Z} = (Z_t)_{t=1}^T$ which are independently identically sampled from distributional tasks $(\mu_t)_{t=1}^T$ and their associated side information $\mathbf{S} = (s_t)_{t=1}^T$. For any task $t = 1, \dots, T$, we consider the average estimator $\bar{w}_t = \frac{1}{n}\sum_{i=1}^n w_{t,i}$ where $w_{t,i}$ are the iterates returned by Algorithm 8 w.r.t. the task t. The performance of Algorithm 8 in this case is evaluated by the multi-task risk, i.e.

$$\frac{1}{T}\sum_{t=1}^T \mathcal{R}_{\mu_t}(\bar{w}_t) - \frac{1}{T}\sum_{t=1}^T \mathcal{R}_{\mu_t}(w_{\mu_t}) \tag{2.15}$$

where the second term is the benchmark multi-task oracle of Algorithm 8. By applying the online-to-batch conversion results presented in Proposition 2.6, we obtain the bound on (2.15) in expectation w.r.t. the datasets $\mathbf{Z}$ as follows. The complete proof of Theorem 4.5 is reported in Appendix B.3.

Theorem 4.5 (Multi-Task Risk Bound for Algorithm 8)**.** *Let the same assumptions in Theorem 4.3 hold in the deterministic heterogeneous tasks setting. Then, for any $M \in \mathbb{R}^{d\times k}, b \in \mathbb{R}^d$, in the expectation w.r.t. the sampling of the datasets $\mathbf{Z} = (Z_t)_{t=1}^T$,*

$$\mathbb{E}_{\mathbf{Z}}\left[\frac{1}{T}\sum_{t=1}^T \mathcal{R}_{\mu_t}(\bar{w}_t)\right] - \frac{1}{T}\sum_{t=1}^T \mathcal{R}_{\mu_t}(w_{\mu_t}) \leq \frac{1}{nT}(A + B + C) \tag{4.21}$$

where

$$
\begin{aligned}
A &= RL\left[eT + \left(2\sqrt{2}\mathrm{Var}_{\mathrm{MTL}}(\tau_{M,b}) + \widehat{\mathrm{Var}}_{\mathrm{MTL}}(\tau_{M,b})\right)\sqrt{nT}\right] \\
B &= RLK\left[\mathscr{E} + \left(2\sqrt{2} + \Psi\left(\mathscr{E}^{-1}\left\|M\right\|_F nT\right)\right)\left\|M\right\|_F \sqrt{nT}\right] \\
C &= RL\left[E + \left(2\sqrt{2} + \Psi\left(E^{-1}\left\|b\right\| nT\right)\right)\left\|b\right\| \sqrt{nT}\right]
\end{aligned}
\tag{4.22}
$$

with

$$
\begin{aligned}
\mathrm{Var}_{\mathrm{MTL}}(\tau_{M,b}) &= \frac{1}{T}\sum_{t=1}^{T}\left\|w_{\mu_t} - M\Phi(s_t) - b\right\|, \\
\widehat{\mathrm{Var}}_{\mathrm{MTL}}(\tau_{M,b}) &= \frac{1}{T}\sum_{t=1}^{T}\Psi\left(e^{-1}\left\|w_{\mu_t} - M\Phi(s_t) - b\right\| n\right)\left\|w_{\mu_t} - M\Phi(s_t) - b\right\|.
\end{aligned}
\tag{4.23}
$$

and $\Psi(a) = \sqrt{\log(1+24a^2)}$.

Proof Sketch. The proof of the statement follows the same strategy we used in the one of Proposition 2.5. We simply apply the across-tasks regret bound obtained in Theorem 4.3 to the right side of the online-to-batch conversion results. □

Random Task (Statistical Meta-Learning Setting)

Next we show that Algorithm 9 can be converted into a parameter-free statistical meta-learning method to learn heterogeneous random tasks and provide statistical performance guarantees for it.

We recall that in the statistical meta-learning setting, we have within-task datasets $(Z_t)_{t=1}^T$ which are sampled independently and identically from distribution $(\mu_t)_{t=1}^T$ correspondingly. The tasks $(\mu_t)_{t=1}^T$ are i.i.d. samples with associated side information s_t from a meta-distribution ρ, namely $(\mu_t, s_t) \sim \rho$. Similarly to the discussion in Section 2.2.4, we consider estimator $\bar{w}_{\tau_{\hat{t}}}(Z) = \frac{1}{n}\sum_{i=1}^n w_{\tau_{\hat{t}},i}(Z)$ which is the average of iterates returned by Algorithm 9 when applied to a new test dataset Z with sampled from a new task μ and its associated side information s. The conditioning function $\tau_{\hat{t}}$ adopted is a function uniformly sampled from the conditioning functions inferred by Algorithm 9 applied to training datasets $\mathbf{Z} = (Z_t)_{t=1}^T$ with side information $\mathbf{S} = (s_t)_{t=1}^T$. We measure the performance of the estimator $\bar{w}_{\tau_{\hat{t}}}(Z)$ by

$$
\mathbb{E}_{(\mu,s)\sim\rho}\mathbb{E}_{Z\sim\mu^n}\mathcal{R}_\mu(\bar{w}_{\tau_{\hat{t}}}(Z)) - \mathbb{E}_{\mu\sim\rho}\mathcal{R}_\mu(w_\mu) \tag{4.24}
$$

where the first term is the transfer risk of the estimator defined in the conditional approach and the second term is the oracle benchmark. The bound on the transfer risk for Algorithm 9 w.r.t. the meta-learning oracle $\mathbb{E}_{\mu\sim\rho}\mathcal{R}_\mu(w_\mu)$ is provided in the following theorem. Proof is attached in Appendix B.5

Theorem 4.6 (Transfer Risk Bound for Algorithm 9)**.** *Let the same assumptions in Theorem 4.4 hold in the statistical meta-learning setting. Then, for any $M \in \mathbb{R}^{d\times k}$ and $b \in \mathbb{R}^d$, the transfer risk bound for Algorithm 9 in expectation w.r.t. the sampling of the datasets $\mathbf{Z}$ and the uniform sampling of $\hat{t} \sim \mathcal{U}(T)$ is expressed as*

$$\mathbb{E}_{\hat{t}\sim\mathcal{U}(T)}\,\mathbb{E}_{\mathbf{Z}}\left[\mathbb{E}_{(\mu,s)\sim\rho}\mathbb{E}_{Z\sim\mu^n}\,\mathcal{R}_\mu(\bar{w}_{\tau_{\hat{t}}}(Z))\right] - \mathbb{E}_{\mu\sim\rho}\,\mathcal{R}_\mu(w_\mu) \leq \frac{1}{nT}(A+B+C) \tag{4.25}$$

where

$$\begin{aligned} A &= RL\left[eT + \left(2\sqrt{2}\mathrm{Var}_{\mathrm{META}}(\tau_{M,b}) + \widehat{\mathrm{Var}}_{\mathrm{META}}(\tau_{M,b})\right)\sqrt{n}T\right] \\ B &= RLK\left[\mathscr{E}n + \left(2\sqrt{2} + \Psi\left(\mathscr{E}^{-1}\left\|M\right\|_F T\right)\right)\left\|M\right\|_F n\sqrt{T}\right] \\ C &= RL\left[En + \left(2\sqrt{2} + \Psi\left(E^{-1}\left\|b\right\| T\right)\right)\left\|b\right\| n\sqrt{T}\right] \end{aligned} \tag{4.26}$$

with

$$\begin{aligned} \mathrm{Var}_{\mathrm{META}}(\tau_{M,b}) &= \mathbb{E}_{(\mu,s)\sim\rho}\left\|w_\mu - M\Phi(s) - b\right\|, \\ \widehat{\mathrm{Var}}_{\mathrm{META}}(\tau_{M,b}) &= \mathbb{E}_{(\mu,s)\sim\rho}\Big[\Psi\left(e^{-1}\left\|w_\mu - M\Phi(s) - b\right\| n\right)\left\|w_{\mu_t} - M\Phi(s) - b\right\|\Big], \end{aligned} \tag{4.27}$$

and $\Psi(a) = \sqrt{\log(1+24a^2)}$.

Proof Sketch. To obtain the desired statement, we have to use an adapted version online-to-batch conversion result which we will state in detail in Appendix B.5. With the conversion result, we then only need to apply the across-tasks regret bound for the lazy variant obtained in Theorem 4.4. □

Chapter 5

Experiment

We present our empirical results in this chapter. In our experiments, we aim to demonstrate the efficacy of our conditional parameter-free fine tuning method in Algorithm 8 and 9 over its unconditional counterpart in [4] from the following perspectives: (1) Can our method learn homogeneous tasks as well as its unconditional counterpart? (2) Does our method outperform its unconditional counterpart when learning heterogeneous tasks?

We implemented the method in Python 3.7.1 and all experiments were performed on a 2.3 GHz Intel Core i5 Macbook Pro, 16 GB RAM, running OS X 10.14.6. The code is available online [1].

5.1 Synthetic Data

We consider two synthetic tasks examples : (1) cluster example and (2) circle example, comparing the performance of the conditional parameter-free aggressive variant in Algorithm 8, lazy variant in Algorithm 9 in our work and their counterparts, i.e. unconditional parameter-free aggressive variant in [**Algorithm 4**, 4] and lazy variant in [**Algorithm 5**, 4]. We also add to the comparison the ITL method (setting $\tau \equiv 0 \in \mathbb{R}^d$).

In the experiments below, we noticed that the variants of our methods estimating the magnitude by the refined coin betting algorithm in [**Algorithm 1**, 2] returned more readable plots w.r.t. the ones proposed in our methods using the KT algorithm in Algorithm 6. Therefore, we report below the results obtained using this more refined variant.

5.1.1 Cluster Examples

Data Generation Following the data generation process described in [3], our cluster dataset consists of a uniform mixture of m clusters (environments), i.e. $\rho_{\mathcal{M}} = \frac{1}{m}\sum_{i=1}^{m} \rho_{\mathcal{M}}^{(m)}$, and for each cluster

[1] Code for the experiments is at https://github.com/younei/paramfree-FT-hetero

$i = 1, \ldots, m$, a regression task $\mu \sim \rho_{\mathcal{M}}^{(i)}$ is generated such that:

1) the ground truth weight vector w_μ is sampled from a Gaussian distribution with mean $w(i) \in \mathbb{R}^d$ associated with cluster i and identity covariance I, i.e. $w_\mu \sim \mathcal{N}(w(i), I)$,

2) the corresponding partially observed dataset $(x_i, y_i)_{i=1}^n$ is constructed by first sampling the inputs $(x_i)_{i=1}^n$ from projected Gaussian distribution on a unit sphere with centroid (mean) $x(i)$ associated with task i, i.e. $x_i \sim \mathcal{PN}(x(i), I)$ (see [12] for more detailed formulations of projected Gaussian distribution) and then generating output labels $(y_i)_{i=1}^n$ according to the equation $y_i = \langle x_i, w_\mu \rangle + \epsilon_i$, where the noise ϵ_i is sampled from $\mathcal{N}(0, \sigma^2 I)$, with σ chosen to have singal-to-noise ratio equals to 1,

3) the associated side information s is an n i.i.d. sample from the projected Gaussian distribution, which is taken as the inputs $s = X = (x_i)_{i=1}^n$ in our experiment.

We considered three different cluster datasets to test the performance of the methods. In all three datasets, we sampled T regression tasks from a mixture of clusters, with corresponding dataset $z_t = (x_{t,i}, y_{t,i})_{i=1}^{n^{\text{tot}}}$ for each task $t \in \{1, \ldots, T\}$. In order to train and test the methods, we split each dataset z_t into $z_t^{\text{tr}} = (x_{t,i}^{\text{tr}}, y_{t,i}^{\text{tr}})_{i=1}^{n^{\text{tr}}}$ for training and $z_t^{\text{te}} = (x_{t,i}^{\text{te}}, y_{t,i}^{\text{te}})_{i=1}^{n^{\text{te}}}$ for testing, where $n^{\text{tot}} = 20$, $n^{\text{tr}} = 50\%\, n^{\text{tot}} = 10$ and $n^{\text{te}} = 50\%\, n^{\text{tot}} = 10$. The dimension of inputs is $d = 20$.

One Cluster To investigate whether our proposed methods can learn homogeneous tasks no worse than their unconditional counterparts, we generate $T = 350$ tasks of one cluster dataset ($m = 1$) following the aforementioned procedure, taking $w(1) = \mathbf{4} \in \mathbb{R}^d$ (the vector in $\mathbb{R}^d$ with all components 4) and $x(1) = \mathbf{1} \in \mathbb{R}^d$. The top panel of Figure 5.1 shows the average across-tasks cumulative error for all the methods w.r.t. an increasing increasing number of datapoints. The bottom panel of Figure 5.1 shows the average statistical multi-task test errors for an increasing number of tasks. We used the absolute loss (1-Lipschitz function) to measure the error and used the mean feature map $\Phi : \bigcup_{n \in \mathbb{N}} \mathcal{X}^n \to \mathbb{R}^d$ defined as $\Phi(X) = \frac{1}{n} \sum_{i=1}^n x_i$ for conditional methods. We also set initial wealth as $e = 1$, $E = 1$, $\mathscr{E} = 1$, and deterministically set input bound as $R = 6$ and feature mapping bound as $K = 6$. In the plots, both unconditional methods and conditional methods converge to the oracle (best performance on the dataset in hindsight) as the number of datapoints/tasks increases, confirming that our conditional methods performs as well as their unconditional counterparts when learning homogeneous tasks. We also see that the aggressive variant of the methods converges to the oracle faster than the lazy variants, which is inline with the theory.

Two Clusters To test the performance of our methods over their counterparts for heterogeneous tasks, we consider two examples of two-clusters dataset ($m = 2$). In the first example, we create an two-cluster environment with $w_\rho = 4$ by taking $w(1) = \mathbf{8} \in \mathbb{R}^d$, $x(1) = \mathbf{1} \in \mathbb{R}^d$ for one cluster and

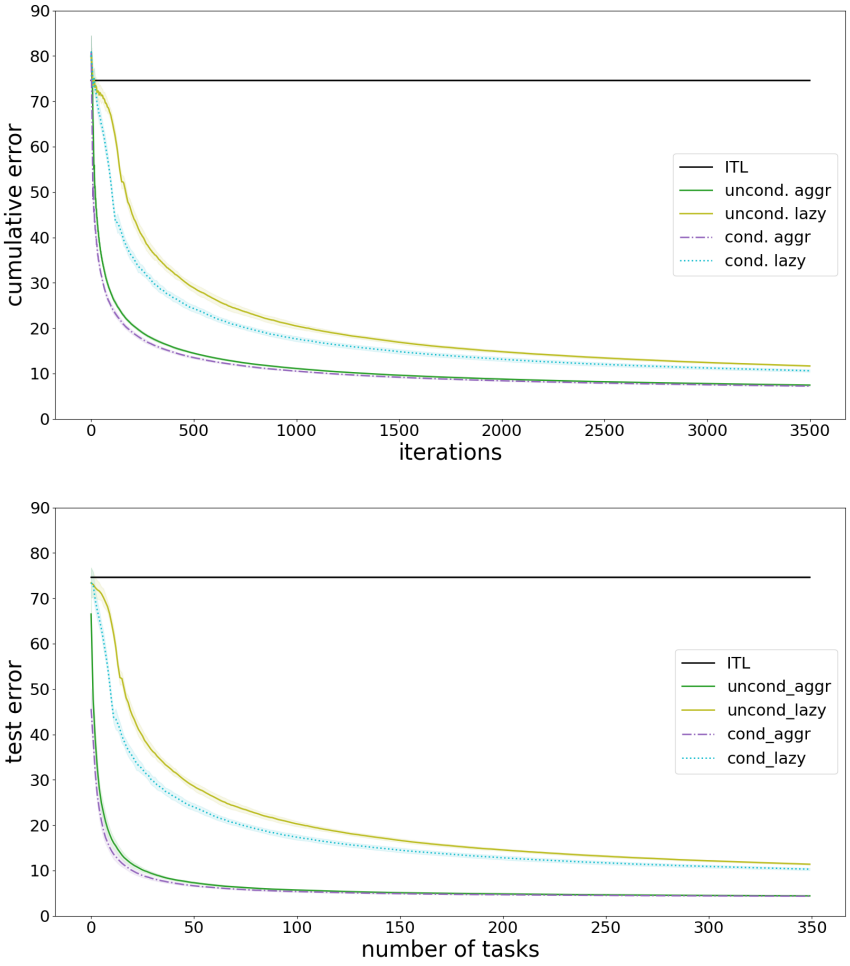

Figure 5.1: (top) Average across-tasks cumulative error (over 10 seeds) of different methods w.r.t. an increasing number of datapoints on one cluster data. (bottom) Average multi-task test error (over 10 seeds) w.r.t. an increasing number of tasks on one cluster data.

setting $w(2) = \mathbf{0} \in \mathbb{R}^d$, $x(2) = -x(1) \in \mathbb{R}^d$ for the other cluster. We sampled $T = 350$ tasks for the first example. In the second example, we create an two-cluster environment with $w_\rho = 0$ by taking $w(1) = \mathbf{4} \in \mathbb{R}^d, x(1) = \mathbf{1} \in \mathbb{R}^d$ for one cluster and setting $w(2) = -w(1) \in \mathbb{R}^d, x(2) = -x(1) \in \mathbb{R}^d$ for the other cluster. We sampled $T = 350$ tasks for the second example. The top and bottom panels of Figure 5.2 present the the average across-tasks cumulative error for all methods on the left side and the average statistical multi-task test errors for all methods on the right side, of the two examples respectively. Same as in the one cluster experiment, we used the absolute loss to measure error and the mean feature map for conditional methods. The initial wealth is set as $e = 1$, $E = 1$, $\mathscr{E} = 1$, and input bound is $R = 6$ and feature mapping bound is $K = 6$. The plots imply that whilst both the conditional and unconditional methods outperform the ITL method substantially, our conditional methods yield much better performance than their counterparts. Moreover, we see that the aggressive variant of our method shows faster rates w.r.t. its lazy counterpart, which is coherent with the derived regret bounds.

5.1.2 Circle Examples

Data Generation We follow the procedure described in [3] to create circle regression tasks. For each task μ, we first sample the associated side information s from a uniform distribution $s \sim \mathcal{U}(0, 1)$. We define a zero-centered circle with radius $r > 0$ as $h(s) = r(\cos(2\pi s), \sin(2\pi s), 0, \ldots, 0)^\top \in \mathbb{R}^d$, and generate a vector $h(s)$ on the circle with dimension $d = 20$. Then the true underlying weight vector w_μ for task μ is sampled from $\mathcal{N}(h(s), I)$. For the corresponding dataset $(x_{t,i}, y_{t,i})_{i=1}^n$, the inputs are sampled from a projected Gaussian distribution on a unit sphere with centroid $x(i)$ and the output labels are created by using the same regression equation introduced in the cluster data generation procedure.

Circle We sample $T = 350$ circle regression tasks to examine the performance of the methods, with each task having training datapoints $n^{\text{tr}} = 10$ and test datapoints $n^{\text{te}} = 10$. We apply two different feature maps to our conditional methods: (1) the ground truth feature map $\Phi(s) = (\cos(2\pi s), \sin(2\pi s))$ and (2) the Fourier random feature map which mimics a Gaussian distribution [17]. The random feature map is defined as $\Phi(X) = \frac{1}{n}\sum_{i=1}^n \phi(x_i)$ where $\phi(\cdot)$ is constructed as

$$\phi(x_i) = \sqrt{\frac{2}{k}} \cos(Ux_i + v) \in \mathbb{R}^k.$$

In ϕ, vector $v \in \mathbb{R}^k$ is sampled from the uniform distribution over $[0, 2\pi]^k$ and matrix $U \in \mathbb{R}^{k\times d}$ is sampled from the Gaussian distribution $\mathcal{N}(0, \sigma I)$ with constant $\sigma \in \mathbb{R}$. $\cos(\cdot)$ operation is applied component-wise to the vector $Ux_i + v$. The input bound is derived $R = 6$ and the feature bound for the ground truth feature map is $K = 1$ and for the Fourier random feature map is $K = 3.5$.

The top panel of Figure 5.4 shows a comparison of the average across-tasks cumulative error among

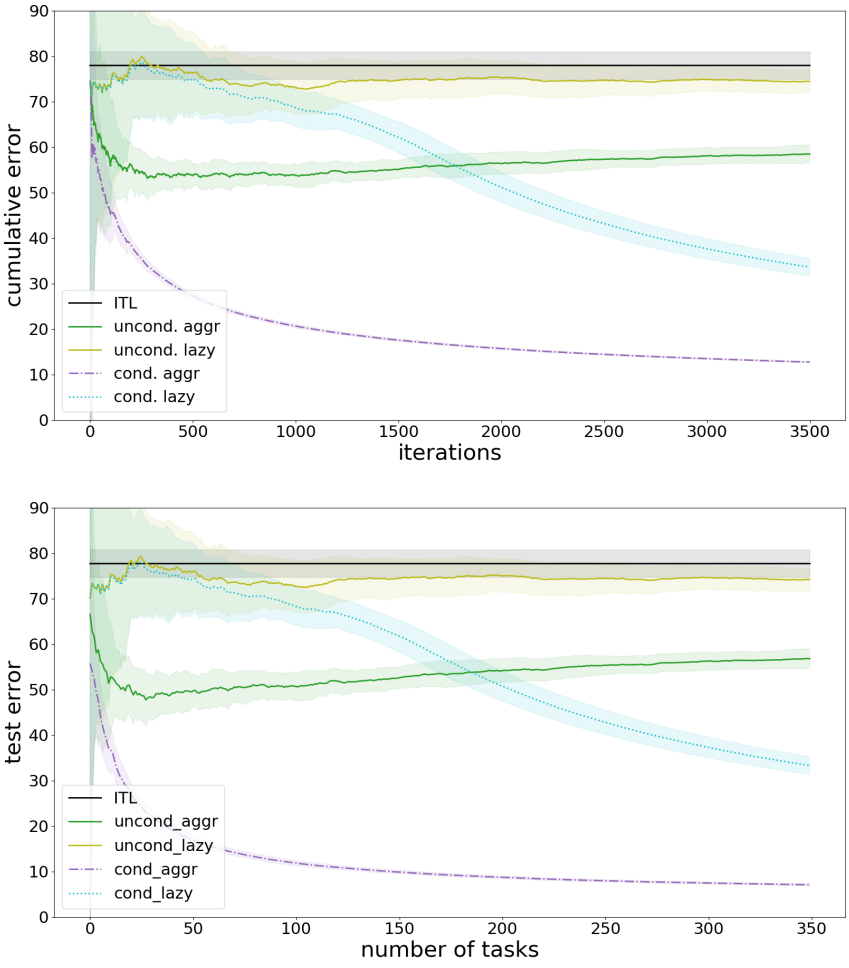

Figure 5.2: (top) Average across-tasks cumulative error (over 20 seeds) of different methods w.r.t. an increasing number of datapoints on two clusters data with $w_\rho = 4$. (bottom) Average multi-task test error (over 20 seeds) w.r.t. an increasing number of tasks on two clusters data with $w_\rho = 4$.

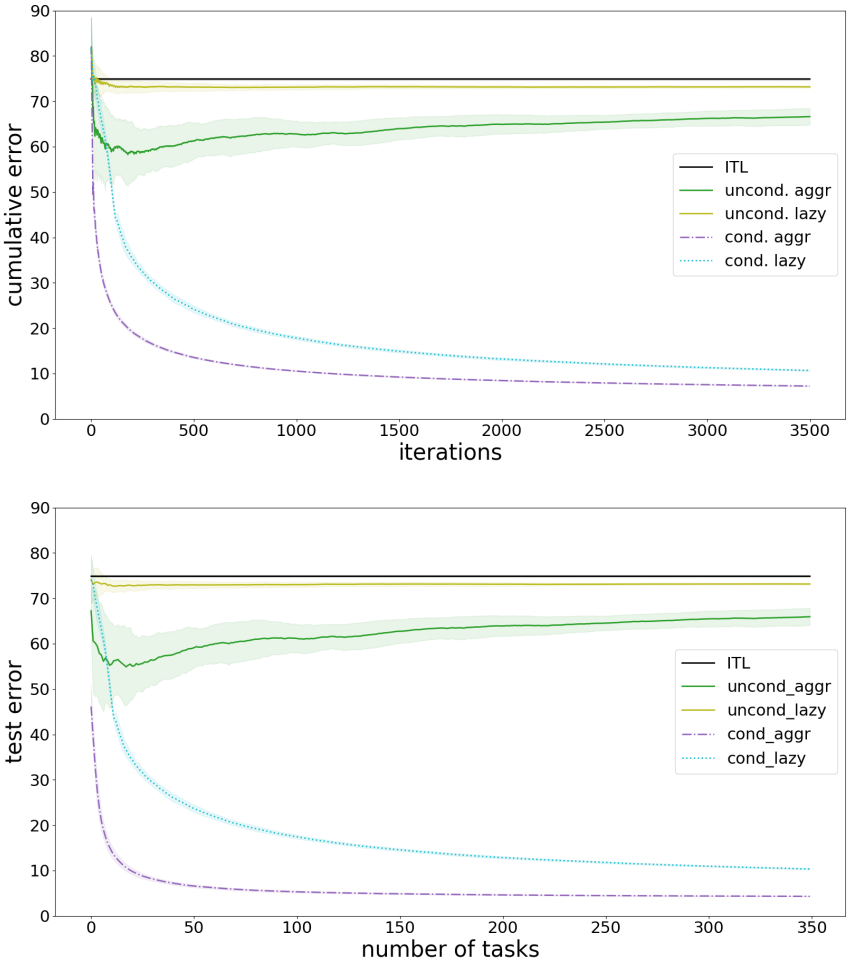

Figure 5.3: (top) Average across-tasks cumulative error (over 20 seeds) of different methods w.r.t. an increasing number of datapoints on two clusters data with $w_\rho = 0$. (bottom) Average multi-task test error (over 20 seeds) w.r.t. an increasing number of tasks on two clusters data with $w_\rho = 0$.

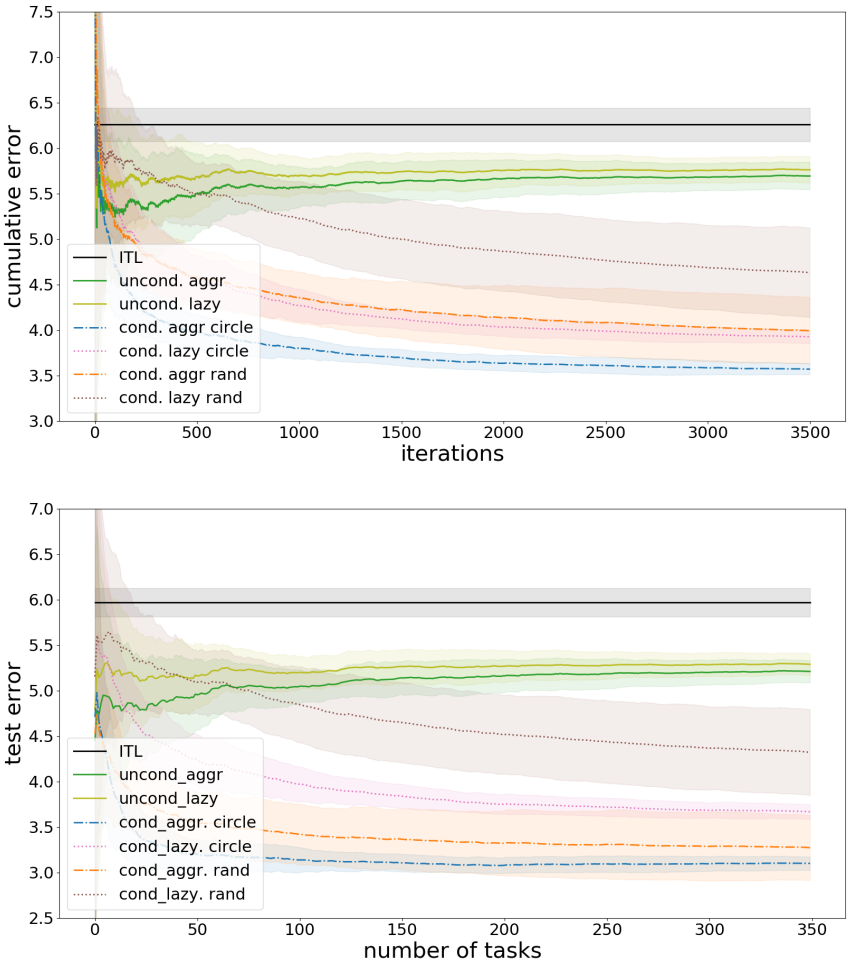

Figure 5.4: (top) Average across-tasks cumulative error (over 30 seeds) of different methods w.r.t. an increasing number of datapoints on circle data. (bottom) Average multi-task test error (over 30 seeds) w.r.t. an increasing number of tasks on circle data.

all methods and the bottom panel of Figure 5.4 shows the average statistical multi-task test errors. The plots reveal that our conditional methods show substantial benefits w.r.t. their unconditional counterparts, which perform slightly better than ITL. The aggressive variant of our method again demonstrate faster convergence rate than the lazy variant, verifying the theoretical findings in the main body. We also see that the conditional methods using the Fourier random feature map gradually approach the ones using the true feature map.

Part III

Conclusions

Chapter 6

Conclusions

In this work, we propose an online parameter-free fine tuning method that learns a conditioning function shared among a growing sequence of low-variance tasks. Our methods empirically show significant improvement over their unconditional counterparts in heterogeneous environments. The method is originally presented in the non-statistical setting and can be of two variants - the aggressive one which has faster convergence rates and can be converted into a statistical MTL method; the lazy one which has standard rates but can be converted into a statistical meta-learning method. In the future, it would be interesting to study different inner algorithms for our parameter-free knowledge-transfer method.

Appendix A

Online subgradient descent in matrix form

In Algorithm 8 and 9, we use the matrix form online subgradient descent algorithm to estimate the direction $\mathscr{P}$ of transition matrix M. Below we report the matrix form of online subgradient descent algorithm and show that the regret bound in Proposition 3.1 holds for the matrix form as well. The algorithm takes input a convex, closed and non-empty set $\mathcal{C} \subset \mathbb{R}^{d\times k}$ with diameter

$$\operatorname{diam}(\mathcal{C}) = \sup_{M,M'\in\mathcal{C}} \|M - M'\|_F .$$

At each iteration i, the algorithm receives a matrix $G_i \in \mathbb{R}^{d\times k}$ with Frobenius norm $\|G_i\|_F \leq C$ for some $C > 0$.

Algorithm 10 Online Projected Subgradient Algorithm, Matrix Form

Input $\mathcal{C} \subset \mathbb{R}^{d\times k}$, $(G_i)_{i=1}^n$, $G_i \in \mathbb{R}^{d\times k}$, $\|G_i\|_F \leq C$
Initialize $M_1 \in \mathcal{C}$
For $i = 1, \ldots, n$
  Receive G_i
  Define $\gamma_i = \frac{\operatorname{diam}(\mathcal{C})}{C\sqrt{2i}}$
  Update $M_{i+1} = \operatorname{proj}_{\mathcal{C}}(M_i - \gamma_i G_i) = \operatorname{argmin}_{V\in\mathcal{C}} \|M_i - \gamma_i G_i - V\|_F$
End
Return $(M_i)_{i=1}^n$

The linear regret of Algorithm 10 has the same bound as of Algorithm 5 (vector form), described in the following proposition.

Proposition A.1 (Linear Regret Bound for Algorithm 10)**.** *The iterates* $(M_i)_{i=1}^n$ *returned by Algorithm*

10 satisfy the following linear regret bound w.r.t. a competitor matrix $M \in \mathcal{C}$,

$$\sum_{i=1}^{n} \langle G_i, M_i - M \rangle_F \leq C\sqrt{2}\,\mathrm{diam}(\mathcal{C})\sqrt{n}.$$

Proof. The proof is adapted from Theorem 3.1 in [COO]. We first drive the following inequality using the update rule for M_{i+1} and the Pythagorean theorem,

$$\begin{aligned}
\|M_{i+1} - M\|_F^2 &= \|\mathrm{proj}_{\mathcal{C}}\left(M_i - \gamma_i G_i\right) - M\|_F^2 \\
&\leq \|M_i - \gamma_i G_i - M\|_F^2 \\
&= \|M_i - M\|_F^2 + \|\gamma_i G_i\|_F^2 - 2\langle M_i - M, \gamma_i G_i \rangle_F
\end{aligned}$$

where the second step follows from the matrix version of Pythagorean theorem using Frobenius norm and the last step follows from the fact that

$$\|A + B\|_F^2 = \|A\|_F^2 + \|B\|_F^2 + \langle A, B \rangle_F \text{ , for all matrices } A, B \in \mathbb{R}^{d \times k}.$$

Rewriting the above inequality, we get

$$\langle M_i - M, G_i \rangle_F \leq \frac{\|M_i - M\|_F^2 - \|M_{i+1} - M\|_F^2}{2\gamma_i} + \frac{\gamma_i}{2}\|G_i\|_F^2 .$$

Summing over the iterations,

$$\begin{aligned}
\sum_{i=1}^{n} \langle M_i - M, G_i \rangle_F &\leq \sum_{i=1}^{n} \frac{\|M_i - M\|_F^2 - \|M_{i+1} - M\|_F^2}{2\gamma_i} + \frac{\gamma_i}{2}\|G_i\|_F^2 \\
&\leq \frac{1}{2}\sum_{i=1}^{n} \|M_i - M\|_F^2 \left(\frac{1}{\gamma_i} - \frac{1}{\gamma_{i-1}}\right) + \frac{\|G_i\|_F^2}{2}\sum_{i=1}^{n}\gamma_i \\
&\leq \frac{1}{2}\,\mathrm{diam}(\mathcal{C})^2 \sum_{i=1}^{n} \left(\frac{1}{\gamma_i} - \frac{1}{\gamma_{i-1}}\right) + \frac{C^2}{2}\sum_{i=1}^{n}\gamma_i \\
&\leq \frac{1}{2}\,\mathrm{diam}(\mathcal{C})^2 \frac{1}{\gamma_n} + \frac{C^2}{2}\sum_{i=1}^{n}\gamma_i
\end{aligned}$$

substituting $\gamma_i = \frac{\operatorname{diam}(\mathcal{B})}{C\sqrt{2i}}$ and using the fact that $\sum_{i=1}^{n} \frac{1}{\sqrt{i}} \leq 2\sqrt{n}$, we obtain

$$\begin{aligned}\sum_{i=1}^{n} \langle M_i - M, G_i \rangle_F &\leq \frac{1}{2} \operatorname{diam}(\mathcal{C}) C\sqrt{2}\sqrt{n} + \frac{C^2}{2} \frac{\operatorname{diam}(\mathcal{C})}{C\sqrt{2}} 2\sqrt{n} \\ &= C\sqrt{2} \operatorname{diam}(\mathcal{C})\sqrt{n}.\end{aligned}$$

□

Appendix B

Proofs of the statements in Chapter 4

B.1 Proof of Proposition 4.1

Proposition (Single-Task Regret Bound for Algorithm 7)**.** *Let Assumption 1 hold and let* $(w_i)_{i=1}^n$ *be the iterates generated by Algorithm 7 with conditional transfer-parameter* $\tau(s)$*. Then, for any* $w \in \mathbb{R}^d$*,*

$$\begin{aligned}&\sum_{i=1}^{n} \ell_i(\langle x_i, w_i\rangle) - \ell_i(\langle x_i, w\rangle) \leq \sum_{i=1}^{n} \langle g_i, w_i - w\rangle \\ &\leq RL\Big[e + \Big(2\sqrt{2} + \Psi\left(e^{-1}\left\|w - M\Phi(s) - b\right\| n\right)\Big)\left\|w - M\Phi(s) - b\right\| \sqrt{n}\Big] \end{aligned} \tag{B.1}$$

where $\Psi(a) = \sqrt{\log(1 + 24a^2)}$.

Proof. The proof follows a similar strategy to that of Proposition 3 in [4]. The key idea is to first decompose the linear regret of the method into the sum of linear regret of coin betting algorithm 6 and linear regret of online subgradient descent algorithm 5, and then use their bounds to derive the desired bound.

The first inequality follows from the definition of subgradient. Since the linear prediction function $\langle x_i, \cdot\rangle$ is convex (also simultaneously concave) and loss function $\ell_i(\cdot)$ is convex (see Assumption 1), the composition $\ell_i(\langle x_i, \cdot\rangle)$ is a convex function. The quantity g_i is constructed as the subdifferential of $\ell_i(\langle x_i, \cdot\rangle)$ at w_i, i.e. $g_i \in \partial\ell_i(\langle x_i, \cdot\rangle)(w_i)$, and hence by definition, g_i satisfies

$$\ell_i(\langle x_i, w_i\rangle) - \ell_i(\langle x_i, w\rangle) \leq \langle g_i, w_i - w\rangle, \ \forall w \in \mathbb{R}^d.$$

Summing over datapoints, we get the first inequality. Then, rewriting w_i and w using $w_i = p_i v_i + \tau_{M,b}(s)$ and $w = pv + \tau_{M,b}(s)$ respectively, we can decompose $\sum_{i=1}^n \langle g_i, w_i - w\rangle$ into the sum of term (1) and term (2) as follows.

$$
\begin{aligned}
\sum_{i=1}^{n}\langle g_i, w_i - w\rangle &= \sum_{i=1}^{n}\langle g_i, p_i v_i + \tau_{M,b}(s) - (pv + \tau_{M,b}(s))\rangle \\
&= \sum_{i=1}^{n}\langle g_i, p_i v_i - pv\rangle \\
&= \sum_{i=1}^{n}\langle g_i, p_i v_i - pv\rangle \pm \langle g_i, pv_i\rangle \\
&= \sum_{i=1}^{n} p_i\langle g_i, v_i\rangle - p\langle g_i, v_i\rangle + p\langle g_i, v_i\rangle - p\langle g_i, v\rangle \\
&= \underbrace{\sum_{i=1}^{n}\langle g_i, v_i\rangle(p_i - p)}_{(1)} + \underbrace{p\sum_{i=1}^{n}\langle g_i, v_i - v\rangle}_{(2)}.
\end{aligned}
$$

Next, we bound the two terms (1) and (2) above to obtain the bound on $\sum_{i=1}^{n}\langle g_i, w_i - w\rangle$. We see that term (1) is equivalent to the linear regret of $(p_i)_{i=1}^{n}$ returned by coin-betting algorithm 6 when applied to estimate the magnitude p_i of vector $w_i - \tau_{M,b}(s)$ in Algorithm 7. In this case, the coin-betting algorithm receives scalars $(\langle g_i, v_i\rangle)_{i=1}^{n}$ which are bounded by

$$
\begin{aligned}
|\langle g_i, v_i\rangle| &\le \|g_i\|\,\|v_i\| \\
&= |h_i|\,\|x_i\| \cdot \|v_i\| \\
&\le LR \cdot 1 = RL
\end{aligned}
\tag{B.2}
$$

where the first inequality of (B.2) follows from the Cauchy-Schwarz inequality and the second equality follows from the definition $g_i = h_i x_i$. As $h_i \in \partial\ell_i(\langle x_i, w_i\rangle)$, following Lemma 14.7 in [18] and the assumption that loss function $\ell_i(\cdot)$ is L-Lipschitz, we have that $|h_i| \le L$. Also, the bounded input assumption $\mathcal{X} \subseteq \mathcal{B}(0, R)$ suggests that $\|x_i\| \le R$ and by construction $\|v_i\| \le 1$, which imply the last inequality.

Recalling the initial wealth $e > 0$ used in the algorithm, by Proposition 3.2, we have the following bound on term (1)

$$
\sum_{i=1}^{n}\langle g_i, v_i\rangle(p_i - p) \le RL\left[e + \Psi\left(e^{-1}|p|n\right)|p|\sqrt{n}\right] \tag{B.3}
$$

where $p = \|w - \tau_{M,b}(s)\|$.

We also see that term (2) is equivalent to the linear regret of $(v_i)_{i=1}^{n}$ generated by online subgradient

descent algorithm 5 multiplied by p, when applied to estimate the direction v_i of vector $w_i - \tau_{M,b}(s)$ in Algorithm 7. The subgradient descent algorithm receives $(g_i)_{i=1}^n$ which is bounded as $\|g_i\| \leq RL$ as explained. As a result, by Proposition 3.1, we have linear regret bound

$$\sum_{i=1}^{n} \langle g_i, v_i - v \rangle \leq RL2\sqrt{2}\sqrt{n}.$$

Multiplying p on both sides, we get

$$p\sum_{i=1}^{n} \langle g_i, v_i - v \rangle \leq RL2\sqrt{2}p\sqrt{n}. \tag{B.4}$$

Finally, substituting (B.3) and (B.4) into term (1) and term (2), we obtain the desired linear regret bound on $(w_i)_{i=1}^n$ returned by Algorithm 7. □

B.2 Proof of Theorem 4.3

Theorem (Across-Tasks Regret Bound for Algorithm 8)**.** *Let Assumption 1 and 2 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 with associated T datasets of side information $\mathbf{S} = (s_t)_{t=1}^T$ w.r.t. each task. Let $(w_{t,i})_{t=1,i=1}^{T,n}$ be the iterates generated by Algorithm 8 over datasets $\mathbf{Z}$ and $\mathbf{S}$. Then, for any sequence $(w_t)_{t=1}^T$, $w_t \in \mathbb{R}^d$ and any $M \in \mathbb{R}^{d\times k}$, $b \in \mathbb{R}^d$,*

$$\sum_{t=1}^{T}\sum_{i=1}^{n} \ell_{t,i}\left(\langle x_{t,i}, w_{t,i}\rangle\right) - \ell_{t,i}\left(\langle x_{t,i}, w_t\rangle\right) \leq \sum_{t=1}^{T}\sum_{i=1}^{n} \left\langle g_{t,i}, w_{t,i} - w_t \right\rangle \leq A + B + C \tag{B.5}$$

where

$$\begin{aligned}
A &= RL\left[eT + \left(2\sqrt{2}\mathrm{Var}(\tau_{M,b}) + \widehat{\mathrm{Var}}(\tau_{M,b})\right)\sqrt{n}T\right] \\
B &= RLK\left[\mathscr{E} + \left(2\sqrt{2} + \Psi\left(\mathscr{E}^{-1}\|M\|_F nT\right)\right)\|M\|_F\sqrt{nT}\right] \\
C &= RL\left[E + \left(2\sqrt{2} + \Psi\left(E^{-1}\|b\| nT\right)\right)\|b\|\sqrt{nT}\right]
\end{aligned} \tag{B.6}$$

with

$$\begin{aligned}\mathrm{Var}(\tau_{M,b}) &= \frac{1}{T}\sum_{t=1}^{T} \|w_t - M\Phi(s_t) - b\|\,,\\ \widehat{\mathrm{Var}}(\tau_{M,b}) &= \frac{1}{T}\sum_{t=1}^{T} \Psi\left(e^{-1}\,\|w_t - M\Phi(s_t) - b\|\,n\right)\|w_t - M\Phi(s_t) - b\|\,.\end{aligned} \tag{B.7}$$

and $\Psi(a) = \sqrt{\log(1+24a^2)}$.

Proof. The idea of the proof, similar to that of Proposition 4.1, is to decompose the linear regret of weights returned by the method into the sum of linear regret of coin betting algorithm and linear regret of online subgradient descent algorithm associated w.r.t. the three quantities we estimate in Algorithm 8, and use their bounds to derive the desired statement.

The first inequality follows from the convexity of $\ell_{t,i}(\langle x_{t,i}, \cdot\rangle)$ and the fact $g_{t,i} \in \partial\ell_{t,i}(\langle x_{t,i}, \cdot\rangle)(w_{t,i})$. Next, we rewrite the linear regret of $w_{t,i}$ using the definition $w_t = p_t v_t + M\Phi(s_t) + b$ in (4.10) as

$$\sum_{t=1}^{T}\sum_{i=1}^{n}\left\langle g_{t,i}, w_{t,i} - w_t\right\rangle = \sum_{t=1}^{T}\sum_{i=1}^{n}\left\langle g_{t,i}, p_{t,i}v_{t,i} + M_{k(t,i)}\Phi(s_t) + b_{k(t,i)} - (p_t v_t + M\Phi(s_t) + b)\right\rangle$$

we then separate the linear regret into three parts,

$$\begin{aligned}
&= \sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, p_{t,i}v_{t,i} - p_t v_t\rangle + \left\langle g_{t,i}, M_{k(t,i)}\Phi(s_t) - M\Phi(s_t)\right\rangle\\
&\quad + \langle g_{t,i}, b_{k(t,i)} - b\rangle\\
&= \sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, p_{t,i}v_{t,i} - p_t v_t\rangle + \mathrm{Tr}\Big(g_{t,i}\Phi(s_t)^\top\left(M_{k(t,i)} - M\right)^\top\Big)\\
&\quad + \langle g_{t,i}, b_{k(t,i)} - b\rangle\\
&= \sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, p_{t,i}v_{t,i} - p_t v_t\rangle + \left\langle g_{t,i}\Phi(s_t)^\top, M_{k(t,i)} - M\right\rangle_F\\
&\quad + \langle g_{t,i}, b_{k(t,i)} - b\rangle\\
&= \sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, p_{t,i}v_{t,i} - p_t v_t\rangle + \left\langle g_{t,i}\Phi(s_t)^\top, \mathscr{P}_{k(t,i)}\mathscr{V}_{k(t,i)} - \mathscr{P}\mathscr{V}\right\rangle_F\\
&\quad + \langle g_{t,i}, P_{k(t,i)}V_{k(t,i)} - PV\rangle
\end{aligned}$$

and further decompose these three terms into the linear regret of coin-betting algorithm and linear regret of online subgradient descent algorithm. Note that the first term coincides with the term we deal with in the proof of Proposition 4.1. Similarly, we obtain the decomposition by adding and subtracting three terms as follows.

$$
\begin{aligned}
&= \sum_{t=1}^{T}\sum_{i=1}^{n} \langle g_{t,i}, p_{t,i}v_{t,i} - p_t v_t\rangle \pm p_t\langle g_{t,i}, v_{t,i}\rangle \\
&\quad + \left\langle g_{t,i}\Phi(s_t)^\top, \mathscr{P}_{k(t,i)}\mathscr{V}_{k(t,i)} - \mathscr{P}\mathscr{V}\right\rangle_F \pm \mathscr{P}\left\langle g_{t,i}\Phi(s_t)^\top, \mathscr{V}_{k(t,i)}\right\rangle_F \\
&\quad + \langle g_{t,i}, P_{k(t,i)}V_{k(t,i)} - PV\rangle \pm P\langle g_{t,i}, V_{k(t,i)}\rangle
\end{aligned}
$$

By reorganizing these terms, we get

$$
\begin{aligned}
&= \underbrace{\sum_{t=1}^{T}\sum_{i=1}^{n} \langle g_{t,i}, v_{t,i}\rangle(p_{t,i} - p_t)}_{(1)} + \underbrace{\sum_{t=1}^{T} p_t \sum_{i=1}^{n}\langle g_{t,i}, v_{t,i} - v_t\rangle}_{(2)} \\
&\quad + \underbrace{\sum_{t=1}^{T}\sum_{i=1}^{n}\left\langle g_{t,i}\Phi(s_t)^\top, \mathscr{V}_{k(t,i)}\right\rangle_F(\mathscr{P}_{k(t,i)} - \mathscr{P})}_{(3)} \\
&\quad + \mathscr{P}\underbrace{\sum_{t=1}^{T}\sum_{i=1}^{n}\left\langle g_{t,i}\Phi(s_t)^\top, \mathscr{V}_{k(t,i)} - \mathscr{V}\right\rangle_F}_{(4)} \\
&\quad + \underbrace{\sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, V_{k(t,i)}\rangle(P_{k(t,i)} - P)}_{(5)} + P\underbrace{\sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, V_{k(t,i)} - V\rangle}_{(6)}.
\end{aligned}
\tag{B.8}
$$

Finally, we bound the above six terms to obtain the linear regret bound of our method. We see that term (1) is equivalent to the linear regret of $(p_{t,i})_{i=1}^n$ generated by the coin-betting algorithm 6 for tasks $t = 1, \ldots, T$, which is adopted to estimate the within-task magnitude p_t of vector $w_t - M\Phi(s_t) - b$ in our method. In this case, the coin-betting algorithm receives $(\langle g_{t,i}, v_{t,i}\rangle)_{i=1}^n$ as inputs and, following the Lipschitz and bounded inputs assumptions in Assumption 1, we have $|\langle g_{t,i}, v_{t,i}\rangle| \leq \|g_{t,i}\|\,\|v_{t,i}\| \leq RL$. Recalling the initial wealth $e > 0$ in the algorithm, by Proposition 3.2, we obtain the following bound on term (1),

$$\sum_{i=1}^{n}\langle g_{t,i}, v_{t,i}\rangle(p_{t,i}-p_t) \le RL\left[e+\Psi(e^{-1}|p_t|n)|p_t|\sqrt{n}\right]$$

and hence,

$$\sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, v_{t,i}\rangle(p_{t,i}-p_t) \le RL\left[eT+\sum_{t=1}^{T}\Psi(e^{-1}|p_t|n)|p_t|\sqrt{n}\right]. \tag{B.9}$$

Next, we see that term (2) is equivalent to the linear regret of $(v_{t,i})_{i=1}^{n}$ generated by the online subgradient descent algorithm 5 on $\mathcal{B}(0,1)$ aiming at estimating the within-task direction v_t of vector $w_t - M\Phi(s_t) - b$ in our method. In this case, the subgradient descent algorithm receives $(g_{t,i})_{i=1}^{n}$ and since $\|g_{t,i}\| \le RL$, by Proposition 3.1, we obtain the bound on term (2)

$$\sum_{i=1}^{n}\langle g_{t,i}, v_{t,i}-v_t\rangle \le RL2\sqrt{2}\sqrt{n}$$

and hence,

$$\sum_{t=1}^{T}p_t\sum_{i=1}^{n}\langle g_{t,i}, v_{t,i}-v_t\rangle \le \sum_{t=1}^{T}RL2\sqrt{2}p_t\sqrt{n}. \tag{B.10}$$

Similarly, term (3) coincides with the linear regret of $(\mathscr{P}_{k(t,i)})_{t=1,i=1}^{T,n}$ generated by the coin-betting algorithm 6 aiming at estimating the magnitude $\mathscr{P}$ of transition matrix M. In this case, the coin-betting algorithm receives $(\langle g_{t,i}\Phi(s_t)^\top, \mathscr{V}_{k(t,i)}\rangle_F)_{t=1,i=1}^{T,n}$, whose value can be bounded as follows.

$$\begin{aligned}\left|\left\langle g_{t,i}\Phi(s_t)^\top, \mathscr{V}_{k(t,i)}\right\rangle_F\right| &\le \left\|g_{t,i}\Phi(s_t)^\top\right\|_F \left\|\mathscr{V}_{k(t,i)}\right\|_F \\ &= \|g_{t,i}\|\,\|\Phi(s_t)\|\left\|\mathscr{V}_{k(t,i)}\right\|_F \\ &\le LR\cdot K\cdot 1\end{aligned}$$

where the second equality follows from the result

$$\left\|g_{t,i}\Phi(s_t)^\top\right\|_F^2 = \mathrm{Tr}\left(\Phi(s_t)g_{t,i}^\top g_{t,i}\Phi(s_t)^\top\right) = \mathrm{Tr}\left(g_{t,i}^\top g_{t,i}\Phi(s_t)^\top\Phi(s_t)\right) = \|g_{t,i}\|^2\|\Phi(s_t)\|^2$$

and the third inequality follows from Assumption 1 and 2, and construction of direction $\mathscr{V}_{k(t,i)}$ on $\mathcal{B}(0,1)$,

$$\|g_{t,i}\| = |h_{t,i}| \, \|x_{t,i}\| \leq LR, \ \|\Phi(s_t)\| \leq K, \left\|\mathscr{V}_{k(t,i)}\right\|_F \leq 1.$$

Recalling the initial wealth $\mathscr{E} > 0$ used in the method, by Proposition 3.2, we get

$$\sum_{t=1}^{T}\sum_{i=1}^{n}\left\langle g_{t,i}\Phi(s_t)^\top, \mathscr{V}_{k(t,i)}\right\rangle_F (\mathscr{P}_{k(t,i)} - \mathscr{P}) \leq RLK\left[\mathscr{E} + \Psi(\mathscr{E}^{-1}|\mathscr{P}|nT)|\mathscr{P}|\sqrt{nT}\right] \tag{B.11}$$

Term (4) coincides with the linear regret of $(\mathscr{V}_{k(t,i)})_{t=1,i=1}^{T,n}$ generated by the online subgradient descent algorithm 5 aiming at estimating direction $\mathscr{V}$ of transition matrix M. In this case, the subgradient descent algorithm receives $g_{t,i}\Phi(s_t)^\top$ and since $\left\|g_{t,i}\Phi(s_t)^\top\right\|_F \leq RLK$, by Proposition 3.1, we have

$$\sum_{t=1}^{T}\sum_{i=1}^{n}\left\langle g_{t,i}\Phi(s_t)^\top, \mathscr{V}_{k(t,i)} - \mathscr{V}\right\rangle_F \leq RLK2\sqrt{2}\sqrt{nT}$$

and hence,

$$\mathscr{P}\sum_{t=1}^{T}\sum_{i=1}^{n}\left\langle g_{t,i}\Phi(s_t)^\top, \mathscr{V}_{k(t,i)} - \mathscr{V}\right\rangle_F \leq RLK2\sqrt{2}\mathscr{P}\sqrt{nT}. \tag{B.12}$$

Following the same logic, term (5) can be viewed as the linear regret of $(P_{k(t,i)})_{t=1,i=1}^{T,n}$ returned by the coin-betting algorithm for inferring the magnitude P of transition b and term (6) can be viewed as the linear regret of $(V_{k(t,i)})_{t=1,i=1}^{T,n}$ returned by the subgradient descent algorithm for inferring the direction V of transition b in our method. Therefore, we can bound the two terms as

$$\sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, V_{k(t,i)}\rangle(P_{k(t,i)} - P) \leq RL\left[E + \Psi(E^{-1}|P|nT)|P|\sqrt{nT}\right] \tag{B.13}$$

and

$$\begin{aligned}\sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, V_{k(t,i)} - V\rangle &\leq RL2\sqrt{2}\sqrt{nT}\\ P\sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, V_{k(t,i)} - V\rangle &\leq RL2\sqrt{2}P\sqrt{nT}.\end{aligned} \tag{B.14}$$

Finally, recalling

$$p = \|w_t - M\Phi(s_t) - b\|\,,\ \mathscr{P} = \|M\|_F\,,\ P = \|b\|$$

and plugging Equations (B.9) - (B.14) into (B.8), we will get the desired bound stated in the proposition. □

B.3 Proof of Theorem 4.4

Theorem. *Let Assumption 1 and 2 hold. Consider* $\mathbf{Z} = (Z_t)_{t=1}^T$, $Z_t = (x_{t,i}, y_{t,i})_{i=1}^n$ *deriving from* T *different tasks with associated side information* $\mathbf{S} = (s_t)_{t=1}^T$. *Let* $(w_{\theta_t,i})_{t=1,i=1}^{T,n}$ *be the iterates generated by Algorithm 9 over* $\mathbf{Z}$ *and* $\mathbf{S}$. *Then for any* $w_t \in \mathbb{R}^d$, $M \in \mathbb{R}^{d\times k}$ *and* $b \in \mathbb{R}^d$,

$$\sum_{t=1}^T \sum_{i=1}^n \ell_{t,i}\left(\langle x_{t,i}, w_{\theta_t,i}\rangle\right) - \ell_{t,i}\left(\langle x_{t,i}, w_t\rangle\right) \le \sum_{t=1}^T \sum_{i=1}^n \left\langle g_{t,i}, w_{\theta_t,i} - w_t\right\rangle \le A + B + C \tag{B.15}$$

where

$$\begin{aligned} A &= RL\left[eT + \left(2\sqrt{2}\mathrm{Var}(\tau_{M,b}) + \widehat{\mathrm{Var}}(\tau_{M,b})\right)\sqrt{n}T\right] \\ B &= RLK\left[\mathscr{E}n + \left(2\sqrt{2} + \Psi\left(\mathscr{E}^{-1}\|M\|_F T\right)\right)\|M\|_F\, n\sqrt{T}\right] \\ C &= RL\left[En + \left(2\sqrt{2} + \Psi\left(E^{-1}\|b\|\, T\right)\right)\|b\|\, n\sqrt{T}\right] \end{aligned} \tag{B.16}$$

and $\Psi(a) = \sqrt{\log(1+24a^2)}$.

Proof. The first inequality follows from the convexity of $\ell_{t,i}(\langle x_{t,i}, \cdot\rangle)$ and the fact $g_{t,i} \in \partial\ell_{t,i}(\langle x_{t,i}, \cdot\rangle)(w_{\theta_t,i})$. The second step follows from the same decomposition strategy we adopt in the proof of Theorem 4.3. Specifically, we use the slightly modified definition (4.10) to rewrite $w_{\theta_t,i}$ and w_t, and decompose the

linear regret into six terms as follows.

$$\begin{aligned}
\sum_{t=1}^{T}\sum_{i=1}^{n}\left\langle g_{t,i}, w_{\theta_t,i} - w_t\right\rangle &= \sum_{t=1}^{T}\sum_{i=1}^{n}\left\langle g_{t,i}, p_{\theta_t,i}v_{\theta_t,i} + M_t\Phi(s_t) + b_t - (p_t v_t + M\Phi(s_t) + b)\right\rangle \\
&\le \underbrace{\sum_{t=1}^{T}\sum_{i=1}^{n}\langle g_{t,i}, v_{\theta_t,i}\rangle(p_{\theta_t,i} - p_t)}_{(1)} + \underbrace{\sum_{t=1}^{T} p_t \sum_{i=1}^{n}\langle g_{t,i}, v_{\theta_t,i} - v_t\rangle}_{(2)} \\
&\quad + \underbrace{\sum_{t=1}^{T}\left\langle G_t\Phi(s_t)^\top, \mathscr{V}_t\right\rangle_F(\mathscr{P}_t - \mathscr{P})}_{(3)} \\
&\quad + \mathscr{P}\underbrace{\sum_{t=1}^{T}\left\langle G_t\Phi(s_t)^\top, \mathscr{V}_t - \mathscr{V}\right\rangle_F}_{(4)} \\
&\quad + \underbrace{\sum_{t=1}^{T}\langle G_t, V_t\rangle(P_t - P)}_{(5)} + P\underbrace{\sum_{t=1}^{T}\langle G_t, V_t - V\rangle}_{(6)}.
\end{aligned} \tag{B.17}$$

We see that term (1) and term (2) can be bounded as demonstrated in the proof of theorem 4.3, i.e. (B.9) and (B.10). To bound the rest, we observe that term (3) is equivalent to the linear regret of $(\mathscr{P}_t)_{t=1}^T$ returned by the coin betting algorithm. In this case, the algorithm receives $(\langle G_t\Phi(s_t)^\top, \mathscr{V}_t\rangle_F)_{t=1}^T$ and

$$\begin{aligned}
\left|\left\langle G_t\Phi(s_t)^\top, \mathscr{V}_t\right\rangle_F\right| &\le \left\|\sum_{i=1}^{n} g_{t,i}\Phi(s_t)^\top\right\|_F \|\mathscr{V}_t\|_F \\
&\le \sum_{i=1}^{n}\|g_{t,i}\Phi(s_t)\|_F \|\mathscr{V}_t\|_F \\
&\le nRLK\cdot 1
\end{aligned}$$

where the second inequality follows from the induction of the triangle inequality. Recalling the initial wealth $\mathscr{E} > 0$, by Proposition 3.2, we obtain

$$\sum_{t=1}^{T}\left\langle G_t\Phi(s_t)^\top, \mathscr{V}_t\right\rangle_F(\mathscr{P}_t - \mathscr{P}) \le RLK\left[\mathscr{E}n + \Psi(\mathscr{E}^{-1}|\mathscr{P}|T)|\mathscr{P}|n\sqrt{T}\right]. \tag{B.18}$$

Term (4) coincides with the linear regret of $(\mathscr{V}_t)_{t=1}^T$ generated by the online subgradient descent

algorithm. In this case, the algorithm receives $(G_t\Phi(s_t)^\top)_{t=1}^T$ and

$$\left\|G_t\Phi(s_t)^\top\right\|_F = \left\|\sum_{i=1}^n g_{t,i}\Phi(s_t)^\top\right\|_F \leq \sum_{i=1}^n \left\|g_{t,i}\Phi(s_t)^\top\right\|_F \leq nRLK.$$

By Proposition 3.1, we get

$$\sum_{t=1}^T \left\langle G_t\Phi(s_t)^\top, \mathscr{V}_t - \mathscr{V}\right\rangle_F \leq RLK2\sqrt{2}n\sqrt{T}$$

and hence,

$$\mathscr{P}\sum_{t=1}^T \left\langle G_t\Phi(s_t)^\top, \mathscr{V}_t - \mathscr{V}\right\rangle_F \leq RLK2\sqrt{2}\mathscr{P}n\sqrt{T} \tag{B.19}$$

Finally, we bound term (5) and term (6) in a similar fashion. We see that term (5) is equivalent to the linear regret of $(P_t)_{t=1}^T$ returned by the coin-betting algorithm which intakes $\langle G_t, V_t\rangle_{t=1}^T$ to infer the magnitude P. Since

$$|\langle G_t, V_t\rangle| \leq \|G_t\|\,\|V_t\| = \left\|\sum_{i=1}^n g_{t,i}\right\|\|V_t\| \leq \sum_{i=1}^n \|g_{t,i}\|\,\|V_t\| \leq RLn,$$

and recall the initial wealth $E > 0$ used in the method, by Proposition 3.1, we have

$$\sum_{t=1}^T \langle G_t, V_t\rangle(P_t - P) \leq RL\left[En + \Psi(E^{-1}|P|T)|P|n\sqrt{T}\right]. \tag{B.20}$$

For term (6), we observe that it coincides with the linear regret of $(V_t)_{t=1}^T$ generated by the online subgradient algorithm which intakes $(G_t)_{t=1}^T$ to infer the direction V. Since $\|G_t\| \leq RLn$, by Proposition 3.1, we have

$$\sum_{t=1}^T \langle G_t, V_t - V\rangle \leq RL2\sqrt{2}n\sqrt{T}$$

and hence,

$$P\sum_{t=1}^T \langle G_t, V_t - V\rangle \leq RL2\sqrt{2}Pn\sqrt{T}. \tag{B.21}$$

Again, recalling

$$p = \|w_t - M\Phi(s_t) - b\|\,,\ \mathscr{P} = \|M\|_F\,,\ P = \|b\|$$

and substituting (B.9), (B.10), (B.18) - (B.21) into (B.17), we obtain the desired bound in the statement. □

B.4 Proof of Theorem 4.5

Theorem (Multi-Task Risk Bound for Algorithm 8)**.** *Let the same assumptions in Theorem 4.3 hold in the deterministic heterogeneous tasks setting. Then, for any $M \in \mathbb{R}^{d\times k}, b \in \mathbb{R}^d$, in the expectation w.r.t. the sampling of the datasets $\mathbf{Z} = (Z_t)_{t=1}^T$,*

$$\mathbb{E}_{\mathbf{Z}}\left[\frac{1}{T}\sum_{t=1}^{T}\mathcal{R}_{\mu_t}(\bar{w}_t)\right] - \frac{1}{T}\sum_{t=1}^{T}\mathcal{R}_{\mu_t}(w_{\mu_t}) \leq \frac{1}{nT}(A+B+C) \tag{B.22}$$

where

$$\begin{aligned} A &= RL\left[eT + \left(2\sqrt{2}\mathrm{Var}_{\mathrm{MTL}}(\tau_{M,b}) + \widehat{\mathrm{Var}}_{\mathrm{MTL}}(\tau_{M,b})\right)\sqrt{nT}\right] \\ B &= RLK\left[\mathscr{E} + \left(2\sqrt{2} + \Psi\left(\mathscr{E}^{-1}\|M\|_F nT\right)\right)\|M\|_F\sqrt{nT}\right] \\ C &= RL\left[E + \left(2\sqrt{2} + \Psi\left(E^{-1}\|b\| nT\right)\right)\|b\|\sqrt{nT}\right] \end{aligned} \tag{B.23}$$

with

$$\begin{aligned} \mathrm{Var}_{\mathrm{MTL}}(\tau_{M,b}) &= \frac{1}{T}\sum_{t=1}^{T}\|w_{\mu_t} - M\Phi(s_t) - b\|\,, \\ \widehat{\mathrm{Var}}_{\mathrm{MTL}}(\tau_{M,b}) &= \frac{1}{T}\sum_{t=1}^{T}\Psi\left(e^{-1}\|w_{\mu_t} - M\Phi(s_t) - b\| n\right)\|w_{\mu_t} - M\Phi(s_t) - b\|\,. \end{aligned} \tag{B.24}$$

and $\Psi(a) = \sqrt{\log(1+24a^2)}$.

Proof. For this proof, we will use the following result taken from [4] (see Appendix C in [4]).

Proposition (Online-To-Batch Conversion, [**Propostion 7**, 4])**.** *Consider an online algorithm that, when applied to a sequence Z, returns weight vectors $(w_i)_{i=1}^n$. Suppose we run the algorithm over datasets*

$\mathbf{Z} = (Z_t)_{t=1}^T$ *and if the datapoints are i.i.d. sampled from distributions, we have*

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

By substituting the across-task regret bound we obtained in Theorem 4.3 specified to target vector w_{μ_t}, we get the right side bound

$$\mathbb{E}_{\mathbf{Z}}\left[\frac{1}{nT}\left(A + B + C\right)\right] = \frac{1}{nT}\left(A + B + C\right)$$

where A, B and C are defined as in Theorem 4.5. □

B.5 Proof of Theorem 4.6

Theorem. *Let the same assumptions in Theorem 4.4 hold in the statistical meta-learning setting. Then, for any $M \in \mathbb{R}^{d\times k}$ and $b \in \mathbb{R}^d$, the transfer risk bound for Algorithm 9 in expectation w.r.t. the sampling of the datasets $\mathbf{Z}$ and the uniform sampling of $\hat{t} \sim \mathcal{U}(T)$ is expressed as*

$$\mathbb{E}_{\hat{t}\sim\mathcal{U}(T)}\,\mathbb{E}_{\mathbf{Z}}\left[\mathbb{E}_{(\mu,s)\sim\rho}\mathbb{E}_{Z\sim\mu^n}\,\mathcal{R}_{\mu}(\bar{w}_{\tau_{\hat{t}}}(Z))\right] - \mathbb{E}_{\mu\sim\rho}\,\mathcal{R}_{\mu}(w_{\mu}) \leq \frac{1}{nT}(A + B + C) \tag{B.25}$$

where

$$\begin{aligned} A &= RL\left[eT + \left(2\sqrt{2}\mathrm{Var}_{\mathrm{META}}(\tau_{M,b}) + \widehat{\mathrm{Var}}_{\mathrm{META}}(\tau_{M,b})\right)\sqrt{n}T\right] \\ B &= RLK\left[\mathscr{E}n + \left(2\sqrt{2} + \Psi\left(\mathscr{E}^{-1}\left\|M\right\|_F T\right)\right)\left\|M\right\|_F n\sqrt{T}\right] \\ C &= RL\left[En + \left(2\sqrt{2} + \Psi\left(E^{-1}\left\|b\right\| T\right)\right)\left\|b\right\| n\sqrt{T}\right] \end{aligned} \tag{B.26}$$

with

$$\begin{aligned} \mathrm{Var}_{\mathrm{META}}(\tau_{M,b}) &= \mathbb{E}_{(\mu,s)\sim\rho}\left\|w_\mu - M\Phi(s) - b\right\|, \\ \widehat{\mathrm{Var}}_{\mathrm{META}}(\tau_{M,b}) &= \mathbb{E}_{(\mu,s)\sim\rho}\Big[\Psi\left(e^{-1}\left\|w_\mu - M\Phi(s) - b\right\| n\right)\left\|w_{\mu_t} - M\Phi(s) - b\right\|\Big], \end{aligned} \tag{B.27}$$

and $\Psi(a) = \sqrt{\log(1 + 24a^2)}$.

Proof. The proof of the expectation bound on transfer risk for Algorithm 9 is adapted from Proposition 10 in [4]. The key step is to use the following online-to-batch conversion lemma to automatically pass

the across-tasks regret bound obtained in Theorem 4.4 to the statistical meta-learning setting. More specifically, by inserting the results we obtained in the Theorem 4.4 into the right side of the inequality in the lemma below, we can obtain the desired statement.

□

Lemma (Online-To-Batch Conversion for Algorithm 9)**.** *Under the same assumptions in Theorem 4.6, the following relation holds*

$$\mathbb{E}_{\hat{t}\sim\mathcal{U}(T)}\,\mathbb{E}_{\mathbf{Z}}\left[\mathbb{E}_{(\mu,s)\sim\rho}\mathbb{E}_{Z\sim\mu^n}\,\mathcal{R}_\mu(\bar{w}(\tau_{\hat{t}}(s),Z))\right]-\mathbb{E}_{\mu\sim\rho}\,\mathcal{R}_\mu(w_\mu)\leq \mathbb{E}_{\mathbf{Z}}\left[\frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\ell_{t,i}\left(\langle x_{t,i},w_{\tau_t,i}\rangle\right)-\ell_{t,i}\left(\langle x_{t,i},w_{\mu_t}\rangle\right)\right].$$

Proof. The proof of the aforementioned lemma is an adaptation of Proposition 10 in [4], with the difference being that we have side information s sampled together with the task μ from the environment, i.e. $(\mu,s)\sim\rho$ and we use conditioning function τ to decide meta-parameter instead of simply using a single meta-parameter in our case.

In the proof, we use the explicit expression of $\mathbb{E}_{\mathbf{Z}}$ written as

$$\mathbb{E}_{\mathbf{Z}}=\mathbb{E}_{(\mu_1,s_1),\ldots,(\mu_T,s_T)\sim\rho^T}\mathbb{E}_{Z_1\sim\mu_1^n,\ldots,Z_T\sim\mu_T^n}.$$

The transfer risk in expectation w.r.t. the uniform sampling of task $\hat{t}$ and the sampling of the datasets $\mathbf{Z}$ can then be re-written as

$$\mathbb{E}_{\hat{t}\sim\mathcal{U}(T)}\mathbb{E}_{(\mu_1,s_1),\ldots,(\mu_T,s_T)\sim\rho^T}\mathbb{E}_{Z_1\sim\mu_1^n,\ldots,Z_T\sim\mu_T^n}\mathbb{E}_{(\mu,s)\sim\rho}\mathbb{E}_{Z\sim\mu^n}\,\mathcal{R}_\mu(\bar{w}(\tau_{\hat{t}}(s),Z))\quad\cdots\ (\star).$$

Writing the expectation w.r.t. $\hat{t}$ explicitly, we have

$$(\star)=\frac{1}{T}\,\mathbb{E}_{(\mu_1,s_1),\ldots,(\mu_T,s_T)\sim\rho^T}\mathbb{E}_{Z_1\sim\mu_1^n,\ldots,Z_T\sim\mu_T^n}\sum_{t=1}^{T}\mathbb{E}_{(\mu,s)\sim\rho}\mathbb{E}_{Z\sim\mu^n}\,\mathcal{R}_{\mu_t}(\bar{w}(\tau_t(s),Z)),$$

and because, by construction, for any $t\in\{1,\ldots,T\}$ the meta-parameter τ_t only depends on $(Z_j)_{j=1}^{i-1}$ and Z_t and (μ_t,s_t) are sampled i.i.d., we can express the dependency more explicitly as

$$=\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{(\mu_1,s_1),\ldots,(\mu_{t-1},s_{t-1})\sim\rho^{t-1}}\,\mathbb{E}_{Z_1\sim\mu_1^n,\ldots,Z_{t-1}\sim\mu_{t-1}^n}\,\mathbb{E}_{(\mu_t,s_t)\sim\rho}\,\mathbb{E}_{Z_t\sim\mu_t^n}\mathcal{R}_{\mu_t}(\bar{w}(\tau_t(s_t),Z_t)),$$

applying the Jensen's inequality to the convex function $\mathcal{R}_{\mu_t}$, we get

$$\leq \frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\mathbb{E}_{(\mu_1,s_1),\ldots,(\mu_{t-1},s_{t-1})\sim\rho^{t-1}}\,\mathbb{E}_{Z_1\sim\mu_1^n,\ldots,Z_{t-1}\sim\mu_{t-1}^n}\,\mathbb{E}_{(\mu_t,s_t)\sim\rho}\,\mathbb{E}_{Z_t\sim\mu_t^n}\mathcal{R}_{\mu_t}(w_i(\tau_t(s_t),Z_t)).$$

Next, we exploit the fact that $w_i(\tau_t(s_t),Z_t)$ depends only on datasets $(Z_k)_{k=1}^{t-1}$ and the first $i-1$ datapoints of the t-th dataset $(z_{t,j})_{j=1}^{i-1}$ to reduce the expectation w.r.t. Z_t (i.e. $\mathbb{E}_{Z_t\sim\mu_t^n}$) to $\mathbb{E}_{(z_{t,j})_{j=1}^{i-1}\sim\mu_t^{i-1}}$. Recall that

$$\begin{aligned}\mathcal{R}_{\mu_t}(w_i(\tau_t(s_t),Z_t)) &= \mathbb{E}_{z\sim\mu_t}\ell(\langle x, w_i(\tau_t(s_t)\rangle, y)\\ &= \mathbb{E}_{z_{t,i}\sim\mu_t}\ell(\langle x_{t,i}, w_i(\tau_t(s_t)\rangle, y_{t,i}),\end{aligned}$$

we obtain,

$$\begin{aligned}&\frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\mathbb{E}_{(\mu_1,s_1),\ldots,(\mu_{t-1},s_{t-1})\sim\rho^{t-1}}\,\mathbb{E}_{Z_1\sim\mu_1^n,\ldots,Z_{t-1}\sim\mu_{t-1}^n}\,\mathbb{E}_{(\mu_t,s_t)\sim\rho}\,\mathbb{E}_{Z_t\sim\mu_t^n}\mathcal{R}_{\mu_t}(w_i(\tau_t(s_t),Z_t))\\
&= \frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\mathbb{E}_{(\mu_1,s_1),\ldots,(\mu_{t-1},s_{t-1})\sim\rho^{t-1}}\,\mathbb{E}_{Z_1\sim\mu_1^n,\ldots,Z_{t-1}\sim\mu_{t-1}^n}\\
&\mathbb{E}_{(\mu_t,s_t)\sim\rho}\,\mathbb{E}_{(z_{t,j})_{j=1}^{i-1}\sim\mu_t^{i-1}}\mathbb{E}_{z_{t,i}\sim\mu_t}\ell_{t,i}(\langle x_{t,i}, w_i(\tau_t(s_t)\rangle)\\
&= \frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\mathbb{E}_{(\mu_1,s_1),\ldots,(\mu_{t-1},s_{t-1})\sim\rho^{t-1}}\,\mathbb{E}_{Z_1\sim\mu_1^n,\ldots,Z_{t-1}\sim\mu_{t-1}^n}\mathbb{E}_{(\mu_t,s_t)\sim\rho}\,\mathbb{E}_{(z_{t,i})_{j=1}^{i}\sim\mu_t^{i}}\ell_{t,i}(\langle x_{t,i}, w_i(\tau_t(s_t)\rangle)\\
&= \mathbb{E}_{(\mu_1,s_1),\ldots,(\mu_T,s_T)\sim\rho^T}\mathbb{E}_{Z_1\sim\mu_1^n,\ldots,Z_T\sim\mu_T^n}\,\frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\ell_{t,i}(\langle x_{t,i}, w_i(\tau_t(s_t)\rangle)\end{aligned} \tag{B.28}$$

where the second to last equality follows by combining $\mathbb{E}_{(z_{t,j})_{j=1}^{i-1}\sim\mu_t^{i-1}}$ and $\mathbb{E}_{z_{t,i}\sim\mu_t}$ into $\mathbb{E}_{(z_{t,i})_{j=1}^{i}\sim\mu_t^{i}}$.

We also see that, by the i.i.d. sampling of the data, we can write

$$\mathbb{E}_{\mu\sim\rho}\,\mathcal{R}_{\mu}(w_\mu) = \mathbb{E}_{(\mu_1,s_1),\ldots,(\mu_T,s_T)\sim\rho^T}\mathbb{E}_{Z_1\sim\mu_1^n,\ldots,Z_T\sim\mu_T^n}\,\frac{1}{nT}\sum_{t=1}^{T}\sum_{i=1}^{n}\ell_{t,i}(\langle x_{t,i}, w_{\mu_t}\rangle). \tag{B.29}$$

Combining (B.28) and (B.29), we derive the desired statement. □

Bibliography

[1] Jonathan Baxter. "A Model of Inductive Bias Learning". In: *Information Sciences* 12 (2000), pp. 149–198.

[2] Ashok Cutkosky and Francesco Orabona. "Black-Box Reductions for Parameter-free Online Learning in Banach Spaces". In: *Conference On Learning Theory (COLT)*. Vol. 75. Proceedings of Machine Learning Research. 2018, pp. 1493–1529.

[3] Giulia Denevi, Massimiliano Pontil, and Carlo Ciliberto. *The Advantage of Conditional Meta-Learning for Biased Regularization and Fine-Tuning*. 2020. arXiv: 2008.10857.

[4] Giulia Denevi, Dimitris Stamos, and Massimiliano Pontil. "Online Parameter-Free Learning of Multiple Low Variance Tasks". In: *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence* 124 (2020).

[5] Giulia Denevi et al. "Learning-to-learn stochastic gradient descent with biased regularization". In: *36th International Conference on Machine Learning* (2019), pp. 2814–2843.

[6] Giulia Denevi et al. "Online-Within-Online Meta-Learning". In: *Advances in Neural Information Processing Systems 32*. 2019, pp. 13110–13120.

[7] Theodoras Evgeniou and Massimiliano Pontil. "Regularized multi-task learning". In: *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* June (2004), pp. 109–117.

[8] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *34th International Conference on Machine Learning, ICML 2017* 3 (2017), pp. 1856–1868.

[9] Chelsea Finn et al. "Online Meta-Learning". In: *Proceedings of the 36th International Conference on Machine Learning* (2019), pp. 1920–1930.

[10] Dylan J Foster et al. "Parameter-Free Online Learning via Model Selection". In: *Advances in Neural Information Processing Systems* (2017), pp. 6020–6030.

[11] Elad Hazan. "Introduction to Online Convex Optimization". In: *Foundations and Trends® in Optimization* 2.3-4 (2016), pp. 157–325.

[12] Daniel Hernandez-Stumpfhauser, F. Jay Breidt, and Mark J. van der Woerd. "The general projected normal distribution of arbitrary dimension: Modeling and Bayesian inference". In: *Bayesian Analysis* 12.1 (2017), pp. 113–133.

[13] Mikhail Khodak, Maria Fiorina Balcan, and Ameet Talwalkar. "Provable guarantees for gradient-based meta-learning". In: *36th International Conference on Machine Learning* (2019), pp. 651–675.

[14] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. "Adaptive Gradient-Based Meta-Learning Methods". In: *Advances in Neural Information Processing Systems* (2019), pp. 5917–5928.

[15] Francesco Orabona. *A Modern Introduction to Online Learning*. 2019. arXiv: 1912.13213.

[16] Francesco Orabona and David Pal. "Coin Betting and Parameter-Free Online Learning". In: *Advances in Neural Information Processing Systems* (2016), pp. 577–585.

[17] Ali Rahimi and Benjamin Recht. "Random Features for Large-Scale Kernel Machines". In: *Advances in Neural Information Processing Systems* (2008), pp. 1177–1184.

[18] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. 2013.

[19] Risto Vuorio et al. "Multimodal Model-Agnostic Meta-Learning via Task-Aware Modulation". In: *Advances in Neural Information Processing Systems* (2019), pp. 1–12.

[20] Ruohan Wang, Yiannis Demiris, and Carlo Ciliberto. *A Structured Prediction Approach for Conditional Meta-Learning*. 2020. arXiv: 2002.08799.

[21] Huaxiu Yao et al. "Hierarchically structured meta-learning". In: *36th International Conference on Machine Learning* (2019), pp. 12189–12209.