The many faces of online learning

Hoeven, D. van der

CHAPTER 5

MetaGrad: Universal Adaptation using Multiple Learning Rates in Online Learning

This chapter is based on: Van Erven, T., Koolen, W. M., and Van der Hoeven, D. (2020a). Metagrad: Universal adaptation using multiple learning rates in online learning. *Manuscript in preparation*.[1]

Abstract

In online convex optimization it is well known that certain subclasses of objective functions are much easier than arbitrary convex functions. We are interested in designing universally adaptive methods that can automatically get fast rates in as many such subclasses as possible, without any manual tuning. We provide a new universally adaptive method, MetaGrad, that is robust to general convex losses but adapts to a broad class of functions, including exp-concave and strongly convex functions, but also various types of stochastic and non-stochastic functions without any curvature. For instance, MetaGrad can achieve logarithmic regret on the unregularized hinge loss over the unit ball, even though the hinge loss has no curvature, if the data come from a favorable probability distribution. We prove this by drawing a connection to the Bernstein condition, which is known to imply fast rates in offline statistical learning. MetaGrad further adapts automatically to the size of the gradients. Its main feature is that it simultaneously considers multiple learning rates. Unlike previous methods with provable regret guarantees, however, its learning rates are not monotonically decreasing over time and are not tuned based on a theoretically derived bound on the regret. Instead, they are weighted directly proportional to their empirical performance on the data using a tilted exponential

[1]The author of this dissertation performed the following tasks: performing the experiments, co-deriving part of the theoretical results, and co-writing the paper

weights meta-algorithm. We provide three versions of MetaGrad. The full matrix version maintains a full covariance matrix and is applicable to learning tasks for which we can afford update time quadratic in the dimension. The other two versions provide speed-ups for high-dimensional learning tasks with an update time that is linear in the dimension: one is based on sketching, the other on running a separate copy of the basic algorithm per coordinate. We compare all versions of MetaGrad on benchmark online classification and regression tasks, showing that they consistently outperform both online gradient descent and AdaGrad.

5.1 Introduction

Methods for *online convex optimization* (OCO) (Shalev-Shwartz, 2011; Hazan et al., 2016) make it possible to optimize parameters sequentially, by processing convex functions in a streaming fashion. This is important in time series prediction where the data are inherently online; but it may also be convenient to process offline data sets sequentially, for instance if the data do not all fit into memory at the same time or if parameters need to be updated quickly when extra data become available.

The difficulty of an OCO task depends on the convex functions $f_1, f_2, \ldots, f_T$ that need to be optimized. The argument of these functions is a d-dimensional parameter vector $\boldsymbol{w}$ from a convex domain $\mathcal{W}$. Although this is abstracted away in the general framework, each function f_t usually measures the loss of the parameters on an underlying example $(\boldsymbol{x}_t, y_t)$ in a machine learning task. For example, in classification f_t might be the *hinge loss* $f_t(\boldsymbol{w}) = \max\{0, 1 - y_t\langle \boldsymbol{w}, \boldsymbol{x}_t\rangle\}$ or the *logistic loss* $f_t(\boldsymbol{w}) = \log\left(1 + e^{-y_t\langle \boldsymbol{w}, \boldsymbol{x}_t\rangle}\right)$, with $y_t \in \{-1, +1\}$. Thus the difficulty depends both on the choice of loss and on the observed data.

There are different methods for OCO, depending on assumptions that can be made about the functions. The simplest and most commonly used strategy is *online gradient descent* (GD). GD updates parameters $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla f_t(\boldsymbol{w}_t)$ by taking a step in the direction of the negative gradient, where the step size is determined by a parameter η_t called the *learning rate*. The goal is to minimize the *regret* over T rounds, which measures the difference in cumulative loss between the online iterates $\boldsymbol{w}_t$ and the best offline parameters $\boldsymbol{u}$. For learning rates $\eta_t \propto 1/\sqrt{t}$, GD guarantees that the regret for general convex functions is bounded by $O(\sqrt{T})$ (Zinkevich, 2003). Alternatively, if it is known beforehand that the functions are of an easier type, then better regret rates are sometimes possible. For instance, if the functions are *strongly convex*, then logarithmic regret $O(\log T)$ can be achieved by GD with much smaller learning rates $\eta_t \propto 1/t$ (Hazan et al., 2007), and, if they are *exp-concave*, then logarithmic regret $O(d \log T)$ can be achieved by the *Online Newton Step* (ONS) algorithm (Hazan et al., 2007).

This partitions OCO tasks into categories, leaving it to the user to choose the appropriate algorithm for their setting. Such a strict partition, apart from being a burden on the user, depends on an extensive cataloguing of all types of easier functions that might occur in practice. (See Section 5.3 for several ways in which the existing list of easy functions can be extended.) It also immediately raises the question of whether there are cases in between logarithmic and square-root regret (there are, see Theorem 21 in Section 5.3), and which algorithm to use then. And,

third, it presents the problem that the appropriate algorithm might depend on (the distribution of) the data (again see Section 5.3), which makes it entirely impossible to select the right algorithm beforehand.

These issues motivate the development of *adaptive* methods, which are no worse than $O(\sqrt{T})$ for general convex functions, but also automatically take advantage of easier functions whenever possible. An important step in this direction are the adaptive GD algorithm of Bartlett et al. (2007) and its proximal improvement by Do et al. (2009), which are able to interpolate between strongly convex and general convex functions if they are provided with a data-dependent strong convexity parameter in each round, and significantly outperform the main non-adaptive method (i.e. Pegasos, (Shalev-Shwartz et al., 2011)) in the experiments of Do et al.. Here we consider a significantly richer class of functions, which includes exp-concave functions, strongly convex functions, general convex functions that do not change between rounds (even if they have no curvature), and stochastic functions whose gradients satisfy the so-called Bernstein condition, which is well-known to enable fast rates in offline statistical learning (Bartlett and Mendelson, 2006; Van Erven et al., 2015; Koolen et al., 2016). The latter group can again include functions without curvature, like the unregularized hinge loss. All these cases are covered simultaneously by a new adaptive method we call *MetaGrad*, for multiple eta gradient algorithm. Theorem 23 below implies the following:

Theorem 19. *Suppose the diameter of the domain $\mathcal{W}$ and the ℓ_2-norms of the gradients $\boldsymbol{g}_t = \nabla f_t(\boldsymbol{w}_t)$ are both bounded by constants, and define $V_T^{\boldsymbol{u}} = \sum_{t=1}^{T} \left((\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t\right)^2$. Then MetaGrad's regret is simultaneously bounded by $O(\sqrt{T \log\log T})$, and by*

$$\sum_{t=1}^{T} f(\boldsymbol{w}_t) - \sum_{t=1}^{T} f_t(\boldsymbol{u}) \;\leq\; \sum_{t=1}^{T} (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t \;\leq\; O\left(\sqrt{V_T^{\boldsymbol{u}}\, d \ln(T/d)} + d \ln(T/d)\right) \tag{5.1.1}$$

for any $\boldsymbol{u} \in \mathcal{W}$.

Theorem 19 bounds the regret in terms of a measure of variance $V_T^{\boldsymbol{u}}$ that depends on the distance of the algorithm's choices $\boldsymbol{w}_t$ to the optimum $\boldsymbol{u}$, and which, in favorable cases, may be significantly smaller than T. Intuitively, this happens, for instance, when there is a stable optimum $\boldsymbol{u}$ that the algorithm's choices $\boldsymbol{w}_t$ converge to. Formal consequences are given in Section 5.3, which shows that this bound implies faster than $O(\sqrt{T})$ regret rates, often logarithmic in T, for all functions in the rich class mentioned above. In all cases the dependence on T in the rates matches what we would expect based on related work in the literature, and in most cases the dependence on the dimension d is also what we would expect. Only for

strongly convex functions is there an extra factor d. It seems that this is a real limitation of the method as presented here. In Section 5.9 we discuss a recent extension of MetaGrad by Zhang et al. (2019) that removes this limitation.

The main difficulty in achieving the regret guarantee from Theorem 19 is tuning a learning rate parameter η. In theory, η should be roughly proportional to $1/\sqrt{V_T^{\boldsymbol{u}}}$, but this is not possible using any existing techniques, because the optimum $\boldsymbol{u}$ is unknown in advance, and tuning in terms of a uniform upper bound $\max_{\boldsymbol{u}} V_T^{\boldsymbol{u}}$ ruins all desired benefits. MetaGrad therefore runs multiple supporting expert algorithms, each with a different learning rate η, and combines them with a novel controller algorithm that learns the empirically best learning rate for the OCO task in hand. Crucially, the overhead for learning the best expert is not of the usual order $O(\sqrt{T})$, which would ruin all desired benefits, but only costs a negligible $O(\log \log T)$.

The experts are instances of exponential weights on the continuous parameters $\boldsymbol{u}$ with a suitable surrogate loss function, which in particular causes the exponential weights distributions to be multivariate Gaussians. The resulting updates are closely related to the ONS algorithm on the original losses, where each expert receives the controller's gradients instead of its own. It is shown that $\lceil \log_2 T \rceil$ experts suffice, which is at most 30 as long as $T \leq 10^9$, and therefore seems computationally acceptable. If not, then the number of experts can be further reduced at the cost of slightly worse constants in the bound.

An important practical consideration for OCO algorithms is whether they can adapt to the Lipschitz-constant of the losses f_t, i.e. the maximum norm of the gradients. For instance, this is an important feature of AdaGrad (Duchi et al., 2011; McMahan and Streeter, 2010). The MetaGrad algorithm is also adaptive in this way. Our approach is a refinement of the techniques of Mhammedi et al. (2019): whereas their procedure may occasionally restart the whole MetaGrad algorithm, we only restart the controller but not the experts. Wherever possible, we further measure the size of the gradients by the (semi-)norm $\max_{\boldsymbol{w} \in \mathcal{W}} |\boldsymbol{w}^\intercal \boldsymbol{g}_t|$ instead of the larger $\max_{\boldsymbol{w}, \boldsymbol{v} \in \mathcal{W}} \|\boldsymbol{w} - \boldsymbol{v}\|_2 \|\boldsymbol{g}_t\|_2$. The difference is crucial in Section 5.5.1, where we consider a domain for which the diameter is infinite, but our norms are under control.

The version of MetaGrad described so far maintains a full covariance matrix of size $d \times d$, where d is the parameter dimension. This requires at least $O(d^2)$ computation steps per round to update, which is prohibitive for large d. We therefore also present two extensions: the first applies the matrix sketching approach of Luo et al. (2017) to approximate the matrix by a rank k sketch, and requires $O(kd)$ update time on average per round. Our second extension was inspired by the diagonal version of

AdaGrad (Duchi et al., 2011; McMahan and Streeter, 2010) and runs a separate copy of full MetaGrad per coordinate, which takes $O(d)$ computation per round, just like vanilla GD and AdaGrad. While the full matrix version of MetaGrad and its sketching approximation naturally favor parameters $\boldsymbol{u}$ with small ℓ_2-norm, the coordinatewise extension is appropriate for the ℓ_∞-norm.

Related Work If we disregard computational efficiency and omit Lipschitz-adaptivity, then the result of Theorem 19 can be achieved by finely discretizing the domain $\mathcal{W}$ and running the Squint algorithm for prediction with experts with each discretization point as an expert (Koolen and Van Erven, 2015). MetaGrad may therefore also be seen as a computationally efficient extension of Squint to the OCO setting.

As already mentioned, Zhang et al. (2019) extend MetaGrad to adapt to strongly convex functions. They further provide an extension for the case that the optimal parameters $\boldsymbol{u}$ vary over time, as measured in terms of the adaptive regret. See also the closely related extension of Squint for the adaptive regret by Neuteboom (2020).

Our focus in this work is on adapting to sequences of functions f_t that are easier than general convex functions, but we require an estimate $\hat{D}$ of the ℓ_2-norm of the optimum $\boldsymbol{u}$ as a hyperparameter. In contrast, a different line of work designs methods that can adapt to the norm of $\boldsymbol{u}$ over all of $\mathbb{R}^d$, but without providing adaptivity to the functions f_t (Mcmahan and Streeter, 2012; Orabona, 2014; Cutkosky and Orabona, 2018). It was thought for some time that these two directions could not be reconciled, because the impossibility result of Cutkosky and Boahen (2017) blocks simultaneous adaptivity to both the size of the gradients of the functions f_t and the norm of $\boldsymbol{u}$. The perspective has recently shifted, however, following discoveries of ways to partially circumvent this lower bound (Kempka et al., 2019; Cutkosky, 2019; Mhammedi and Koolen, 2020).

Another notion of adaptivity is explored in a series of work obtaining tighter bounds for linear functions f_t that vary little between rounds, as measured either by their deviation from the mean function or by successive differences (Hazan and Kale, 2010; Chiang et al., 2012; Steinhardt and Liang, 2014). Such bounds imply super fast rates for optimizing a fixed linear function, but reduce to slow $O(\sqrt{T})$ rates in the other cases of easy functions that we consider. Finally, the way MetaGrad's experts maintain a Gaussian distribution on parameters $\boldsymbol{u}$ is similar in spirit to AROW and related confidence weighted methods, as analyzed by (Crammer et al., 2009) in the mistake bound model.

Outline We start with the main definitions in the next section. Then Section 5.3 contains an extensive set of examples where Theorem 19 leads to fast rates, Section 5.4 presents the Full Matrix version of the MetaGrad algorithm, and Section 5.5 describes the faster sketching and coordinatewise extensions. Section 5.6 provides the analysis leading to Theorem 23 for the Full Matrix version of MetaGrad, which is a more detailed statement of Theorem 19 with several quantities replaced by data-dependent versions and with exact constants. Section 5.7 extends this analysis to the two other versions of MetaGrad. Then, in Section 5.8, we compare all versions of MetaGrad to GD and to AdaGrad in experiments with several benchmark classification and regression data sets. We conclude with possible further extensions of MetaGrad in Section 5.9.

5.2 Setup

We consider algorithms for OCO, which operate according to the protocol displayed in Protocol 9. In each round, the environment reveals a closed convex domain $\mathcal{W}_t$, which we assume contains the origin $\mathbf{0}$ (if not, it needs to be translated). In the introduction, we assumed that $\mathcal{W}_t = \mathcal{W}$ was fixed beforehand, but for the remainder of the paper we allow it to vary between rounds, which is needed in the context of the sketching version of MetaGrad (Section 5.5.1). Let $\boldsymbol{w}_t \in \mathcal{W}_t$ be the iterate produced by the algorithm in round t, let $f_t : \mathcal{W}_t \to \mathbb{R}$ be the convex loss function produced by the environment and let $\boldsymbol{g}_t = \nabla f_t(\boldsymbol{w}_t)$ be the (sub)gradient, which is the feedback given to the algorithm.[2] The *regret* over T rounds $R_T^{\boldsymbol{u}}$, its linearization $\tilde{R}_T^{\boldsymbol{u}}$ and our measure of variance $V_T^{\boldsymbol{u}}$ are defined as

$$R_T^{\boldsymbol{u}} = \sum_{t=1}^{T} \left(f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u}) \right), \qquad \tilde{R}_T^{\boldsymbol{u}} = \sum_{t=1}^{T} (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t,$$

$$V_T^{\boldsymbol{u}} = \sum_{t=1}^{T} \left((\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t \right)^2 \qquad \text{with respect to any } \boldsymbol{u} \in \bigcap_{t=1}^{T} \mathcal{W}_t.$$

By convexity of f_t, we always have $f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u}) \leq (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t$, which implies the first inequality in Theorem 19: $R_T^{\boldsymbol{u}} \leq \tilde{R}_T^{\boldsymbol{u}}$. Finally, wherever possible we measure the size of the gradient $\boldsymbol{g}_t$ in the *intrinsic (semi-)norm* for the domain $\mathcal{W}_t$:

$$\|\boldsymbol{g}\|_t = \max_{\boldsymbol{w} \in \mathcal{W}_t} |\boldsymbol{w}^\intercal \boldsymbol{g}|.$$

This is a norm in the typical case that $\mathcal{W}_t$ has full dimension d, and it is still a semi-norm in general. We note that the intrinsic norm is smaller than the usual

[2] If f_t is not differentiable at $\boldsymbol{w}_t$, any choice of subgradient $\boldsymbol{g}_t \in \partial f_t(\boldsymbol{w}_t)$ is allowed.

Algorithm 9 Online Convex Optimization from First-order Information

for $t = 1, 2, \ldots$ **do**
  Environment reveals convex domain $\mathcal{W}_t \subseteq \mathbb{R}^d$ containing the origin $\mathbf{0}$
  Learner plays $\boldsymbol{w}_t \in \mathcal{W}_t$
  Environment chooses a convex loss function $f_t : \mathcal{W}_t \to \mathbb{R}$
  Learner incurs loss $f_t(\boldsymbol{w}_t)$ and observes (sub)gradient $\boldsymbol{g}_t = \nabla f_t(\boldsymbol{w}_t)$
end for

upper bounds based on Hölder's inequality: $\|\boldsymbol{g}\|_t \leq \|\boldsymbol{g}\| \max_{\boldsymbol{w} \in \mathcal{W}_t} \|\boldsymbol{w}\|^*$ for any dual norms $\|\cdot\|$ and $\|\cdot\|^*$. The difference becomes essential in Section 5.5.1, where we consider a domain $\mathcal{W}_t$ that has an infinite radius $\max_{\boldsymbol{w} \in \mathcal{W}_t} \|\boldsymbol{w}\|^*$ in any norm $\|\cdot\|^*$, but for which $\|\boldsymbol{g}_t\|_t$ is still bounded. MetaGrad depends on (upper bounds on) the sizes of the gradients per round b_t, as well as their running maximum B_t:

$$b_t \geq \|\boldsymbol{g}_t\|_t, \qquad B_t = \max_{s \leq t} b_s, \tag{5.2.1}$$

with the convention that $B_0 = 0$. We would normally take the best upper bound $b_t = \|\boldsymbol{g}_t\|_t$, except if this is difficult to compute. In such cases, we may for example let $b_t = \|\boldsymbol{g}_t\| \max_{\boldsymbol{u} \in \mathcal{W}_t} \|\boldsymbol{u}\|^*$.

Further Notation We denote by $\lceil z \rceil_+ = \max\{\lceil z \rceil, 1\}$ the smallest integer that is at least z and at least 1.

5.3 Fast Rates Examples

In this section, we motivate our interest in the adaptive bound (5.1.1) by giving a series of examples in which it provides fast rates. Although MetaGrad is designed to handle time varying domains, for simplicity we will assume that the domain is fixed in this section. In this section we will also assume that the following standard boundedness assumptions hold for all $\boldsymbol{u}, \boldsymbol{w} \in \mathcal{W}$ and all t: $\|\boldsymbol{u} - \boldsymbol{w}\|_2 \leq D'$ and $\|\boldsymbol{g}_t\|_2 \leq G'$. The fast rates are all derived from two general sufficient conditions: one based on the directional derivative of the functions f_t and one for stochastic gradients that satisfy the *Bernstein condition*, which is the standard condition for fast rates in off-line statistical learning. Simple simulations that illustrate the conditions are provided in Section 5.10.1 and proofs are also postponed to Section 5.10.

Directional Derivative Condition In order to control the regret with respect to some point $\boldsymbol{u}$, the first condition requires a quadratic lower bound on the curvature of the functions f_t in the direction of $\boldsymbol{u}$:

Theorem 20. *Suppose, for a given $\boldsymbol{u} \in \mathcal{W}$, there exist constants $a, b > 0$ such that the functions f_t all satisfy*

$$f_t(\boldsymbol{u}) \geq f_t(\boldsymbol{w}) + a(\boldsymbol{u}-\boldsymbol{w})^\intercal \nabla f_t(\boldsymbol{w}) + b\left((\boldsymbol{u}-\boldsymbol{w})^\intercal \nabla f_t(\boldsymbol{w})\right)^2 \qquad \text{for all } \boldsymbol{w} \in \mathcal{W}. \tag{5.3.1}$$

Then any method with regret bound (5.1.1) *incurs logarithmic regret, $R_T^{\boldsymbol{u}} = O(d \ln T)$, with respect to $\boldsymbol{u}$.*

The case $a = 1$ of this condition was introduced by (Hazan et al., 2007), who show that it is satisfied for all $\boldsymbol{u} \in \mathcal{W}$ by exp-concave and strongly convex functions. These are both requirements on the curvature of f_t that are stronger than mere convexity: α-exp-concavity of f for $\alpha > 0$ means that $e^{-\alpha f}$ is concave or, equivalently, that $\nabla^2 f \succeq \alpha \nabla f \nabla f^\intercal$; α-strong convexity means that $\nabla^2 f \succeq \alpha \boldsymbol{I}$. We see that α-strong convexity implies $(\alpha/\|\nabla f\|_2^2)$-exp-concavity. The rate $O(d \log T)$ is also what we would expect by summing the asymptotic offline rate obtained by ridge regression on the squared loss (Srebro et al., 2010, Section 5.2), which is exp-concave. Our extension to $a > 1$ is technically a minor step, but it makes the condition much more liberal, because it may then also be satisfied by functions that do *not* have any curvature. For example, suppose that $f_t = f$ is a fixed convex function that does not change with t. Then, when $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}} f(\boldsymbol{u})$ is the offline minimizer, we have $(\boldsymbol{u}^* - \boldsymbol{w})^\intercal \nabla f(\boldsymbol{w}) \in [-G'D', 0]$, so that

$$\begin{aligned} f(\boldsymbol{u}^*) - f(\boldsymbol{w}) \geq& (\boldsymbol{u}^* - \boldsymbol{w})^\intercal \nabla f(\boldsymbol{w}) \\ \geq& 2(\boldsymbol{u}^* - \boldsymbol{w})^\intercal \nabla f(\boldsymbol{w}) + \frac{1}{D'G'}\left((\boldsymbol{u}^* - \boldsymbol{w})^\intercal \nabla f(\boldsymbol{w})\right)^2, \end{aligned}$$

where the first inequality uses only convexity of f. Thus condition (5.3.1) is satisfied by *any fixed convex function*, even if it does not have any curvature at all, with $a = 2$ and $b = 1/(G'D')$.

Bernstein Stochastic Gradients The possibility of getting fast rates even without any curvature is intriguing, because it goes beyond the usual strong convexity or exp-concavity conditions. In the online setting, the case of fixed functions $f_t = f$ seems rather restricted, however, and may in fact be handled by offline optimization methods. We therefore seek to loosen this requirement by replacing it by a stochastic condition on the distribution of the functions f_t. The relation between variance bounds like Theorem 19 and fast rates in the stochastic setting is studied in depth by (Koolen et al., 2016), who obtain fast rate results both in expectation and in probability. Here we provide a direct proof only for the expected regret, which allows a simplified analysis.

Suppose the functions f_t are independent and identically distributed (i.i.d.), with common distribution $\mathbb{P}$. Then we say that the gradients satisfy the (B, β)-*Bernstein condition* with respect to the stochastic optimum $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u} \in \mathcal{W}} \mathbb{E}_{f \sim \mathbb{P}}[f(\boldsymbol{u})]$ if for all $\boldsymbol{w} \in \mathcal{W}$.

$$(\boldsymbol{w} - \boldsymbol{u}^*)^\intercal \mathop{\mathbb{E}}_{f} [\nabla f(\boldsymbol{w}) \nabla f(\boldsymbol{w})^\intercal] (\boldsymbol{w} - \boldsymbol{u}^*) \;\leq\; B\Big((\boldsymbol{w} - \boldsymbol{u}^*)^\intercal \mathop{\mathbb{E}}_{f} [\nabla f(\boldsymbol{w})]\Big)^\beta. \quad (5.3.2)$$

This is an instance of the well-known Bernstein condition from offline statistical learning (Bartlett and Mendelson, 2006; Van Erven et al., 2015), applied to the linearized excess loss $(\boldsymbol{w} - \boldsymbol{u}^*)^\intercal \nabla f(\boldsymbol{w})$. As shown in Section 5.14, imposing the condition for the linearized excess loss is a weaker requirement than imposing it for the original excess loss $f(\boldsymbol{w}) - f(\boldsymbol{u}^*)$.

Theorem 21. *If the gradients satisfy the (B, β)-Bernstein condition for $B > 0$ and $\beta \in (0, 1]$ with respect to $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u} \in \mathcal{W}} \mathbb{E}_{f \sim \mathbb{P}}[f(\boldsymbol{u})]$, then any method with regret bound* (5.1.1) *incurs expected regret*

$$\mathbb{E}[R_T^{\boldsymbol{u}^*}] = O\left((Bd \ln T)^{1/(2-\beta)} T^{(1-\beta)/(2-\beta)} + d \ln T\right).$$

For $\beta = 1$, the rate becomes $O(d \ln T)$, just like for fixed functions, and for smaller β it is in between logarithmic and $O(\sqrt{dT})$. For instance, the hinge loss on the unit ball with i.i.d. data satisfies the Bernstein condition with $\beta = 1$, which implies an $O(d \log T)$ rate. (See Section 5.10.4.) It is common to add ℓ_2-regularization to the hinge loss to make it strongly convex, but this example shows that that is not necessary to get logarithmic regret.

5.4 Full Matrix Version of the MetaGrad Algorithm

In this section, we explain the full matrix version of the MetaGrad algorithm: METAGRAD$^{\text{FULL}}$. Computationally more efficient extensions follow in Section 5.5. METAGRAD$^{\text{FULL}}$ will be defined by means of the following *surrogate loss* $\ell_t^\eta(\boldsymbol{u})$:

$$\ell_t^\eta(\boldsymbol{u}) \;\coloneqq\; \eta(\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t + \big(\eta(\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t\big)^2. \quad (5.4.1)$$

This surrogate loss consists of a linear and a quadratic part, whose relative importance is controlled by a learning rate parameter $\eta > 0$. The sum of the quadratic parts is what appears in the regret bound of Theorem 19. They may be viewed as causing a "time-varying regularizer" (Orabona et al., 2015b) or "temporal adaptation of the proximal function" (Duchi et al., 2011).

METAGRAD$^{\text{FULL}}$ is a two-level hierarchical construction: at the top is a main controller, shown in Algorithm 10, which manages multiple η-experts, shown in Algorithm 11. Each η-expert produces predictions for the surrogate loss ℓ_t^η with its own value of η, and the controller is responsible for learning the best η by starting and stopping multiple η-experts on demand, and aggregating their predictions.

Algorithm 10 Full MetaGrad: Controller

1: **for** $t = 1, 2, \ldots$ **do**
2: Receive domain $\mathcal{W}_t$
3: Start and stop η-experts to manage active set $\mathcal{A}_t$ (see (5.4.2)).
Give newly started η-experts weight $p_t(\eta) = 1$.
4: **if** Nobody active: $\mathcal{A}_t = \emptyset$ **then**
5: Predict $\boldsymbol{w}_t = \mathbf{0}$ ▷ *Make a default prediction*
6: **else**
7: Have active η-experts project onto $\mathcal{W}_t$
8: Collect prediction $\boldsymbol{w}_t^\eta$ for every active η-expert
9: Predict

$$\boldsymbol{w}_t \;=\; \frac{\sum_{\eta\in\mathcal{A}_t} p_t(\eta)\eta\boldsymbol{w}_t^\eta}{\sum_{\eta\in\mathcal{A}_t} p_t(\eta)\eta}$$

10: **end if**
11: Receive gradient $\boldsymbol{g}_t = \nabla f_t(\boldsymbol{w}_t)$ and range bound b_t (see (5.2.1))
12: Update every active η-expert with unclipped surrogate loss ℓ_t^η
13: **if** No reset needed after round t (see (5.4.3)) **then**
14: Update based on the clipped surrogate losses (see (5.4.4)):

$$p_{t+1}(\eta) = \frac{p_t(\eta)\exp\bigl(-\bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)\bigr)}{\sum_{\eta\in\mathcal{A}_t} p_t(\eta)\exp\bigl(-\bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)\bigr)}\bigl(\textstyle\sum_{\eta\in\mathcal{A}_t} p_t(\eta)\bigr) \qquad \text{for all } \eta\in\mathcal{A}_t.$$

15: **else**
16: Set $p_{t+1}(\eta) = 1$ for all $\eta \in \mathcal{A}_t$ ▷ *Reset*
17: **end if**
18: **end for**

Controller Online learning of the best learning rate η is notoriously difficult because the regret is non-monotonic over rounds and may have multiple local minima as a function of η (see (Koolen et al., 2014) for a study in the expert setting). The standard technique is therefore to derive a monotonic upper bound on the regret and tune the learning rate optimally *for the bound*. In contrast, our approach, inspired by the approach for combinatorial games of Koolen and Van Erven (2015, Section 4), is to weigh the different η depending on their empirical performance using exponential weights with sleeping experts (line 14), except that

in the predictions the weights of the η-experts are *tilted* by their learning rates (line 9), having the effect of giving a larger weight to larger η. Although we provide a formal analysis of the regret, the controller algorithm does not depend on the outcome of this analysis, so any slack in our bounds does not feed back into the algorithm.

To be able to adapt to the norms of the gradients, the controller maintains a finite grid $\mathcal{A}_t$ of active learning rates η, which is dynamically adjusted over time:

$$\mathcal{A}_t = \begin{cases} \emptyset & \text{while } B_{t-1} = 0, \\ \{2^i \mid i \in \mathbb{Z}\} \cap \left(\frac{1}{4\left(\sum_{s=1}^{t-1} b_s \frac{B_{s-1}}{B_s} + B_{t-1}\right)}, \frac{1}{4B_{t-1}} \right] & \text{afterwards.} \end{cases} \tag{5.4.2}$$

Using that $b_s \frac{B_{s-1}}{B_s} \leq B_{t-1}$, it can be seen that the number of active learning rates never exceeds $|\mathcal{A}_t| \leq \lceil \log_2 T \rceil$. In the first two rounds, or if there is a sudden enormous gradient such that B_{t-1} dwarfs $\sum_{s=1}^{t-1} b_s B_{s-1}/B_s$, it may also happen that $\mathcal{A}_t$ is empty, which signals that all previous rounds were negligible compared to the last round. In such cases the controller decides it has not yet learned anything, and makes a default prediction: $\boldsymbol{w}_t = \mathbf{0}$.

There are two further mechanisms to deal with extreme changes in the size of the gradients. The first mechanism is that extremely large gradients may trigger a *reset* of the controller's weights on η-experts. This splits the controller's learning process into epochs. When running in an epoch starting at time $\tau + 1$, a reset and new epoch will be triggered after the first round t such that

$$B_t > B_\tau \sum_{s=1}^{t} \frac{b_s}{B_s}. \tag{5.4.3}$$

As the sum on the right-hand side will typically grow linearly in t, we only expect a reset to occur when the effective size of the gradients grows by more than a factor t compared to the largest size seen before the start of the epoch. This should normally be very rare except perhaps for a few initial rounds when t is still small.

The second mechanism to protect against extreme gradients is that the controller measures performance of the experts by a *clipped* version of their corresponding surrogate losses:

$$\bar{\ell}_t^\eta(\boldsymbol{u}) \;\coloneqq\; \eta(\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \bar{\boldsymbol{g}}_t + \left(\eta(\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \bar{\boldsymbol{g}}_t\right)^2, \tag{5.4.4}$$

which are based on the clipped gradients

$$\bar{\boldsymbol{g}}_t \;\coloneqq\; \frac{B_{t-1}}{B_t} \boldsymbol{g}_t.$$

This is a trick first used by Cutkosky (2019), which makes the effective sizes of the gradients predictable one round in advance: $\max_{\boldsymbol{u}\in\mathcal{W}_t} |\boldsymbol{u}^\intercal \bar{\boldsymbol{g}}_t| \le B_{t-1}$.

Algorithm 11 Full MetaGrad: η-Expert

Input: Learning rate $\eta > 0$, estimate $\hat{D} > 0$ of comparator norm $\|\boldsymbol{u}\|_2$, activation round $a \equiv a^\eta$

1: Initialize $\widetilde{\boldsymbol{w}}_a^\eta = \boldsymbol{0}, \Sigma_a^\eta = \hat{D}^2\boldsymbol{I}$ and $\boldsymbol{\Lambda}_a^\eta = \frac{1}{\hat{D}^2}\boldsymbol{I}$
2: **for** $t = a, a+1, \ldots$ **do**
3: Project $\boldsymbol{w}_t^\eta = \arg\min_{\boldsymbol{u}\in\mathcal{W}_t} (\boldsymbol{u} - \widetilde{\boldsymbol{w}}_t^\eta)^\intercal \boldsymbol{\Lambda}_t^\eta (\boldsymbol{u} - \widetilde{\boldsymbol{w}}_t^\eta)$
4: Predict $\boldsymbol{w}_t^\eta$
5: Observe gradient $\boldsymbol{g}_t = \nabla f_t(\boldsymbol{w}_t)$ ▷ *Gradient at* controller *prediction* $\boldsymbol{w}_t$
6: Update:

$$\Sigma_{t+1}^\eta = \Sigma_t^\eta - \frac{2\eta^2(\Sigma_t^\eta \boldsymbol{g}_t)(\boldsymbol{g}_t^\intercal \Sigma_t^\eta)}{1 + 2\eta^2 \boldsymbol{g}_t^\intercal \Sigma_t^\eta \boldsymbol{g}_t}$$

$$\boldsymbol{\Lambda}_{t+1}^\eta = \boldsymbol{\Lambda}_t^\eta + \boldsymbol{g}_s \boldsymbol{g}_s^\intercal$$

$$\widetilde{\boldsymbol{w}}_{t+1}^\eta = \boldsymbol{w}_t^\eta - (1 + 2\eta(\boldsymbol{w}_t^\eta - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t)\, \eta \Sigma_{t+1}^\eta \boldsymbol{g}_t$$

7: **end for**

η-Experts Each η-expert is active for a single contiguous sequence of rounds for which $\eta \in \mathcal{A}_t$. Upon activation, its job is to issue predictions $\boldsymbol{w}_t^\eta \in \mathcal{W}_t$ for the (unclipped) surrogate loss ℓ_t^η that achieve small regret compared to any $\boldsymbol{u} \in \bigcap_{t:\eta\in\mathcal{A}_t} \mathcal{W}_t$. This is a standard online convex optimization task with a quadratic loss function and time-varying domain, which we assume is non-empty. We use continuous exponential weights with a Gaussian prior, which is a standard approach for quadratic losses (Vovk, 2001), because the corresponding posterior exponential weights distribution is also Gaussian with mean $\boldsymbol{w}_t^\eta$ and covariance matrix $\Sigma_t^\eta = \left(\frac{1}{\hat{D}^2}\boldsymbol{I} + 2\eta^2 \sum_{s=a}^t \boldsymbol{g}_s \boldsymbol{g}_s^\intercal\right)^{-1}$. Algorithm 11 presents the update equations in a computationally efficient form. To avoid inverting Σ_t^η, it maintains its inverse $\boldsymbol{\Lambda}_t^\eta = (\Sigma_t^\eta)^{-1}$ separately. For a recent overview of continuous exponential weights see Van der Hoeven et al. (2018). It can be seen that our η-expert algorithm is nearly identical to Online Newton Step (ONS) (Hazan et al., 2007), which is not surprising because ONS is minimizing a quadratic loss that is nearly identical to our ℓ_t^η. The differences are that each η-expert receives the controller's gradient $\boldsymbol{g}_t = \nabla f_t(\boldsymbol{w}_t)$ instead of its own $\nabla f_t(\boldsymbol{w}_t^\eta)$, and that an additional term $(1 + 2\eta(\boldsymbol{w}_t^\eta - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t)$ in line 6 adjusts for the difference between the η-expert's parameters $\boldsymbol{w}_t^\eta$ and the controller's parameters $\boldsymbol{w}_t$. MetaGrad is therefore a bona fide first-order algorithm that only accesses f_t through $\boldsymbol{g}_t$. We also note that we have chosen the Greedy projections version that iteratively updates and projects (see line 6). One might

alternatively consider the Lazy Projection version (as in Zinkevich (2004); Nesterov (2009); Xiao (2010)) that forgets past projections when updating on new data. Since projections are typically computationally expensive, we have opted for the Greedy projection version, which we expect to project less often, since a projected point seems less likely to update to a point outside of the domain than an unprojected point.

5.4.1 Practical Considerations

Although METAGRAD$^{\text{FULL}}$ is adaptive to the maximum effective size of the gradients B_T, its performance degrades when B_T becomes too large. In applications, it is therefore important that the domain $\mathcal{W}_t$ is small enough along the direction of $\boldsymbol{g}_t$ to keep the effective gradient size b_t under control.

It is further required to choose the hyperparameter $\hat{D}$, which is an estimate of the ℓ_2-norm of the comparator $\boldsymbol{u}$. Theorem 23 quantifies the trade-off between underestimating and overestimating this parameter. Note that overestimating $\|\boldsymbol{u}\|_2$ only incurs a logarithmic penalty, so it is less expensive to use a too large value rather than a too small value.

Finally, we note that there is no gain in pre-processing the data by scaling all gradients by a fixed constant factor, since the regret bound in Theorem 23 is scale-free. In fact, the METAGRAD$^{\text{FULL}}$ algorithm is almost invariant under such rescaling, except for the term $\{2^i \mid i \in \mathbb{Z}\}$ in the definition of $\mathcal{A}_t$. If one wants to make the algorithm fully invariant under rescaling, this term may be replaced by $\{2^i/B_\tau \mid i \in \mathbb{Z}\}$, where τ is the first round that $B_\tau > 0$. Or, equivalently, one may replace all gradients by $\boldsymbol{g}_t/B_\tau$ for $t > \tau$. Since we do not expect any noticeable difference in performance from this modification, we have left it out.

Run Time The run time of METAGRAD$^{\text{FULL}}$ is dominated by computations for the η-experts. Ignoring the projection step, an η-expert takes $O(d^2)$ time to update. If there are at most k' active η-experts in any round, this makes the overall computational effort $O(k'd^2)$, both in time per round and in memory. Since $|\mathcal{A}_t| \leq \lceil \log_2 T \rceil$, it is guaranteed that $k' \leq 30$ as long as $T \leq 10^9$. We note that all η-experts share the same gradient $\boldsymbol{g}_t$, which is only computed once. We remark that a potential speed-up is possible by running the η-experts in parallel. If the factor k' is still considered too large, it is possible to reduce the size of $|\mathcal{A}_t|$ by spacing the learning rates by a factor larger than 2, at the cost of a worse constant in the regret bound.

In addition, each η-expert may incur the cost of a projection, which depends on the shape of the domain $\mathcal{W}_t$. To get a sense for the projection cost, we consider

the Euclidean ball as a typical example. If the matrix Σ_t^η were diagonal, we could project to any desired precision using a few iterations of Newton's method. Since each such iteration takes $O(d)$ time, this would be affordable. But for the non-diagonal Σ_t^η that occur in the algorithm, we first need to reduce to the diagonal case by a basis transformation, which takes $O(d^3)$ to compute using a singular value decomposition. We therefore see that the projection dwarfs the other run time by an order of magnitude. This has motivated Luo et al. (2017) to define a different domain (see Section 5.5.1), for which projections can be computed in closed form with $O(d)$ computation steps. In this case, the computation for the projections is negligible and the total computational complexity is $O(d^2)$ per round. We refer to Duchi et al. (2011) for examples of how to compute projections for various other domains $\mathcal{W}_t$.

5.5 Faster Extension Algorithms

As discussed above, METAGRAD$^{\text{FULL}}$ requires at least $O(d^2)$ computation per round, which makes it slow in high dimensions. We therefore present two extensions to speed up the algorithm. The first is a straightforward adaption of the sketching approach of Luo et al. (2017), which we apply to approximate the matrix Σ_t^η in the η-experts. This reduces the computation per round to $O(kd)$, where k is a hyper-parameter that determines the sketch size. The second extension is to run a separate copy of the algorithm per dimension, which was inspired by the diagonal version of AdaGrad (Duchi et al., 2011). This requires $O(d)$ computation per round.

5.5.1 Sketched MetaGrad with Closed-form Projections

In this section, we are mixing matrices of different dimensions. The identity matrix $\boldsymbol{I}_d \in \mathbb{R}^d$ and the all-zeros matrix $\mathbf{0}_{a\times b} \in \mathbb{R}^{a\times b}$ are therefore annotated with subscripts to make their dimensions explicit. To simplify notation, we further assume without loss of generality that the η-experts are started in round $a^\eta = 1$.

Luo et al. (2017) develop several sketching approaches for Online Newton Step, which transfer directly to our η-experts. They combine these with a computationally efficient choice of the domain that applies to loss functions of the form $f_t(\boldsymbol{w}) = h_t(\boldsymbol{w}^\intercal \boldsymbol{x}_t)$, where the input vectors $\boldsymbol{x}_t \in \mathbb{R}^d$ are assumed to be known at the start of round t, but the convex functions $h_t : \mathbb{R} \to \mathbb{R}$ are not. They then choose the domain to be

$$\mathcal{W}_t = \{\boldsymbol{w} : |\boldsymbol{w}^\intercal \boldsymbol{x}_t| \leq C\} \qquad \text{for a fixed constant } C. \tag{5.5.1}$$

Let $\boldsymbol{G}_t = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_t)^\intercal \in \mathbb{R}^{t \times d}$, such that $\Sigma_{t+1}^\eta = (\frac{1}{\hat{D}^2}\boldsymbol{I}_d + 2\eta^2 \boldsymbol{G}_t^\intercal \boldsymbol{G}_t)^{-1}$. The idea of sketching is to replace Σ_{t+1}^η by an approximation

$$\widetilde{\Sigma}_{t+1}^\eta = \left(\tfrac{1}{\hat{D}^2}\boldsymbol{I}_d + 2\eta^2 \boldsymbol{S}_t^\intercal \boldsymbol{S}_t\right)^{-1},$$

where $\boldsymbol{S}_t \in \mathbb{R}^{k \times d}$ for a given *sketch size* $k \leq d$, so that $\boldsymbol{S}_t^\intercal \boldsymbol{S}_t$ has rank at most k. Abbreviating $\hat{\boldsymbol{g}}_t = (1 + 2\eta(\boldsymbol{w}_t^\eta - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t)\,\eta \boldsymbol{g}_t$, we then need to compute

$$\begin{aligned} \boldsymbol{w}_t^\eta &= \operatorname*{arg\,min}_{\boldsymbol{u} \in \mathcal{W}_t} (\boldsymbol{u} - \widetilde{\boldsymbol{w}}_t^\eta)^\intercal (\widetilde{\Sigma}_t^\eta)^{-1} (\boldsymbol{u} - \widetilde{\boldsymbol{w}}_t^\eta) && \text{(projection)} \\ \widetilde{\boldsymbol{w}}_{t+1}^\eta &= \boldsymbol{w}_t^\eta - \widetilde{\Sigma}_{t+1}^\eta \hat{\boldsymbol{g}}_t. && \text{(update)} \end{aligned}$$

The key to an efficient implementation of these steps is to rewrite $\widetilde{\Sigma}_{t+1}^\eta$ using the Woodbury identity (Golub and Van Loan, 2012):

$$\widetilde{\Sigma}_{t+1}^\eta = \hat{D}^2(\boldsymbol{I}_d - 2\eta^2 \boldsymbol{S}_t^\intercal (\tfrac{1}{\hat{D}^2}\boldsymbol{I}_k + 2\eta^2 \boldsymbol{S}_t \boldsymbol{S}_t^\intercal)^{-1} \boldsymbol{S}_t) = \hat{D}^2(\boldsymbol{I}_d - 2\eta^2 \boldsymbol{S}_t^\intercal \boldsymbol{H}_t^\eta \boldsymbol{S}_t),$$

where we have introduced the abbreviation $\boldsymbol{H}_t^\eta = (\frac{1}{\hat{D}^2}\boldsymbol{I}_k + 2\eta^2 \boldsymbol{S}_t \boldsymbol{S}_t^\intercal)^{-1}$. Let $\mathrm{s}_C(y) = \operatorname{sign}(y) \max\{|y| - C, 0\}$. By Lemma 1 of Luo et al. (2017), the projection step then becomes

$$\boldsymbol{w}_t^\eta = \widetilde{\boldsymbol{w}}_t^\eta - \frac{\mathrm{s}_C(\boldsymbol{x}_t^\intercal \widetilde{\boldsymbol{w}}_t^\eta)}{(\boldsymbol{x}_t^\intercal \boldsymbol{x}_t - 2\eta^2 \boldsymbol{x}_t^\intercal \boldsymbol{S}_t^\intercal \boldsymbol{H}_t^\eta \boldsymbol{S}_t \boldsymbol{x}_t)} (\boldsymbol{x}_t - 2\eta^2 \boldsymbol{S}_t^\intercal \boldsymbol{H}_t^\eta \boldsymbol{S}_t \boldsymbol{x}_t),$$

and the update step can be written as

$$\widetilde{\boldsymbol{w}}_{t+1}^\eta = \boldsymbol{w}_t^\eta - \hat{D}^2(\hat{\boldsymbol{g}}_t - 2\eta^2 \boldsymbol{S}_t^\intercal \boldsymbol{H}_t^\eta \boldsymbol{S}_t \hat{\boldsymbol{g}}_t).$$

Assuming that $\boldsymbol{S}_t$ and $\boldsymbol{H}_t^\eta$ can be efficiently maintained, the operations involving $\boldsymbol{S}_t \boldsymbol{x}_t$ or $\boldsymbol{S}_t \hat{\boldsymbol{g}}_t$ require $O(kd)$ computation time and matrix-vector products with $\boldsymbol{H}_t^\eta$ can be performed in $O(k^2)$ time. As noted by Luo et al. (2017), both of these are only a factor k more than the $O(d)$ time required by first-order methods. They describe two sketching techniques to maintain $\boldsymbol{S}_t$ and $\boldsymbol{H}_t^\eta$, each requiring $O(kd)$ storage and $O(kd)$ average computation time per round. The first technique is based on Frequent Directions (FD) sketching; the other one on Oja's algorithm. We adopt the FD approach, which comes with a guaranteed bound on the regret. Luo et al. (2017) further develop an extension of FD for sparse gradients, and yet another option would have been the Robust Frequent Directions sketching method of Luo et al. (2019).

Frequent Directions Sketching

Some sketching approaches are randomized, but Frequent Directions sketching (Ghashami et al., 2016) is a deterministic method. The simplest version (Luo et al., 2017, Algorithm 2) performs a singular value decomposition (SVD) of $\boldsymbol{S}_t$ every round at the cost of $O(k^2 d)$ computation time, but there also exists a refined epoch-based version which only performs an SVD once per epoch. Each epoch takes m rounds and $k = 2m$, leading to an average runtime of $O(kd)$ per round. We describe here the epoch version, adapted from Algorithm 6 of Luo et al. (2017) and summarized in Algorithm 12.

Algorithm 12 Frequent Directions Sketching

Initialize $\boldsymbol{S}_0 = \boldsymbol{0}_{2m\times d}$, and $\boldsymbol{H}_0^\eta = \hat{D}^2 \boldsymbol{I}_{2m}$.
for $t = 1, 2, \ldots$ **do**
  Let $\tau = t \bmod m$ and insert $\boldsymbol{g}_t^\intercal$ in the $(m+\tau)$-th row of $\boldsymbol{S}_{t-1}$ to obtain $\tilde{\boldsymbol{S}}$.
  if $\tau \neq 0$ **then**
    Set $\boldsymbol{S}_t = \tilde{\boldsymbol{S}}$.
    Let $\boldsymbol{e} \in \mathbb{R}^{2m}$ be the basis vector in direction $m+\tau$ and $\boldsymbol{q} = 2\eta^2(\tilde{\boldsymbol{S}}\boldsymbol{g}_t - \frac{\boldsymbol{g}_t^\intercal \boldsymbol{g}_t}{2}\boldsymbol{e})$.
    Update $\boldsymbol{H}_t^\eta = \tilde{\boldsymbol{H}} - \frac{\tilde{\boldsymbol{H}}\boldsymbol{e}\boldsymbol{q}^\intercal \tilde{\boldsymbol{H}}}{1+\boldsymbol{q}^\intercal \tilde{\boldsymbol{H}}\boldsymbol{e}}$, where $\tilde{\boldsymbol{H}} = \boldsymbol{H}_{t-1}^\eta - \frac{\boldsymbol{H}_{t-1}^\eta \boldsymbol{q}\boldsymbol{e}^\intercal \boldsymbol{H}_{t-1}^\eta}{1+\boldsymbol{e}^\intercal \boldsymbol{H}_{t-1}^\eta \boldsymbol{q}}$.
  else
    From the SVD of $\tilde{\boldsymbol{S}}$, compute the top-m singular values $\sigma_1 \geq \cdots \geq \sigma_m$ and corresponding right-singular vectors as $\boldsymbol{V} \in \mathbb{R}^{d\times m}$.
    Set $\boldsymbol{S}_t = \operatorname{diag}(\sigma_1^2 - \sigma_m^2, \ldots, \sigma_m^2 - \sigma_m^2)^{1/2} \boldsymbol{V}^\intercal$.
    Set $\boldsymbol{H}_t^\eta = \operatorname{diag}(\frac{1}{\hat{D}^{-2}+2\eta^2(\sigma_1^2-\sigma_m^2)}, \ldots, \frac{1}{\hat{D}^{-2}+2\eta^2(\sigma_m^2-\sigma_m^2)}, \frac{1}{\hat{D}^{-2}}, \ldots, \frac{1}{\hat{D}^{-2}})$.
  end if
end for

Recall that $\boldsymbol{S}_t^\intercal \boldsymbol{S}_t$ is an approximation of $\boldsymbol{G}_t^\intercal \boldsymbol{G}_t$. At the start of each epoch, we have the invariant that only the first $m-1$ rows of $\boldsymbol{S}_t$ contribute to this approximation and the remaining $m+1$ rows are filled with zeros. During the τ-th round in any epoch we first add the incoming gradient $\boldsymbol{g}_t^\intercal$ to row $m+\tau$ of $\boldsymbol{S}_{t-1}$ to obtain an intermediate result $\tilde{\boldsymbol{S}}$. If we are not yet in the last round of the epoch (i.e. $\tau < m$), then we simply set $\boldsymbol{S}_t = \tilde{\boldsymbol{S}}$, and we use that

$$(\boldsymbol{H}_t^\eta)^{-1} = (\boldsymbol{H}_{t-1}^\eta)^{-1} + \boldsymbol{q}\boldsymbol{e}^\intercal + \boldsymbol{e}\boldsymbol{q}^\intercal,$$

where $\boldsymbol{e} \in \mathbb{R}^{2m}$ is the basis vector in direction $m + \tau$ and $\boldsymbol{q} = 2\eta^2(\tilde{\boldsymbol{S}}\boldsymbol{g}_t - \frac{\boldsymbol{g}_t^\intercal \boldsymbol{g}_t}{2}\boldsymbol{e})$. It follows that we can compute $\boldsymbol{H}_t^\eta$ from $\boldsymbol{H}_{t-1}^\eta$ using two rank-one updates with the Sherman-Morrison formula:

$$\boldsymbol{H}_t^\eta = \tilde{\boldsymbol{H}} - \frac{\tilde{\boldsymbol{H}}\boldsymbol{e}\boldsymbol{q}^\intercal\tilde{\boldsymbol{H}}}{1 + \boldsymbol{q}^\intercal\tilde{\boldsymbol{H}}\boldsymbol{e}}, \text{ where } \tilde{\boldsymbol{H}} = \boldsymbol{H}_{t-1}^\eta - \frac{\boldsymbol{H}_{t-1}^\eta\boldsymbol{q}\boldsymbol{e}^\intercal\boldsymbol{H}_{t-1}^\eta}{1 + \boldsymbol{e}^\intercal\boldsymbol{H}_{t-1}^\eta\boldsymbol{q}}.$$

Otherwise, if we are in the last round of the epoch (i.e. $\tau = m$), the invariant is restored by eigen decomposing $\tilde{\boldsymbol{S}}^\intercal\tilde{\boldsymbol{S}}$ into $\boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^\intercal$, where $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_{2m})$ contains the potentially non-zero eigenvalues in non-decreasing order $\lambda_1 \geq \cdots \geq \lambda_{2m}$ and the columns of $\boldsymbol{W} \in \mathbb{R}^{d\times 2m}$ contain the corresponding eigenvectors. Then we set $\boldsymbol{S}_t = \operatorname{diag}(\lambda_1 - \lambda_m, \ldots, \lambda_m - \lambda_m, 0, \ldots, 0)^{1/2}\boldsymbol{W}^\intercal$. Since the rows of $\boldsymbol{S}_t$ are now orthogonal,

$$\begin{aligned}\boldsymbol{H}_t^\eta &= (\tfrac{1}{\hat{D}^2}\boldsymbol{I}_{2m} + 2\eta^2\boldsymbol{S}_t\boldsymbol{S}_t^\intercal)^{-1} \\ &= \operatorname{diag}\Big(\frac{1}{\hat{D}^{-2} + 2\eta^2(\lambda_1 - \lambda_m)}, \ldots, \frac{1}{\hat{D}^{-2} + 2\eta^2(\lambda_m - \lambda_m)}, \frac{1}{\hat{D}^{-2}}, \ldots, \frac{1}{\hat{D}^{-2}}\Big)\end{aligned}$$

is a diagonal matrix.

Implementation Details When implementing the FD procedure, we can calculate the eigen decomposition of $\tilde{\boldsymbol{S}}^\intercal\tilde{\boldsymbol{S}}$ via an SVD of $\tilde{\boldsymbol{S}}$, which can be performed in $O(m^2 d)$ computation steps. The eigenvalues λ_i then correspond to the squared singular values σ_i^2 of $\tilde{\boldsymbol{S}}$, and $\boldsymbol{W}$ contains the corresponding right-singular vectors. In fact, we only need the top-m singular values and the corresponding m right-singular vectors $\boldsymbol{V} \in \mathbb{R}^{d\times m}$ to compute $\boldsymbol{S}_t = \operatorname{diag}(\lambda_1 - \lambda_m, \ldots, \lambda_m - \lambda_m, 0, \ldots, 0)^{1/2}\boldsymbol{W}^\intercal = \operatorname{diag}(\sigma_1^2 - \sigma_m^2, \ldots, \sigma_m^2 - \sigma_m^2)^{1/2}\boldsymbol{V}^\intercal$.

We further observe that, since $\boldsymbol{S}_t$ does not depend on η, we only need to compute it once when sketching for multiple η-experts with different learning rates η. The matrix $\boldsymbol{H}_t^\eta$, however, does need to be computed for each η separately.

Practical Considerations

Sketching introduces an extra hyper-parameter $k = 2m$, which controls the sketch size. In theory, larger k provides a better approximation of the full version of MetaGrad, at the cost of more computation.

5.5.2 Coordinate MetaGrad

Duchi et al. (2011) introduce a full and a diagonal version of their AdaGrad algorithm. The diagonal version, which is the version that is widely used in

applications, may be interpreted as running a copy of online gradient descent (Zinkevich, 2003) for each dimension separately, with a separate data-dependent tuning of the step size per dimension. This approach of running a separate copy per dimension can be applied to any online learning algorithm, and works out as follows.

We output a joint prediction $\boldsymbol{w}_t = (w_{t,1}, \ldots, w_{t,d})^\intercal$, where each $w_{t,i}$ is the output of the copy of the algorithm for dimension i. Each of these copies gets as inputs the 1-dimensional losses $f_{t,i}(w) = wg_{t,i}$, where $g_{t,i}$ is the i-th component of the joint gradient $\boldsymbol{g}_t = \nabla f_t(\boldsymbol{w}_t)$. This works because the linearized regret decomposes per dimension:

$$\sum_{t=1}^{T}(\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t = \sum_{i=1}^{d}\sum_{t=1}^{T}(f_{t,i}(w_{t,i}) - f_{t,i}(u_i)),$$

so our joint linearized regret is simply the sum of the linearized regrets per dimension.

One limitation of this approach, if we apply it as is, is that the domain cannot introduce dependencies between the dimensions, so we are limited to rectangular domains:

$$\mathcal{W}_t^{\text{rect}} = \{\boldsymbol{w} \in \mathbb{R}^d \mid a_{t,i} \leq w_i \leq z_{t,i} \text{ for } i = 1, \ldots, d\},$$

with our only freedom consisting of choosing the boundaries $a_{t,i}$ and $z_{t,i}$.

Practical Considerations

Running a copy of MetaGrad per dimension potentially introduces a separate hyperparameter $\hat{D}_i$ per dimension i. Like Duchi et al. (2011), we reduce the complexity of hyperparameter tuning by letting $\hat{D}_i = \hat{D}$ be the same for all dimensions. If no specific domain is required and the components of the gradients are approximately standardized, it is also generally sufficient to set the dimensions of the rectangular domain to $a_{t,i} = -q$ and $z_{t,i} = q$ for a fixed parameter q.

5.6 Analysis of the Full Matrix Version of MetaGrad

The high-level goal of MetaGrad is to deliver a tight data-dependent regret bound. Such bounds could be achieved in principle by existing algorithms, were their learning rate tuned certain a-priori unknown data-dependent quantities. The practical approach implemented in MetaGrad is to run multiple instances of a baseline

"η-expert" algorithm, each with different candidate tuning. A controller then aggregates these η-expert predictions and manages their lifetimes to always have the required tuning present.

The METAGRAD$^{\text{FULL}}$ η-experts are Exponentially Weighted Average forecasters starting from a Gaussian prior and taking in our quadratic surrogate losses. Their efficient implementation is a variant of Online Newton Step, where the losses are centred at the prediction of the controller instead of that of the η-expert. In turn, the controller is a specialists (also known as sleeping experts) algorithm to deal with the starting and retiring of η-experts. It is further designed to give a non-uniform regret guarantee, obtaining especially small regret when the best learning rate turns out to be high. Finally, our approach for adapting to the Lipschitz constant is speculative. Starting at zero, we monitor the implied Lipschitz constant of the incoming gradients. If it is increasing slowly, the controller is able to accommodate the overshoots in a lower-order term. If it makes a large jump, then the controller may need to reset. We do so by resetting the controller weights without changing the state of the affected η-experts.

5.6.1 Controller

Denote by $\mathcal{G} = \{2^i : i \in \mathbb{Z}\}$ and by a^η the starting time of an η-expert (for the exact definition of a^η see definition 3 in Section 5.11). Let us introduce the concept of expiration.

Definition 2. *We say that $\eta \in \mathcal{G}$ is* expired *after T rounds (or, equivalently, after round T) if $\eta > \frac{1}{4B_{T-1}}$.*

Note that expiration can be checked *before* the round happens (it is "predictable"). All learning rates used by Algorithm 10 by means of the active set $\mathcal{A}_t$ (5.4.2) are not expired. Also note the "lifecycle" of any fixed learning rate η. It starts inactive unexpired. Then it becomes active unexpired. And finally it expires, after which it loses all relevance.

For the controller, we prove that it behaviour approximates that of any η-expert not expired, when measured in the η surrogate loss (5.4.1).

Lemma 15 (Controller Surrogate Regret Bound)**.** *For any learning rate $\eta \in \mathcal{G}$ not expired after T rounds and any comparator $\boldsymbol{u} \in \bigcap_{t=1}^{T} \mathcal{W}_t$,* METAGRAD$^{\text{FULL}}$

ensures

$$R_t^\eta(\boldsymbol{u}) \;\leq\; \underbrace{\frac{1}{2} + 4\eta B_T}_{\text{small}} + \underbrace{2\ln\left\lceil 2\log_2\left(\sum_{t=1}^{T-1}\frac{b_t}{B_t}+1\right)\right\rceil_+}_{\text{specialist regret for epoch, } O(\ln\ln T)} + \underbrace{\sum_{t=a^\eta}^{T}\left(\ell_t^\eta(\boldsymbol{w}_t^\eta) - \ell_t^\eta(\boldsymbol{u})\right)}_{\ell^\eta\text{-regret of }\eta\text{-expert w.r.t. }\boldsymbol{u}}.$$

The proof is in Section 5.11. It follows the MetaGrad analysis of Mhammedi et al. (2019), including the range clipping technique due to Cutkosky (2019), and the reset technique of Mhammedi et al. (2019), which in particular ensures that whenever a reset occurs, the accumulated regret up until the *previous* reset is small. As such, we only have to pay for the controller regret for the last two epochs.

We further streamline the approach by using a standard specialists (sleeping experts) algorithm on a discrete grid of η-experts as our controller algorithm. Of note here is our use of the improper log-uniform prior. We also employ a slightly tightened measure b_t of the effective loss range.

To make further progress, we need to make use of the details of the η-experts.

5.6.2 Full η-Experts

Next we establish a $O(d\log T)$ regret bound in terms of the surrogate loss for each MetaGrad$^{\text{FULL}}$ η-expert. The η-experts implement Follow-the-Regularised-Leader with the quadratic losses ℓ_t^η and the squared Euclidean norm regulariser. Equivalently, we can see them as implementing the exponentially weighted average forecaster for the quadratic losses ℓ_t^η starting from a Gaussian prior. Algorithms for the specific quadratic loss arising in linear regression were designed and analysed by Vovk (2001). The general quadratic case goes back (at least) to Hazan et al. (2007), who unfortunately do not separate the analysis for general quadratic losses from the reduction of exp-concave losses to quadratics, even though these ideas are clearly present. The explicit analysis by van Erven and Koolen (2016) includes an unnecessary range restriction, which was subsequently removed by Van der Hoeven et al. (2018). As pointed out by Luo et al. (2017), the extension to time-varying domains is trivial.

Lemma 16 (Surrogate regret bound)**.** *Consider the* MetaGrad$^{\text{FULL}}$ *η-expert in Algorithm 11 with learning rate $\eta \leq \frac{1}{4B_T}$ starting from time a^η. Its surrogate regret after round $T \geq a^\eta$ w.r.t. any comparator $\boldsymbol{u} \in \bigcap_{t=a^\eta}^T \mathcal{W}_t$ is bounded by*

$$\sum_{t=a^\eta}^{T}\left(\ell_t^\eta(\boldsymbol{w}_t^\eta) - \ell_t^\eta(\boldsymbol{u})\right) \;\leq\; \frac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + \ln\det\left(\boldsymbol{I} + 2\eta^2\hat{D}^2\sum_{t=a^\eta}^{T}\boldsymbol{g}_t\boldsymbol{g}_t^\intercal\right).$$

The proof of Lemma 16 can be found in Section 5.12. We note that the condition on η in the lemma is slightly stricter than not being expired (Definition 2), which only requires $\eta \leq \frac{1}{4B_{T-1}}$. The reason is that the η-expert operates off the *unclipped* surrogate loss and gradients.

5.6.3 Composition (bounding the actual regret)

To complete the analysis of METAGRAD$^{\text{FULL}}$, we put the regret bounds for the controller and η-experts together. We then optimize η over the grid $\mathcal{G}$ to get our main result. For the purpose of this section, let us define the *essential horizon* and *gradient covariance* by

$$Q_T \coloneqq \sum_{t=1}^{T-1} \frac{b_t}{B_t} + 1 \qquad \text{and} \qquad \boldsymbol{F}_T \coloneqq \sum_{t=1}^{T} \boldsymbol{g}_t \boldsymbol{g}_t^\intercal.$$

Theorem 22 (Grid point regret)**.** *Let $\eta \in \mathcal{G}$ be such that $\eta \leq \frac{1}{4B_T}$. Then* METAGRAD$^{\text{FULL}}$ *guarantees that the linearized regret w.r.t. any comparator $\boldsymbol{u} \in \bigcap_{t=1}^{T} \mathcal{W}_t$ is at most*

$$\tilde{R}_T^{\boldsymbol{u}} \leq \eta V_T^{\boldsymbol{u}} + \frac{\ln\det\left(\boldsymbol{I} + 2\eta^2\hat{D}^2\boldsymbol{F}_T\right) + \frac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + 2\ln\lceil 2\log_2 Q_T\rceil_+ + \frac{1}{2}}{\eta} + 4B_T.$$

Proof. Combining the controller and η-expert surrogate regret bounds Lemma 15 and Lemma 16, we obtain

$$\sum_{t=1}^{T}\left(\ell_t^\eta(\boldsymbol{w}_t) - \ell_t^\eta(\boldsymbol{u})\right) \leq \frac{1}{2} + 4\eta B_T + 2\ln\left\lceil 2\log_2\left(\sum_{t=1}^{T-1}\frac{b_t}{B_t}+1\right)\right\rceil_+ + \frac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + \ln\det\left(\boldsymbol{I} + 2\eta^2\hat{D}^2\sum_{t=1}^{T}\boldsymbol{g}_t\boldsymbol{g}_t^\intercal\right).$$

The definition of the surrogate loss (5.4.1) gives $\ell_t^\eta(\boldsymbol{w}_t) - \ell_t^\eta(\boldsymbol{u}) = \eta(\boldsymbol{w}_t - \boldsymbol{u})^\intercal\boldsymbol{g}_t - \left(\eta(\boldsymbol{u}-\boldsymbol{w}_t)^\intercal\boldsymbol{g}_t\right)^2$ and the theorem follows by reorganising and dividing by η. □

The final step is to properly select the learning rate $\eta \in \mathcal{G}$ in the regret bound Theorem 22. This leads to our main result. The proof is in Section 5.13.

Theorem 23 (MetaGrad Full Regret Bound)**.** *For all $\boldsymbol{u} \in \bigcap_{t=1}^{T} \mathcal{W}_t$ the linearized regret of* METAGRAD$^{\text{FULL}}$ *is simultaneously bounded by*

$$\tilde{R}_T^{\boldsymbol{u}} \leq \frac{5}{2}\sqrt{V_T^{\boldsymbol{u}}\left(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + Z_T\right)} + 10B_T\left(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + Z_T\right) + 4B_T,$$

where $Z_T = \operatorname{rk}(\boldsymbol{F}_T) \ln\left(1 + \frac{\hat{D}^2 \sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2}{8B_T^2 \operatorname{rk}(\boldsymbol{F}_T)}\right) + 2\ln\lceil 2\log_2 T\rceil_+ + \frac{1}{2}$, *and by*

$$\tilde{R}_T^{\boldsymbol{u}} \leq \frac{5}{2}\sqrt{\left(V_T^{\boldsymbol{u}} + 2\hat{D}^2 \sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2\right)\left(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + Z_T'\right)} + 10B_T\left(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + Z_T'\right) + 4B_T,$$

where $Z_T' = 2\ln\lceil 2\log_2 T\rceil_+ + \frac{1}{2}$.

Since $\operatorname{rk}(\boldsymbol{F}_T) \leq d$, Theorem 19 follows when we assume that the diameter of the domain $\mathcal{W}_t$ and the gradient norms are both uniformly bounded over all rounds, which implies $Z_T = O(d\log(T/d))$. If the eigenvalues of $\boldsymbol{F}_T$ satisfy a decay condition, then a more refined bound is possible instead of the first term in the definition of Z_T, as can be seen from the proof.

5.7 Extensions for Faster MetaGrad Analysis

5.7.1 Sketching: Analysis

The analysis for the frequent directions sketching version of MetaGrad with sketch size $k = 2m$ proceeds like the analysis of the full matrix version, except that we obtain a different bound for the η-expert regret. This bound depends on the spectral decay of $\boldsymbol{F}_T = \boldsymbol{G}_T^\intercal \boldsymbol{G}_T = \sum_{t=1}^T \boldsymbol{g}_t \boldsymbol{g}_t^\intercal$. Let λ_i be the i-th eigenvalue of $\boldsymbol{G}_T^\intercal \boldsymbol{G}_T$ and define $\Omega_q = \sum_{i=q+1}^d \lambda_i$. Then the surrogate regret of the η-expert algorithm with FD sketching is bounded as follows:

Lemma 17. *Consider the sketching version of the MetaGrad η-expert algorithm with learning rate $\eta \leq \frac{1}{4B_T}$ starting from time a^η. Its surrogate regret after round $T \geq a^\eta$ w.r.t. any comparator $\boldsymbol{u} \in \bigcap_{t=a^\eta}^T \mathcal{W}_t$ is bounded by*

$$\sum_{t=a^\eta}^T (\ell_t^\eta(\boldsymbol{w}_t^\eta) - \ell_t^\eta(\boldsymbol{u})) \leq \frac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + \log(\det(\boldsymbol{I} + 2\eta^2\hat{D}^2 \boldsymbol{S}_T^\intercal \boldsymbol{S}_T)) + \frac{2\eta^2\hat{D}^2 m\Omega_q}{m-q}$$

for any $q = 0, \ldots, m-1$.

Compared to Lemma 16, we see that $\boldsymbol{G}_T^\intercal \boldsymbol{G}_T = \sum_{t=1}^T \boldsymbol{g}_t \boldsymbol{g}_t^\intercal$ in the logarithmic term has been replaced by its sketching approximation $\boldsymbol{S}_T^\intercal \boldsymbol{S}_T$. We therefore pay logarithmically for the top m directions, which are captured by the sketch. What we

lose is the rightmost term of order $O(\eta^2\Omega_q)$, which corresponds to the remaining $d-q$ directions that are not captured.

The proof of Lemma 17 is a straightforward adaptation of the proof of Theorem 3 by Luo et al. (2017). For the details, we refer to Chapter 4 of Deswarte (2018), with two minor remarks: the first is that Deswarte uses a slightly stricter bound on η, which allows him to bound $\frac{1}{2}\left(1+2\eta\left\langle \boldsymbol{w}_t-\boldsymbol{w}_t^\eta, \boldsymbol{g}_t\right\rangle\right)^2 \leq 1$, whereas we get an upper bound of 2 from (5.12.1) and therefore obtain a final result that is a factor of 2 larger. The other remark is that we have described the fast version of FD sketching, which corresponds to Algorithm 6 of Luo et al. (2017) instead of the simpler slow version in their Algorithm 2. They and Deswarte consider the slow version in their analysis, but this makes no difference for the proof because the fast algorithm satisfies the same guarantees (Ghashami et al., 2016).

Analogously with Theorem 22, we find:

Theorem 24 (Sketching Grid Point Regret)**.** *Let $\eta \in \mathcal{G}$ be such that $\eta \leq \frac{1}{4B_T}$. Then* METAGRAD$^{\text{SKETCH}}$ *guarantees that the linearized regret w.r.t. any comparator $\boldsymbol{u} \in \bigcap_{t=1}^T \mathcal{W}_t$ is at most*

$$\tilde{R}_T^{\boldsymbol{u}} \leq \frac{\ln\det\left(\boldsymbol{I}+2\eta^2\hat{D}^2\boldsymbol{S}_T^\intercal\boldsymbol{S}_T\right)+\frac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2+2\ln\lceil 2\log_2 Q_T\rceil_+ +\frac{1}{2}}{\eta} + \eta V_T^{\boldsymbol{u}} + \frac{2\eta\hat{D}^2 m\Omega_q}{m-q} + 4B_T,$$

for any $q = 0, \ldots, m-1$.

As shown in Section 5.13, optimizing η leads to the following final result:

Theorem 25 (MetaGrad Sketching Regret Bound)**.** *For all $\boldsymbol{u} \in \bigcap_{t=1}^T \mathcal{W}_t$ the linearized regret of* METAGRAD$^{\text{SKETCH}}$ *is simultaneously bounded by*

$$\tilde{R}_T^{\boldsymbol{u}} \leq \frac{5}{2}\sqrt{(V_T^{\boldsymbol{u}}+\frac{2\hat{D}^2 m\Omega_q}{m-q})(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2+Z_T)}+10B_T(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2+Z_T)+4B_T,$$

where $Z_T = \operatorname{rk}(\boldsymbol{S}_T^\intercal\boldsymbol{S}_T)\ln\left(1+\frac{\hat{D}^2\operatorname{Tr}(\boldsymbol{S}_T^\intercal\boldsymbol{S}_T)}{8B_T^2\operatorname{rk}(\boldsymbol{S}_T^\intercal\boldsymbol{S}_T)}\right)+2\ln\lceil 2\log_2 T\rceil_+ +\frac{1}{2}$*, and by*

$$\tilde{R}_T^{\boldsymbol{u}} \leq \frac{5}{2}\sqrt{\left(V_T^{\boldsymbol{u}}+2\hat{D}^2\sum_{t=1}^T\|\boldsymbol{g}_t\|_2^2+\frac{2\hat{D}^2 m\Omega_q}{m-q}\right)\left(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2+Z_T'\right)} + 10B_T\left(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2+Z_T'\right)+4B_T,$$

for any $q = 0, \ldots, m-1$, where $Z_T' = 2\ln\lceil 2\log_2 T\rceil_+ + \frac{1}{2}$.

Compared to Theorem 23, all occurrences of $\boldsymbol{G}_T^\intercal \boldsymbol{G}_T$ have been replaced by their sketched approximations: $\boldsymbol{F}_T = \boldsymbol{G}_T^\intercal \boldsymbol{G}_T$ has become $\boldsymbol{S}_T^\intercal \boldsymbol{S}_T$ everywhere, with $\mathrm{rk}(\boldsymbol{S}_T^\intercal \boldsymbol{S}_T) \leq 2m$, and $\sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2 = \mathrm{tr}(\boldsymbol{G}_T^\intercal \boldsymbol{G}_T)$ is now $\mathrm{tr}(\boldsymbol{S}_T^\intercal \boldsymbol{S}_T)$. We further see the additional term involving Ω_k, which corresponds to the directions not captured by the sketch.

5.7.2 MetaGrad Coordinate: Analysis

First, we define $b_{t,i} = |g_{t,i}| \max\{|a_{t,i}|, |z_{t,i}|\}$ and $B_{t,i} = \max_{s \leq t} b_{s,i}$. The analysis of the coordinate version of MetaGrad, which we denote by MetaGrad$^{\text{COOR}}$, is straightforward as we can simply apply the regret bound of MetaGrad$^{\text{FULL}}$ to each dimension. The formal statement can be found below.

Theorem 26. *Let $V_{T,i}^{u_i} = (u_i - w_{t,i})^2 g_{t,i}^2$. For any $\boldsymbol{u} \in \bigcap_{t=1}^T \mathcal{W}_t^{\text{rect}}$, the linearized regret of* MetaGrad$^{\text{COOR}}$ *is simultaneously bounded by*

$$\tilde{R}_T^{\boldsymbol{u}} \leq \sum_{i=1}^d \frac{5}{2} \sqrt{V_{T,i}^{u_i} \left(\tfrac{1}{2\hat{D}^2} u_i^2 + Z_{T,i}\right)} + 10 B_{T,i} \left(\tfrac{1}{2\hat{D}^2} u_i^2 + Z_{T,i}\right) + 4 B_{T,i},$$

where $Z_{T,i} = \ln\left(1 + \frac{\hat{D}^2 \sum_{t=1}^T g_{t,i}^2}{8 B_{T,i}^2}\right) + 2 \ln\lceil 2 \log_2 T \rceil + \frac{1}{2}$, *and by*

$$\begin{aligned} \tilde{R}_T^{\boldsymbol{u}} \leq \sum_{i=1}^d \frac{5}{2} \sqrt{\left(V_{T,i}^{u_i} + 2\hat{D}^2 \sum_{t=1}^T g_{t,i}^2\right)\left(\tfrac{1}{2\hat{D}^2} u_i^2 + Z_T'\right)} \\ + 10 B_{T,i} \left(\tfrac{1}{2\hat{D}^2} u_i^2 + Z_T'\right) + 4 B_{T,i}, \end{aligned}$$

where $Z_T' = 2 \ln\lceil 2 \log_2 T \rceil + \frac{1}{2}$.

5.8 Experiments

For an experimental evaluation we implemented six versions of MetaGrad, AdaGrad, Online Gradient Descent with learning rate $\eta_T = \frac{\hat{D}}{\sqrt{\sum_{s=1}^t \|\boldsymbol{g}_s\|_2^2 + 1e^{-8}}}$ (abbreviated as GDn), Online Gradient Descent with learning rate $\eta_t = \frac{\hat{D}}{G\sqrt{t}}$ (abbreviated as GDt) in python. The six versions of MetaGrad we used are the full version(abbreviated as MGFull), the coordinate version(abbreviated as MGCo), and two versions of MetaGrad that employ either Frequent Directions sketching with $m = 1$, $m = \min\{10, d\}$, $m = \min\{25, d\}$, and $m = \min\{50, d\}$ (abbreviated as MGF1, MGF10, MGF25, and MGF50 respectively). We compared the algorithms on seventeen datasets from the LIBSVM library (Chang and Lin, 2011), with T

ranging from 252 to 581012 and d ranging from 6 to 300. Of the seventeen datasets six had real outcomes and eleven had binary outcomes. A summary of the datasets can be found in Table 5.2 in Section 5.15. If available we used scaled versions of the datasets. For all datasets we used the features $\boldsymbol{x}_t$ without any adjustments in the following manner: we estimate $\hat{y} = \boldsymbol{w}_t^\intercal \boldsymbol{x}_t$ and feed this prediction to loss function $f(\hat{y}_t, y_t)$. For binary datasets we used logistic loss and hinge loss. For datasets with real outcomes we made use of squared loss and absolute loss. The settings of the algorithms can be found in Table 5.1 in Section 5.15. For AdaGrad and the coordinate version of MetaGrad we used $U = \{\boldsymbol{w} : \|\boldsymbol{w}\|_\infty \leq D\}$, where D was set to $3\|\boldsymbol{u}\|_\infty$, where $\boldsymbol{u} = \arg\min_{\boldsymbol{u}} \sum_{t=1}^T f(\boldsymbol{u}^\intercal \boldsymbol{x}_t, y_t)$. For the other algorithms we used $\mathcal{W} = \cap_{t=1}^T \mathcal{W}_t = \{\boldsymbol{w} : |\boldsymbol{x}_t^T \boldsymbol{w}| \leq C\}$ as domain, where $C = 3 \max_t |\boldsymbol{u}^\intercal \boldsymbol{x}_t|$. Hyperparamaters for algorithms involving a norm of the minimizer were set to three times the norm of the comparator $\boldsymbol{u}$. In other words $\hat{D} = D$. Hyperparameters involving an upper bound on a norm of $\boldsymbol{g}_t$ were set as follows. For $t = 1$, $G_1 = \|\boldsymbol{g}_1\|_2$ and then we update. For any subsequent round, if $G_{t-1} \leq \|\boldsymbol{g}_t\|_2$ set $G_t = \|\boldsymbol{g}_t\|_2$, otherwise $G_t = G_{t-1}$.

5.8.1 Experimental Results

In Figure 5.1 we plotted the regret of three versions of MetaGrad and AdaGrad versus the regret of GDt on a logarithmic scale. We decided to use GDt as a baseline algorithm since it is the algorithm with the lowest regret that is not MetaGrad (in nine experiments either AdaGrad or GDn had lower regret than GDt). Table 5.3 in Section 5.15 contains the regrets of all algorithms on all datasets.

Out of 34 experiments in nine experiments a version of MetaGrad did not have the lowest regret. Among the six version of MetaGrad MGFull appears to be the best version as it had the lowest regret for the most datasets (thirteen). As predicted by theory, increasing the sketching size mostly improved the performance of Frequent Directions. With $m = \min\{50, d\}$, the Frequent Directions version of MetaGrad is very close to the performance of the full version of MetaGrad. Overall, the coordinate version of MetaGrad is close to the performance of the Full version of MetaGrad, which suggests that on the datasets that we used the correlations between the features are of little importance.

At first sight it may seem surprising that Online Gradient Descent outperformed MetaGrad on a9a, bodyfat, housing, ijcnn, and mg when the loss had curvature. However, upon closer inspection of the regret bounds we see that even in theory the regret bound of GDt is no worse than the regret of MetaGrad. For example, on the dataset bodyfat ($d = 14$, $T = 252$) with the squared loss the full version

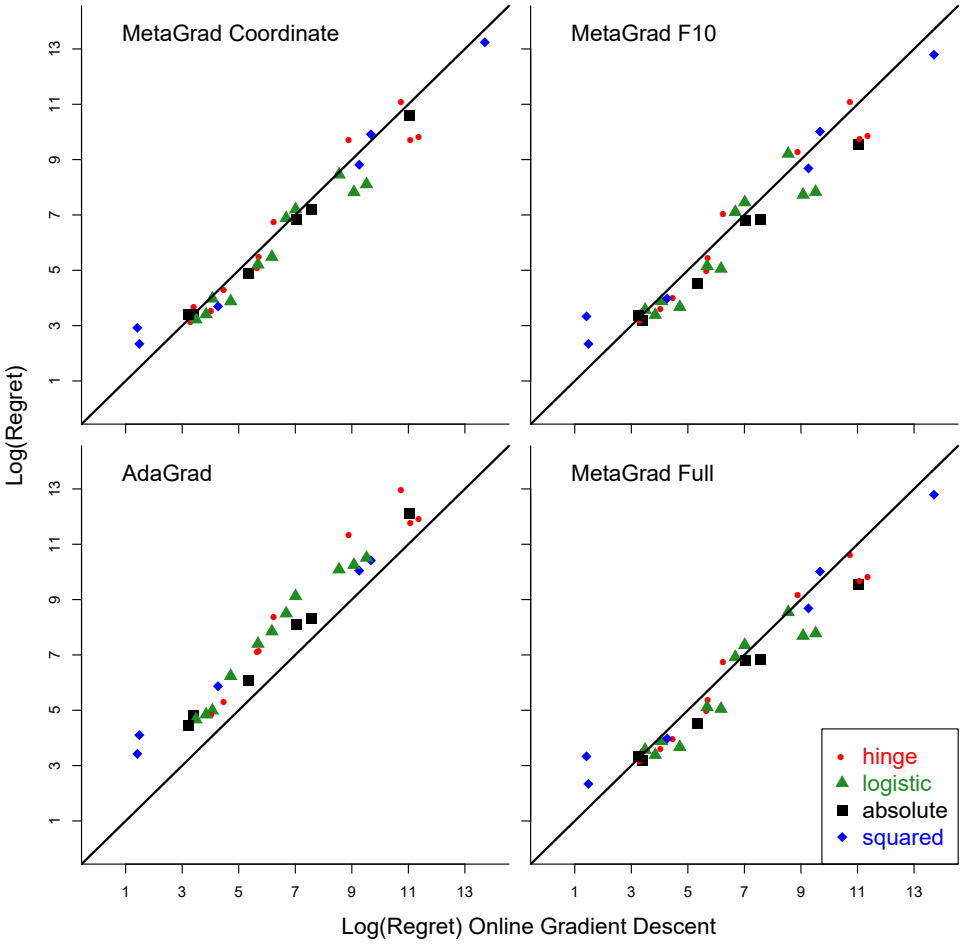


Figure 5.1: Comparison of the logarithm of the regret of three versions of MetaGrad and AdaGrad and the logarithm of the regret of GDt.

of MetaGrad has $O(\min\{d \log T, \sqrt{T}\}) = O(\sqrt{T})$ regret, whereas GDt also has $O(\sqrt{T})$ regret.

To our surprise, AdaGrad had the worst performance of all algorithms. However, upon closer review of the literature we see that in the experiments of Luo et al. (2017) and Chen et al. (2018) AdaGrad also had the worst performance, albeit in a different measure of performance (progressive misclassification rate and log objective gap, respectively).

Overall, we see that MetaGrad often outperforms AdaGrad and Online Gradient Descent with various learning rates.

5.9 Conclusion and Possible Extensions

We provide a new universally adaptive method, MetaGrad, which is robust to general convex losses but simultaneously can take advantage of special structure in the losses, like curvature or if the data come from a fixed distribution. The main new technique is to consider multiple learning rates in parallel: each learning rate η has its own surrogate loss (5.4.1) and there is a single controller method that aggregates the predictions of η-experts corresponding to the different surrogate losses.

An important feature of the controller is that its contribution to the final regret is the log of the number of experts, which is typically dominated by other terms in the bound. It is therefore cheap to add more experts for possibly different surrogate losses. To make the proof go through, a sufficient requirement on any such surrogates is that they replace the term $\big(\eta(\boldsymbol{u}-\boldsymbol{w}_t)^\intercal \boldsymbol{g}_t\big)^2$ in (5.4.1) by an upper bound. This possibility is exploited by Zhang et al. (2019), who add extra experts with surrogates that contain $\big(\eta\|\boldsymbol{g}_t\|_2\|\boldsymbol{u}-\boldsymbol{w}_t\|_2\big)^2$ instead. Since these surrogates are quadratic in all directions, and not just in the direction of $\boldsymbol{g}_t$, they are better suited for strongly convex losses, which then leads to an even more universally applicable extension of MetaGrad that also gets the optimal rate $O(\log T)$ for strongly convex losses.

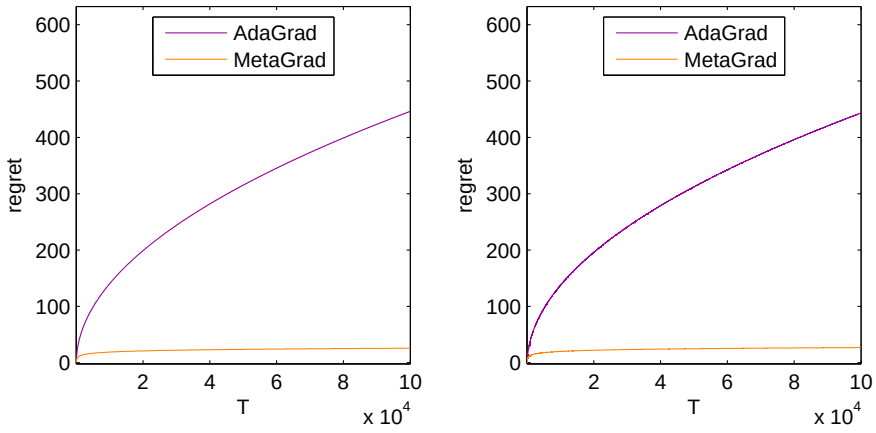
Another way to extend MetaGrad is to replace the exponential weights update in the controller by a different experts algorithm. Zhang et al. (2019) use this to extend MetaGrad for the case that the optimal parameters $\boldsymbol{u}$ vary over time, as measured in terms of the adaptive regret. See also Neuteboom (2020), who provides a similar extension of the closely related Squint algorithm for adaptive regret.

As a final possible extension, we mention the sliding window variant of Full Matrix AdaGrad (Agarwal et al., 2018). The same sliding window idea could be used to base the covariance matrix Σ_t^η in our Algorithm 11 only on the k most recent gradients. This has both computational advantages, because Σ_t^η then becomes a matrix of fixed rank $d+k$, and it could be beneficial for non-convex optimization when older covariance information needs to be discarded.

5.10 Extra Material Related to Section 5.3

In this section we gather extra material related to the fast rate examples from Section 5.3. We first provide simulations. Then we present the proofs of Theorems 20 and 21. And finally we give an example in which the unregularized hinge loss satisfies the Bernstein condition.

5.10.1 Simulations: Logarithmic Regret without Curvature



(a) Offline: $f_t(u) = |u - 1/4|$

(b) Stochastic Online: $f_t(u) = |u - x_t|$ where $x_t = \pm\frac{1}{2}$ i.i.d. with probabilities 0.4 and 0.6.

Figure 5.2: Examples of fast rates on functions without curvature. MetaGrad incurs logarithmic regret $O(\log T)$, while AdaGrad incurs $O(\sqrt{T})$ regret, matching its bound.

We provide two simple simulation examples to illustrate the sufficient conditions from Theorems 20 and 21, and to show that such fast rates are not automatically obtained by previous methods for general functions. Both our examples are one-dimensional, and have a stable optimum (that good algorithms will converge to); yet the functions are based on absolute values, which are neither strongly convex nor smooth, so the gradient norms do not vanish near the optimum. As our baseline we include AdaGrad (Duchi et al., 2011), because it is commonly used in practice (Mikolov et al., 2013; Schmidhuber, 2015) and because, in the one-dimensional case, it implements GD with an adaptive tuning of the learning rate that is applicable to general convex functions.

In the first example, we consider offline convex optimization of the fixed function $f_t(u) \equiv f(u) = |u - \frac{1}{4}|$, which satisfies (5.3.1), because it is convex. In the second example, we look at stochastic optimization with convex functions $f_t(u) = |u - x_t|$, where the outcomes $x_t = \pm\frac{1}{2}$ are chosen i.i.d. with probabilities 0.4 and 0.6. These probabilities satisfy (5.3.2) with $\beta = 1$. Their values are by no means essential, as long we avoid the worst case where the probabilities are equal.

Figure 5.2 graphs the results. We see that in both cases the regret of AdaGrad follows its $O(\sqrt{T})$ bound, while MetaGrad achieves an $O(\ln T)$ rate, as predicted by Theorems 20 and 21. This shows that MetaGrad achieves a type of adaptivity

that is not achieved by AdaGrad. We should be careful in extending this conclusion to higher dimensions, though: whereas (the diagonal version of) AdaGrad uses a separate learning rate per dimension, METAGRAD$^{\text{FULL}}$ shares learning rates between dimensions (unless we run a METAGRAD$^{\text{COOR}}$ rather than METAGRAD$^{\text{FULL}}$).

5.10.2 Proof of Theorem 20

Proof. By (5.3.1), applied with $\boldsymbol{w} = \boldsymbol{w}_t$, and Theorem 19, there exists a $C > 0$ (depending on a) such that, for all sufficiently large T,

$$\begin{aligned} R_T^{\boldsymbol{u}} &\leq a\tilde{R}_T^{\boldsymbol{u}} - bV_T^{\boldsymbol{u}} \leq C\sqrt{V_T^{\boldsymbol{u}}\, d\ln T} + Cd\ln T - bV_T^{\boldsymbol{u}} \\ &\leq \frac{\gamma}{2}CV_T^{\boldsymbol{u}} + \left(\frac{1}{2\gamma}+1\right)Cd\ln T - bV_T^{\boldsymbol{u}} \qquad \text{for all } \gamma > 0, \end{aligned}$$

where the last inequality is based on $\sqrt{xy} = \min_{\gamma>0} \frac{\gamma}{2}x + \frac{y}{2\gamma}$ for all $x, y > 0$. The result follows upon taking $\gamma = \frac{2b}{C}$. □

5.10.3 Proof of Theorem 21

Proof. Abbreviate $\tilde{r}_t^{\boldsymbol{u}} = (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t$. Then, by (5.1.1), Jensen's inequality and the Bernstein condition, there exists a constant $C > 0$ such that, for all sufficiently large T, the expected linearized regret is at most

$$\begin{aligned} \mathbb{E}\left[\tilde{R}_T^{\boldsymbol{u}^*}\right] &\leq C\,\mathbb{E}\left[\sqrt{V_T^{\boldsymbol{u}^*}\, d\ln T}\right] + Cd\ln T \leq C\sqrt{\mathbb{E}\left[V_T^{\boldsymbol{u}^*}\right]\, d\ln T} + Cd\ln T \\ &\leq C\sqrt{B\sum_{t=1}^{T}\left(\mathbb{E}\left[\tilde{r}_t^{\boldsymbol{u}^*}\right]\right)^\beta\, d\ln T} + Cd\ln T. \end{aligned}$$

We will repeatedly use the fact that

$$x^\alpha y^{1-\alpha} = c_\alpha \inf_{\gamma>0}\left(\frac{x}{\gamma} + \gamma^{\frac{\alpha}{1-\alpha}} y\right) \qquad \text{for any } x, y \geq 0 \text{ and } \alpha \in (0,1), \quad (5.10.1)$$

where $c_\alpha = (1-\alpha)^{1-\alpha}\alpha^\alpha$. Applying this first with $\alpha = 1/2$, $x = Bd\ln T$ and $y = \sum_{t=1}^T \left(\mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*}]\right)^\beta$, we obtain

$$\sqrt{B\sum_{t=1}^{T}\left(\mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*}]\right)^\beta\, d\ln T} \leq c_{1/2}\gamma_1 \sum_{t=1}^{T}\left(\mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*}]\right)^\beta + \frac{c_{1/2}}{\gamma_1}Bd\ln T \quad \text{for any } \gamma_1 > 0.$$

If $\beta = 1$, then $\sum_{t=1}^T \left(\mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*}]\right)^\beta = \mathbb{E}[\tilde{R}_T^{\boldsymbol{u}^*}]$ and the result follows by taking $\gamma_1 = \frac{1}{2Cc_{1/2}}$. Alternatively, if $\beta < 1$, then we apply (5.10.1) a second time, with $\alpha = \beta$,

$x = \mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*}]$ and $y = 1$, to find that, for any $\gamma_2 > 0$,

$$\begin{aligned}
&\sqrt{B \sum_{t=1}^{T} \left(\mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*}]\right)^{\beta} \, d \ln T} \\
&\leq c_\beta c_{1/2} \gamma_1 \sum_{t=1}^{T} \left(\frac{\mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*}]}{\gamma_2} + \gamma_2^{\beta/(1-\beta)} \right) + \frac{c_{1/2}}{\gamma_1} B d \ln T \\
&= \frac{c_\beta c_{1/2} \gamma_1}{\gamma_2} \, \mathbb{E}[\tilde{R}_T^{\boldsymbol{u}^*}] + c_\beta c_{1/2} \gamma_1 \gamma_2^{\beta/(1-\beta)} T + \frac{c_{1/2}}{\gamma_1} B d \ln T.
\end{aligned}$$

Taking $\gamma_1 = \frac{\gamma_2}{2 c_\beta c_{1/2} C}$, this yields

$$\mathbb{E}[\tilde{R}_T^{\boldsymbol{u}^*}] \leq \gamma_2^{1/(1-\beta)} T + \frac{4 C^2 c_{1/2}^2 c_\beta B d \ln T}{\gamma_2} + 2 C d \ln T.$$

We may optimize over γ_2 by a third application of (5.10.1), now with $x = 4 C^2 c_{1/2}^2 c_\beta B d \ln T$, $y = T$ and $\alpha = 1/(2-\beta)$, such that $\alpha/(1-\alpha) = 1/(1-\beta)$:

$$\begin{aligned}
\mathbb{E}[\tilde{R}_T^{\boldsymbol{u}^*}] &\leq \frac{1}{c_{1/(2-\beta)}} \left(4 C^2 c_{1/2}^2 c_\beta B d \ln T \right)^{1/(2-\beta)} T^{(1-\beta)/(2-\beta)} + 2 C d \ln T \\
&= O\left((B d \ln T)^{1/(2-\beta)} \, T^{(1-\beta)/(2-\beta)} + d \ln T \right),
\end{aligned}$$

which completes the proof. □

5.10.4 Unregularized Hinge Loss Example

As shown by Koolen et al. (2016), the Bernstein condition is satisfied in the following classification task:

Lemma 18 (Unregularized Hinge Loss Example)**.** *Suppose that $(\boldsymbol{X}_1, Y_1), (\boldsymbol{X}_2, Y_2), \ldots$ are i.i.d. with Y_t taking values in $\{-1, +1\}$, and let $f_t(\boldsymbol{u}) = \max\{0, 1 - Y_t \langle \boldsymbol{u}, \boldsymbol{X}_t \rangle\}$ be the* hinge loss*. Assume that both $\mathcal{W}$ and the domain for $\boldsymbol{X}_t$ are the d-dimensional unit ball. Then the (B, β)-Bernstein condition is satisfied with $\beta = 1$ and $B = \frac{2\lambda_{max}}{\|\boldsymbol{\mu}\|_2}$, where λ_{max} is the maximum eigenvalue of $\mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\intercal]$ and $\boldsymbol{\mu} = \mathbb{E}[Y\boldsymbol{X}]$, provided that $\|\boldsymbol{\mu}\|_2 > 0$.*

In particular, if $\boldsymbol{X}_t$ is uniformly distributed on the sphere and $Y_t = \operatorname{sign}(\langle \bar{\boldsymbol{u}}, \boldsymbol{X}_t \rangle)$ is the noiseless classification of $\boldsymbol{X}_t$ according to the hyperplane with normal vector $\bar{\boldsymbol{u}}$, then $B \leq \frac{c}{\sqrt{d}}$ for some absolute constant $c > 0$.

Thus the version of the Bernstein condition that implies an $O(d \log T)$ rate is always satisfied for the hinge loss on the unit ball, except when $\|\boldsymbol{\mu}\|_2 = 0$, which is very

natural to exclude, because it implies that the expected hinge loss is 1 (its maximal value) for all $\boldsymbol{u}$, so there is nothing to learn. It is common to add ℓ_2-regularization to the hinge loss to make it strongly convex, but this example shows that that is not necessary to get logarithmic regret.

5.11 Controller Regret Bound (Proof of Lemma 15)

We prove Lemma 15 in two parts.

5.11.1 Decomposing the Surrogate Regret

Fix a comparator point $\boldsymbol{u} \in \bigcap_{t=1}^{T} \mathcal{W}_t$. We will first bound the surrogate regret

$$R_T^\eta(\boldsymbol{u}) \;\coloneqq\; \sum_{t=1}^{T} (\ell_t^\eta(\boldsymbol{w}_t) - \ell_t^\eta(\boldsymbol{u}))$$

for any $\eta \in \mathcal{G}$ not expired after T rounds (see Definition 2). Note that by definition (5.4.1), the surrogate loss $\ell_t^\eta(\boldsymbol{w}_t)$ of the controller is always zero, but we believe writing it helps interpretation. We will then use this surrogate regret bound to control the (non-surrogate) regret.

For the first half of this section, we fix a final time T, and a grid-point $\eta \in \mathcal{G}$ that is still not expired after time T, (see Definition 2)

Definition 3. *We define the wakeup time of learning rate $\eta \in \mathcal{G}$ by*

$$a^\eta \;\coloneqq\; \inf\left\{ t \le T \,\middle|\, \eta > \frac{1}{4\left(\sum_{s=1}^{t-1} b_s \frac{B_{s-1}}{B_s} + B_{t-1}\right)} \right\} \wedge (T+1).$$

Note that we manually set to $T+1$ the wakeup time of an η that does not wake up during the first T rounds. We do this so that $[1, a^\eta - 1]$ and $[a^\eta, T]$ always partition rounds $[1, T]$.

Our strategy will be to split the regret in three parts, which we will analyse separately.

Proposition 1. *We have*

$$R_T^\eta(\boldsymbol{u}) \;=\; \underbrace{\sum_{t=1}^{a^\eta-1} \left(\ell_t^\eta(\boldsymbol{w}_t) - \ell_t^\eta(\boldsymbol{u})\right)}_{\ell^\eta\text{-regret of controller w.r.t. } \boldsymbol{u}} + \underbrace{\sum_{t=a^\eta}^{T} \left(\ell_t^\eta(\boldsymbol{w}_t) - \ell_t^\eta(\boldsymbol{w}_t^\eta)\right)}_{\ell^\eta\text{-regret of controller w.r.t. } \eta\text{-expert}}$$
$$+ \underbrace{\sum_{t=a^\eta}^{T} \left(\ell_t^\eta(\boldsymbol{w}_t^\eta) - \ell_t^\eta(\boldsymbol{u})\right)}_{\ell^\eta\text{-regret of } \eta\text{-expert w.r.t. } \boldsymbol{u}}$$

Proof. The choice of a^η makes all $\boldsymbol{w}_t^\eta$ defined. We can hence merge the sums. □

We think of the three sums as follows. The first sum is "startup nuisance", and it will turn out to be small. The second sum is controlled by the controller, and it only depends on its construction. The third sum is controlled by the η-experts, and it only depends on their construction.

We will now proceed to bound the three parts above. First, we reduce to the clipped surrogate losses (5.4.4) at almost negligible cumulative cost using the clipping technique of Cutkosky (2019).

Lemma 19 (Clipping in the controller is cheap)**.**

$$\underbrace{\sum_{t=1}^{a^\eta-1} \left(\ell_t^\eta(\boldsymbol{w}_t) - \ell_t^\eta(\boldsymbol{u})\right)}_{\ell^\eta\text{-regret of controller w.r.t. } \boldsymbol{u}} + \underbrace{\sum_{t=a^\eta}^{T} \left(\ell_t^\eta(\boldsymbol{w}_t) - \ell_t^\eta(\boldsymbol{w}_t^\eta)\right)}_{\ell^\eta\text{-regret of controller w.r.t. } \eta\text{-expert}}$$
$$\leq \underbrace{\sum_{t=1}^{a^\eta-1} \left(\bar{\ell}_t^\eta(\boldsymbol{w}_t) - \bar{\ell}_t^\eta(\boldsymbol{u})\right)}_{\bar{\ell}^\eta\text{-regret of controller w.r.t. } \boldsymbol{u}} + \underbrace{\sum_{t=a^\eta}^{T} \left(\bar{\ell}_t^\eta(\boldsymbol{w}_t) - \bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)\right)}_{\bar{\ell}^\eta\text{-regret of controller w.r.t. } \eta\text{-expert}} + 2\eta B_T$$

Proof. For any $\boldsymbol{u} \in \mathcal{W}_t$ (which includes the case $\boldsymbol{u} = \boldsymbol{w}_t^\eta$), we may use the definition of the range bound (5.2.1), the surrogate loss (5.4.1) and its clipped version (5.4.4) to find

$$\begin{aligned}
&\left(\ell_t^\eta(\boldsymbol{w}_t) - \ell_t^\eta(\boldsymbol{u})\right) - \left(\bar{\ell}_t^\eta(\boldsymbol{w}_t) - \bar{\ell}_t^\eta(\boldsymbol{u})\right) \\
&= \eta \frac{B_t - B_{t-1}}{B_t} (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t - \underbrace{\eta^2 \frac{B_t^2 - B_{t-1}^2}{B_t^2} \left((\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t\right)^2}_{\geq 0} \\
&\leq 2\eta \frac{B_t - B_{t-1}}{B_t} b_t \;\leq\; 2\eta \left(B_t - B_{t-1}\right).
\end{aligned}$$

Summing over rounds completes the proof. □

Next we deal with the clipped surrogate regret. We first handle the case of the early rounds before a^η. The key idea is that when η has not yet woken up, it is very small. Since the surrogate loss scales with η, it is small as well, even in sum.

Lemma 20. *For any η and any $\boldsymbol{u} \in \bigcap_{s=1}^{a^\eta-1} \mathcal{W}_s$*

$$\underbrace{\sum_{t=1}^{a^\eta-1} \left(\bar{\ell}_t^\eta(\boldsymbol{w}_t) - \bar{\ell}_t^\eta(\boldsymbol{u})\right)}_{\bar{\ell}^\eta\text{-regret of controller w.r.t. } \boldsymbol{u}} \;\leq\; \frac{1}{2}.$$

Proof. By definition of the clipped surrogate loss $\bar{\ell}_t^\eta$ in (5.4.4), the range bound b_t in (5.2.1) and the wakeup time a_t in Definition 3,

$$\begin{aligned}\sum_{t=1}^{a^\eta-1} \bar{\ell}_t^\eta(\boldsymbol{w}_t) - \bar{\ell}_t^\eta(\boldsymbol{u}) \;&\leq\; \sum_{t=1}^{a^\eta-1} \eta(\boldsymbol{w}_t - \boldsymbol{u})^\intercal \bar{\boldsymbol{g}}_t \\ &\leq \sum_{t:\eta \leq \frac{1}{4\left(\sum_{s=1}^{t-1} b_s \frac{B_{s-1}}{B_s} + B_{t-1}\right)}} \eta 2 b_t \frac{B_{t-1}}{B_t} \;\leq\; \frac{1}{2}.\end{aligned}$$

□

In the next subsection we deal with the middle sum in Proposition 1. This part only depends on the construction of the controller. We deal with the final sum in the section after that.

5.11.2 Controller surrogate regret bound

The controller is a specialists algorithm, which sometimes resets. We call the time segments between resets epochs. In every epoch, the controller guarantees a certain specialists regret bound w.r.t. any η-expert in its grid.

The η-expert that we need can be active during several epochs. Our strategy, following Mhammedi et al. (2019), will be the following. We incur the controller regret in the last and one-before-last epochs. We further separately prove, using the reset condition, that the total regret in all earlier epochs is small.

Theorem 27. *Consider an epoch starting at time $\tau + 1$ and fix any later time t in that same epoch. Fix any grid point $\eta \in \mathcal{G}$ not expired after t rounds (meaning*

$\eta \leq \frac{1}{2B_{t-1}}$). *Then the MetaGrad controller guarantees*

$$\underbrace{\sum_{s\in(\tau,t]:s\geq a^\eta} \left(\bar{\ell}_t^\eta(\boldsymbol{w}_t) - \bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)\right)}_{\text{specialist } \bar{\ell}^\eta\text{-regret of controller w.r.t. } \eta\text{-expert on } (\tau,t]} \leq \ln\left\lceil 2\log_2\left(\sum_{s=1}^{t-1}\frac{b_s}{B_s}+1\right)\right\rceil_+ .$$

Note that it is not important what the η-experts do at this point, the only feature that we use in the proof is that $\boldsymbol{w}_t^\eta \in \mathcal{W}_t$ for each active η. Also, note that the right-hand side is $O(\ln\ln T)$. We choose to stay with the current more detailed expression as it can be much smaller. This occurs whenever the actually observed loss ranges b_s are smaller than their respective upper bounds B_s.

Proof. We first observe that Algorithm 10, as far as it maintains the weights $p_t(\eta)$ between resets, implements Specialists Exponential Weights (called SBayes by Freund et al., 1997). In our particular case it is applied to specialists $\eta \in \mathcal{G}$, loss function $\eta \mapsto \ell_t^\eta(\boldsymbol{w}_t^\eta)$, active set $\mathcal{A}_t \subseteq \mathcal{G}$ and uniform (improper) prior on $\mathcal{G}$. The specialists regret bound (Freund et al., 1997, Theorem 1) directly yields[3]

$$\sum_{s\in(\tau,t]:s\geq a^\eta} -\ln \mathop{\mathbb{E}}_{p_t(\eta)}\left[e^{-\bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)}\right] \leq \ln\left|\bigcup_{s\in(\tau,t]}\mathcal{A}_s\right| + \sum_{s\in(\tau,t]:s\geq a^\eta}\bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta).$$

Algorithm 10 further chooses the controller iterate

$$\boldsymbol{w}_t = \frac{\mathbb{E}_{p_t(\eta)}\left[\eta\boldsymbol{w}_t^\eta\right]}{\mathbb{E}_{p_t(\eta)}\left[\eta\right]}$$

which we claim ensures that

$$0 \leq -\ln \mathop{\mathbb{E}}_{p_t(\eta)}\left[e^{-\bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)}\right].$$

To see why, we use the definition (5.4.4) of clipped loss and gradient to obtain $(\boldsymbol{w}_t - \boldsymbol{w}_t^\eta)^\intercal \bar{\boldsymbol{g}}_t \geq -2B_{t-1}$, and we further use that p_t is supported on $\mathcal{A}_t$, which implies that $\eta \leq \frac{1}{4B_{t-1}}$. Together these license[4] the "prod bound" ($e^{x-x^2} \leq 1+x$

[3] Our improper prior does not cause any trouble here, because renormalizing the prior, in hindsight, to the finite set of η-experts that were ever active preserves the algorithm's output and hence its regret bound.

[4] Here we motivate our controller algorithm using the loss function $\eta \mapsto \bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)$. One can alternatively base it on the loss function $\eta \mapsto -\ln\left(1+\eta(\boldsymbol{w}_t - \boldsymbol{w}_t^\eta)^\intercal\bar{\boldsymbol{g}}_t\right)$ (These two versions are called Squint and iProd respectively by Koolen and Van Erven, 2015). As the second is always smaller (by the prod bound), using it would give a strictly tighter theorem here. We do not see a way to ultimately harvest this gain, as we would still need to invoke the prod bound at a later point in the analysis to express our regret bound in second-order form. We chose to present the "Squint-style" version here as we believe it is the more intuitive of the two.

for $x \geq -\frac{1}{2}$) yielding

$$-\ln \mathop{\mathbb{E}}_{p_t(\eta)} \left[e^{-\bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)} \right] \;\geq\; -\ln \mathop{\mathbb{E}}_{p_t(\eta)} \left[1 + \eta (\boldsymbol{w}_t - \boldsymbol{w}_t^\eta)^\intercal \bar{\boldsymbol{g}}_t \right] \;=\; 0.$$

Inserting $\ell_t^\eta(\boldsymbol{w}_t) = 0$, this implies

$$\sum_{s \in (\tau,t] : s \geq a^\eta} \left(\bar{\ell}_t^\eta(\boldsymbol{w}_t) - \bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta) \right) \;\leq\; \ln \left| \bigcup_{s \in (\tau,t]} \mathcal{A}_s \right|.$$

It remains to bound the maximum number of awake grid-points during any epoch. Recall that the active set at any time t is

$$\mathcal{A}_t \;=\; \left(\frac{1}{4 \left(\sum_{s=1}^{t-1} b_s \frac{B_{s-1}}{B_s} + B_{t-1} \right)}, \frac{1}{4 B_{t-1}} \right]$$

Both endpoints are decreasing with t. Since our epoch starts at time $\tau + 1$, the maximal η awake in the epoch is

$$\max \left\{ \eta \in \mathcal{G} \;\middle|\; \eta \leq \frac{1}{4 B_\tau} \right\}.$$

As the epoch lasts until at least time $t \geq \tau + 1$, the smallest η active in the epoch is

$$\min \left\{ \eta \in \mathcal{G} \;\middle|\; \eta \geq \frac{1}{4 \left(\sum_{s=1}^{t-1} b_s \frac{B_{s-1}}{B_s} + B_{t-1} \right)} \right\}.$$

And since $\mathcal{G}$ is exponentially spaced with base 2, the maximum number of η that could possibly have been awake is

$$\begin{aligned}
\left\lceil \log_2 \frac{\left(\sum_{s=1}^{t-1} b_s \frac{B_{s-1}}{B_s} + B_{t-1} \right)}{B_\tau} \right\rceil \;&\leq\; \left\lceil \log_2 \frac{B_{t-1} \left(\sum_{s=1}^{t-1} \frac{b_s}{B_s} + 1 \right)}{B_\tau} \right\rceil \\
&\overset{(5.4.3)}{\leq} \left\lceil \log_2 \left(\left(\sum_{s=1}^{t-1} \frac{b_s}{B_s} \right) \left(\sum_{s=1}^{t-1} \frac{b_s}{B_s} + 1 \right) \right) \right\rceil \\
&\leq \left\lceil 2 \log_2 \left(\sum_{s=1}^{t-1} \frac{b_s}{B_s} + 1 \right) \right\rceil_+,
\end{aligned}$$

so our prior costs for the improper (uniform on $\mathcal{G}$) prior are upper bounded by

$$\ln \left| \bigcup_{s \in (\tau,t]} \mathcal{A}_s \right| \;\leq\; \ln \left\lceil 2 \log_2 \left(\sum_{s=1}^{t-1} \frac{b_s}{B_s} + 1 \right) \right\rceil_+. \tag{5.11.1}$$

□

We now have a specialists regret bound that we can apply to each epoch.

Lemma 21 (Total regret in far past is small)**.** *Consider two consecutive epochs, starting after $\tau_1 < \tau_2$, and let η be not expired after τ_1 rounds. Then*

$$\sum_{s\in[1,\tau_1],s\geq a^\eta} \left(\bar{\ell}_s^\eta(\boldsymbol{w}_s) - \bar{\ell}_s^\eta(\boldsymbol{w}_s^\eta)\right) \;\leq\; 2\eta B_{\tau_2}$$

Proof.

$$\begin{aligned} -\sum_{s\in[1,\tau_1],s\geq a^\eta} \bar{\ell}_s^\eta(\boldsymbol{w}_s^\eta) \;\leq\; 2\eta \sum_{s=1}^{\tau_1} b_s \frac{B_{s-1}}{B_s} &\leq 2\eta B_{\tau_1} \sum_{s=1}^{\tau_1} \frac{b_s}{B_s} \\ &\leq 2\eta B_{\tau_1} \sum_{s=1}^{\tau_2} \frac{b_s}{B_s} \\ &\leq 2\eta B_{\tau_2}, \end{aligned}$$

where the last inequality is the reset condition (5.4.3) at time τ_2. □

We are now ready to compose the main theorem.

Theorem 28 (Overall controller specialists regret bound)**.** *Let η be not expired after T rounds. Then*

$$\sum_{t=a^\eta}^{T} \left(\bar{\ell}_t^\eta(\boldsymbol{w}_t) - \bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)\right) \;\leq\; 2\eta B_T + 2\ln\left\lceil 2\log_2\left(\sum_{t=1}^{T-1} \frac{b_t}{B_t} + 1\right)\right\rceil. \quad (5.11.2)$$

Proof. We make a case distinction based on the number of epochs started by the algorithm. First, let us check the general case of ≥ 3 epochs (at least two normal epochs after the startup epoch). We apply the controller regret bound, Theorem 27, to the last two epochs each. Suppose these start after τ_1 and τ_2. For any $\eta \in \mathcal{G}$ not expired, we find

$$\begin{aligned} &-\sum_{t\in(\tau_1,\tau_2],t\geq a^\eta} \bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta) - \sum_{t\in(\tau_2,T],t\geq a^\eta} \bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta) \\ &\leq\; \ln\left\lceil 2\log_2\left(\sum_{s=1}^{\tau_2-1} \frac{b_s}{B_s} + 1\right)\right\rceil + \ln\left\lceil 2\log_2\left(\sum_{t=1}^{T-1} \frac{b_t}{B_t} + 1\right)\right\rceil. \end{aligned}$$

The regret on all epochs except the last two is bounded by Lemma 21. So together we obtain the theorem. Alternatively, suppose there are 2 epochs. Then, since we

get no clipped regret in the 1st epoch, we apply the controller regret bound only in the second epoch to get

$$-\sum_{t\in[1,T],t\geq a^\eta} \bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta) \leq \ln\left\lceil 2\log_2\left(\sum_{t=1}^{T-1}\frac{b_t}{B_t}+1\right)\right\rceil,$$

and (5.11.2) also holds. Finally, if there is only 1 epoch, then our clipped regret is 0 because nobody is awake, so (5.11.2) also holds. □

5.12 Proof of Lemma 16

Proof. The η-expert algorithm implements the exponentially weighted average forecaster with ℓ_t^η as the quadratic loss, unit learning rate, and with greedy projections (of the mean) onto $\mathcal{W}_t$. By (Hazan et al., 2007, Proof of Theorem 2), we obtain that

$$\sum_{t=a^\eta}^{T}\left(\ell_t^\eta(\boldsymbol{w}_t^\eta)-\ell_t^\eta(\boldsymbol{u})\right) \;\leq\; \frac{\|\boldsymbol{u}\|_2^2}{2\hat{D}^2}+\frac{1}{2}\sum_{t=a^\eta}^{T}\boldsymbol{g}_t'^{\intercal}\Sigma_{t+1}^\eta\boldsymbol{g}_t'$$

where $\boldsymbol{g}_t' = \eta\left(1+2\eta\left\langle\boldsymbol{w}_t-\boldsymbol{w}_t^\eta,\boldsymbol{g}_t\right\rangle\right)\boldsymbol{g}_t$ and where we recall that $(\Sigma_{t+1}^\eta)^{-1} = \frac{1}{\hat{D}^2}\boldsymbol{I}+2\eta^2\sum_{s=a^\eta}^{t}\boldsymbol{g}_s\boldsymbol{g}_s^{\intercal}$. Expanding, we obtain

$$\boldsymbol{g}_t'^{\intercal}\Sigma_{t+1}^\eta\boldsymbol{g}_t' \;=\; \frac{1}{2}\left(1+2\eta\left\langle\boldsymbol{w}_t-\boldsymbol{w}_t^\eta,\boldsymbol{g}_t\right\rangle\right)^2\cdot 2\eta^2\boldsymbol{g}_t^{\intercal}\left(\frac{1}{\hat{D}^2}\boldsymbol{I}+2\eta^2\sum_{s=a^\eta}^{t}\boldsymbol{g}_s\boldsymbol{g}_s^{\intercal}\right)^{-1}\boldsymbol{g}_t$$

Now we may use that

$$\frac{1}{2}\left(1+2\eta\left\langle\boldsymbol{w}_t-\boldsymbol{w}_t^\eta,\boldsymbol{g}_t\right\rangle\right)^2 \leq \frac{1}{2}(1+4\eta b_t)^2 \leq \frac{1}{2}(1+1)^2 = 2 \qquad (5.12.1)$$

by the assumed upper bound on η. Moreover, by concavity of the log determinant, we have

$$\begin{aligned}
&2\eta^2\boldsymbol{g}_t^{\intercal}\left(\frac{1}{\hat{D}^2}\boldsymbol{I}+2\eta^2\sum_{s=a^\eta}^{t}\boldsymbol{g}_s\boldsymbol{g}_s^{\intercal}\right)^{-1}\boldsymbol{g}_t\\
&\leq\; \ln\det\left(\frac{1}{\hat{D}^2}\boldsymbol{I}+2\eta^2\sum_{s=a^\eta}^{t}\boldsymbol{g}_s\boldsymbol{g}_s^{\intercal}\right)-\ln\det\left(\frac{1}{\hat{D}^2}\boldsymbol{I}+2\eta^2\sum_{s=a^\eta}^{t-1}\boldsymbol{g}_s\boldsymbol{g}_s^{\intercal}\right).
\end{aligned}$$

Summing over rounds and telescoping, we find

$$\frac{1}{2}\sum_{t=a^\eta}^{T}\boldsymbol{g}_t'^{\intercal}\Sigma_{t+1}^\eta\boldsymbol{g}_t' \;\leq\; \ln\det\Big(\boldsymbol{I}+2\eta^2\hat{D}^2\sum_{t=a^\eta}^{T}\boldsymbol{g}_t\boldsymbol{g}_t^{\intercal}\Big)$$

and obtain the result. □

5.13 Composition Proofs of Theorems 23 and 25

We combine the proofs of Theorems 23 and 25, which are both special cases of the following more abstract result:

Theorem 29. *Suppose there exist a number $\omega \geq 0$ and a positive semi-definite matrix $\boldsymbol{F}'$ such that the linearized regret is at most*

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \eta\omega + \frac{\ln\det\left(\boldsymbol{I} + 2\eta^2\hat{D}^2\boldsymbol{F}'\right) + \frac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + 2\ln\left\lceil 2\log_2 Q_T\right\rceil_+ + \frac{1}{2}}{\eta} + 4B_T.$$

Then the linearized regret is both bounded by

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \frac{5}{2}\sqrt{\omega\left(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + Z_T\right)} + 10B_T\left(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + Z_T\right) + 4B_T,$$

where $Z_T = \operatorname{rk}(\boldsymbol{F}')\ln\left(1 + \frac{\hat{D}^2\sum_{t=1}^T\|\boldsymbol{g}_t\|_2^2}{8B_T^2\operatorname{rk}(\boldsymbol{F}')}\right) + 2\ln\left\lceil 2\log_2 T\right\rceil_+ + \frac{1}{2}$, and by

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \frac{5}{2}\sqrt{\left(\omega + 2\hat{D}^2\operatorname{tr}(\boldsymbol{F}')\right)\left(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + Z'_T\right)} + 10B_T\left(\tfrac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + Z'_T\right) + 4B_T,$$

where $Z'_T = 2\ln\left\lceil 2\log_2 T\right\rceil_+ + \frac{1}{2}$.

Theorem 23 corresponds to the case $\omega = V_T^{\boldsymbol{u}}$ and $\boldsymbol{F}' = \boldsymbol{F}_T$, such that $\operatorname{Tr}(\boldsymbol{F}') = \sum_{t=1}^T\|\boldsymbol{g}_t\|_2^2$; Theorem 25 is obtained with $\omega = V_T^{\boldsymbol{u}} + \frac{2\hat{D}^2 m\Omega_q}{m-q}$ and $\boldsymbol{F}' = \boldsymbol{S}_T^\intercal\boldsymbol{S}_T$. The precondition of Theorem 29 is established by Theorems 22 and 24, respectively.

To prove Theorem 29 we start with a general lemma about optimizing in η:

Lemma 22. *For any $X, Y > 0$,*

$$\min_{\eta\in\mathcal{G}\,:\,\eta\leq\frac{1}{4B_T}} \eta X + \frac{Y}{\eta} \;\leq\; \frac{5}{2}\sqrt{XY} + 10B_T Y.$$

Proof. Let us denote the unconstrained optimizer of the left-hand side by $\hat{\eta} = \sqrt{Y/X}$. We distinguish two cases: first, when $\hat{\eta} \leq \frac{1}{4B_T}$, we upper bound the left-hand side by choosing the closest grid point $\eta \in \mathcal{G}$ below $\hat{\eta}$ (which, in the worst case, is at $\hat{\eta}/2$) to obtain

$$\min_{\eta\in\mathcal{G}\,:\,\eta\leq\frac{1}{4B_T}} \eta X + \frac{Y}{\eta} \;\leq\; \max_{\eta\in[\hat{\eta}/2,\hat{\eta}]} \eta X + \frac{Y}{\eta} = \frac{5}{2}\sqrt{XY}.$$

In the second case, if $\hat{\eta} > \frac{1}{4B_T}$, we plug in the highest available grid point (for which the worst case is $\frac{1}{8B_T}$) to find

$$\min_{\eta\in\mathcal{G}\,:\,\eta\leq\frac{1}{4B_T}} \eta X + \frac{Y}{\eta} \;\leq\; \frac{1}{8B_T}X + 8B_T Y \;<\; 10B_T Y,$$

where the second inequality follows by the assumption that $\hat{\eta} > \frac{1}{4B_T}$. In both cases the conclusion of the lemma follows. □

Proof. **(Theorem 29)** We start with the first claim of the theorem. By assumption, for any $\eta \leq \frac{1}{4B_T}$ in the grid $\mathcal{G}$, we have

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \eta\omega + \frac{A}{\eta} + 4B_T$$
$$\text{where} \qquad A \;=\; \ln\det\left(\boldsymbol{I} + \frac{1}{8B_T^2}\hat{D}^2\boldsymbol{F'}\right) + \frac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + 2\ln\lceil 2\log_2 T\rceil + \frac{1}{2}.$$

Lemma 22 therefore implies that

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \frac{5}{2}\sqrt{\omega A} + 10B_T A + 4B_T.$$

The proof of the first claim follows by applying the inequality $\log\det(\boldsymbol{I}+\boldsymbol{M}) \leq \mathrm{rk}(\boldsymbol{M})\log\left(1+\frac{\mathrm{Tr}(\boldsymbol{M})}{\mathrm{rk}(\boldsymbol{M})}\right)$ (see Lemma 23 below) to the matrix $\boldsymbol{M} = \frac{1}{8B_T^2}\hat{D}^2\boldsymbol{F'}$.

For the second claim of the theorem, we again start from Theorem 22 and now bound $\ln\det(\boldsymbol{I}+\boldsymbol{M}) \leq \mathrm{Tr}(\boldsymbol{M})$ for $\boldsymbol{M} = 2\eta^2\hat{D}^2\boldsymbol{F'}$ (again see Lemma 23) to obtain

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \eta\omega + 2\eta\hat{D}^2\,\mathrm{tr}(\boldsymbol{F'}) + \frac{A'}{\eta} + 4B_T$$

where

$$A' \;=\; \frac{1}{2\hat{D}^2}\|\boldsymbol{u}\|_2^2 + 2\ln\lceil 2\log_2 T\rceil + \frac{1}{2}.$$

Using Lemma 22, the second claim follows, which completes the proof of the theorem. □

Lemma 23. *For any positive semi-definite matrix $\boldsymbol{M} \in \mathbb{R}^d$*

$$\log\det(\boldsymbol{I}+\boldsymbol{M}) \leq \mathrm{rk}(\boldsymbol{M})\log\left(1+\frac{\mathrm{Tr}(\boldsymbol{M})}{\mathrm{rk}(\boldsymbol{M})}\right)$$

and

$$\log\det(\boldsymbol{I}+\boldsymbol{M}) \leq \mathrm{Tr}(\boldsymbol{M}).$$

Proof. Let $\lambda_1, \dots, \lambda_d$ be the eigenvalues of $\boldsymbol{M}$. Then $(1+\lambda_1), \dots, (1+\lambda_d)$ are the eigenvalues of $\boldsymbol{I}+\boldsymbol{M}$, and Jensen's inequality implies

$$\begin{aligned}\log\det(\boldsymbol{I}+\boldsymbol{M}) &= \sum_{i=1}^{d}\log(1+\lambda_i) = \operatorname{rk}(\boldsymbol{M})\sum_{i:\lambda_i\neq 0}\frac{1}{\operatorname{rk}(\boldsymbol{M})}\log(1+\lambda_i)\\ &\leq \operatorname{rk}(\boldsymbol{M})\log\left(1+\sum_{i:\lambda_i\neq 0}\frac{\lambda_i}{\operatorname{rk}(\boldsymbol{M})}\right) = \operatorname{rk}(\boldsymbol{M})\log\left(1+\frac{\operatorname{Tr}(\boldsymbol{M})}{\operatorname{rk}(\boldsymbol{M})}\right),\end{aligned}$$

which proves the first inequality. The second inequality follows because

$$\log\det(\boldsymbol{I}+\boldsymbol{M}) = \sum_{i=1}^{d}\log(1+\lambda_i) \leq \sum_{i=1}^{d}\lambda_i = \operatorname{Tr}(\boldsymbol{M}).$$

□

5.14 Bernstein for Linearized Excess Loss

Let $f : \mathcal{W} \to \mathbb{R}$ be a convex function drawn from distribution $\mathbb{P}$ with stochastic optimum $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u}\in\mathcal{W}} \mathbb{E}_{f\sim\mathbb{P}}[f(\boldsymbol{u})]$. For any $\boldsymbol{w} \in \mathcal{W}$, we now show that the Bernstein condition for the excess loss $X := f(\boldsymbol{w}) - f(\boldsymbol{u}^*)$ implies the Bernstein condition with the same exponent β for the linearized excess loss $Y := (\boldsymbol{w} - \boldsymbol{u}^*)^\intercal \nabla f(\boldsymbol{w})$. These variables satisfy $Y \geq X$ by convexity of f and $Y \leq C := D'G'$.

Lemma 24. *For $\beta \in (0,1]$, let X be a (B,β)-Bernstein random variable:*

$$\mathbb{E}[X^2] \leq B\,\mathbb{E}[X]^\beta.$$

Then any bounded random variable $Y \leq C$ with $Y \geq X$ pointwise satisfies the (B',β)-Bernstein condition

$$\mathbb{E}[Y^2] \leq B'\,\mathbb{E}[Y]^\beta$$

for $B' = \max\left\{B, \frac{2}{\beta}C^{2-\beta}\right\}$.

Proof. For $\beta \in (0,1)$ we will use the fact that

$$z^\beta \;=\; c_\beta \inf_{\gamma>0}\left(\frac{z}{\gamma} + \gamma^{\frac{\beta}{1-\beta}}\right) \qquad \text{for any } z \geq 0,$$

with $c_\beta = (1-\beta)^{1-\beta}\beta^\beta$. For $\gamma = \left(\frac{1-\beta}{\beta}\,\mathbb{E}[Y]\right)^{1-\beta}$ we therefore have

$$\begin{aligned}
\mathbb{E}[X^2] - B'\,\mathbb{E}[X]^\beta &\geq \mathbb{E}[X^2] - B' c_\beta \left(\frac{\mathbb{E}[X]}{\gamma} + \gamma^{\frac{\beta}{1-\beta}}\right) \\
&\geq \mathbb{E}[Y^2] - B' c_\beta \left(\frac{\mathbb{E}[Y]}{\gamma} + \gamma^{\frac{\beta}{1-\beta}}\right) && (5.14.1) \\
&= \mathbb{E}[Y^2] - B'\,\mathbb{E}[Y]^\beta, && (5.14.2)
\end{aligned}$$

where the second inequality holds because $x^2 - c_\beta B' x/\gamma$ is a decreasing function of $x \leq C$ for $\gamma \leq \frac{c_\beta B'}{2C}$, which is satisfied by the choice of B'. This proves the lemma for $\beta \in (0,1)$. The claim for $\beta = 1$ follows by taking the limit $\beta \to 1$ in (5.14.2). □

5.15 Details of Experiments

Algorithm	$\hat{D}$	Domain	Domain Parameter	G
AdaGrad	$3\|\boldsymbol{u}\|_\infty$	$\mathcal{W}_t = \{\boldsymbol{w} : \|\boldsymbol{w}\|_\infty \leq D\}$	$D = 3\|\boldsymbol{u}\|_\infty$	$\cdot$
GDn	$3\|\boldsymbol{u}\|_2$	$\mathcal{W}_t = \{\boldsymbol{w} : \|\langle \boldsymbol{w}, \boldsymbol{x}_t\rangle\| \leq C\}$	$C = 3\max_t \|\langle \boldsymbol{u}, \boldsymbol{x}_t\rangle\|$	$\cdot$
GDt	$3\|\boldsymbol{u}\|_2$	$\mathcal{W}_t = \{\boldsymbol{w} : \|\langle \boldsymbol{w}, \boldsymbol{x}_t\rangle\| \leq C\}$	$C = 3\max_t \|\langle \boldsymbol{u}, \boldsymbol{x}_t\rangle\|$	$\max_{s\leq t} \|\boldsymbol{g}_s\|_2$
MGFull	$3\|\boldsymbol{u}\|_2$	$\mathcal{W}_t = \{\boldsymbol{w} : \|\langle \boldsymbol{w}, \boldsymbol{x}_t\rangle\| \leq C\}$	$C = 3\max_t \|\langle \boldsymbol{u}, \boldsymbol{x}_t\rangle\|$	$\cdot$
MGDiag	$3\|\boldsymbol{u}\|_\infty$	$\mathcal{W}_t = \{\boldsymbol{w} : \|\boldsymbol{w}\|_\infty \leq D\}$	$D = 3\|\boldsymbol{u}\|_\infty$	$\cdot$
MGF1	$3\|\boldsymbol{u}\|_2$	$\mathcal{W}_t = \{\boldsymbol{w} : \|\langle \boldsymbol{w}, \boldsymbol{x}_t\rangle\| \leq C\}$	$C = 3\max_t \|\langle \boldsymbol{u}, \boldsymbol{x}_t\rangle\|$	$\cdot$
MGF10	$3\|\boldsymbol{u}\|_2$	$\mathcal{W}_t = \{\boldsymbol{w} : \|\langle \boldsymbol{w}, \boldsymbol{x}_t\rangle\| \leq C\}$	$C = 3\max_t \|\langle \boldsymbol{u}, \boldsymbol{x}_t\rangle\|$	$\cdot$

Table 5.1: The settings of each algorithm.

Dataset	T	d	Outcome	P(y = 1)
a9a	32561	123	binary	0.24
abalone	4177	8	real	
australian	690	14	binary	0.44
bodyfat	252	14	real	
breast-cancer	683	9	binary	0.35
covtype	581012	54	binary	0.49
cpusmall	8192	12	real	
diabetes	768	8	binary	0.65
heart	270	13	binary	0.44
housing	506	13	real	
ijcnn1	91701	22	binary	0.10
ionosphere	351	34	binary	0.64
mg	1385	6	real	
space_ga	3107	6	real	
splice	1000	60	binary	0.52
w1atest	47272	300	binary	0.03
w8a	49479	300	binary	0.03

Table 5.2: Summary of the datasets used in the experiments.

Dataset	Loss	AdaGrad	GDnorm	GDt	MGCo	MGF1	MGF10	MGF25	MGF50	MGFull
a9a	hinge	85411	12712	**7619**	17012	14064	10821	10354	9858	9743
a9a	logistic	9185	2158	**1109**	1340	2306	1732	1668	1590	1568
abalone	absolute	4144	2529	1959	1317	2738	**938**	**938**	**938**	939
abalone	squared	23051	12484	10545	6725	9607	**5900**	**5900**	**5900**	5901
australian	hinge	124	42	**32**	41	46	35	35	35	35
australian	logistic	511	156	112	48	42	39	39	39	**39**
bodyfat	absolute	125	38	30	30	34	24	24	24	**24**
bodyfat	squared	60	5	**4**	10	10	10	10	10	10
breast-cancer	hinge	98	36	28	**24**	25	25	25	25	25
breast-cancer	logistic	107	41	33	**25**	36	36	36	36	36
covtype	hinge	445023	83124	48461	66797	121064	67126	59301	42043	**41985**
covtype	logistic	24043	11065	5157	**4713**	17698	10009	8662	5155	5147
cpusmall	absolute	183731	67098	61563	40537	89234	13974	**13818**	**13818**	13946
cpusmall	squared	2671536	806408	894232	561505	728070	**358831**	358832	358832	358833
diabetes	hinge	203	107	91	75	95	56	56	56	**55**
diabetes	logistic	147	80	58	53	54	**49**	**49**	**49**	49
heart	hinge	127	77	59	**35**	46	38	38	38	38
heart	logistic	127	71	47	30	30	30	30	30	**30**
housing	absolute	3301	1282	1147	946	1044	888	886	886	**886**
housing	squared	33324	**15560**	15909	20191	22244	22333	22336	22336	22336
ijcnn1	hinge	4413	1216	**537**	885	1522	1156	883	883	883
ijcnn1	logistic	4912	1404	**795**	976	1559	1219	1013	1013	1013
ionosphere	hinge	1245	431	299	169	166	**150**	150	150	150
ionosphere	logistic	2564	730	480	240	172	157	156	**156**	156
mg	absolute	85	33	**26**	30	45	29	29	29	28
mg	squared	31	10	**4**	19	28	28	28	28	28
space_ga	absolute	441	314	208	133	370	92	92	92	**91**
space_ga	squared	354	115	71	**40**	69	53	53	53	53
splice	hinge	1296	369	315	243	242	235	234	234	**225**
splice	logistic	1636	323	293	183	175	173	171	168	**166**
w1atest	hinge	134146	52780	67627	16910	17951	17436	16815	17143	**16636**
w1atest	logistic	28340	7500	8735	2498	2436	2271	2229	2207	**2182**
w8a	hinge	152227	60303	90302	**18789**	20764	19872	19783	19239	19229
w8a	logistic	36683	8370	13620	3324	2725	2519	2449	2421	**2392**

Table 5.3: The regret of each algorithm for the various datasets and loss functions. Boldface indicates smallest regret.