Journal of Machine Learning Research 22 (2021) 1-61 Submitted 12/20; Published 7/21

MetaGrad: Adaptation using Multiple Learning Rates in Online Learning[*]

Tim van Erven TIM@TIMVANERVEN.NL
Korteweg-de Vries Institute for Mathematics
University of Amsterdam
Science Park 107, 1098 XG Amsterdam, The Netherlands

Wouter M. Koolen WMKOOLEN@CWI.NL
Centrum Wiskunde & Informatica
Science Park 123, 1098 XG Amsterdam, The Netherlands

Dirk van der Hoeven DIRKVDERHOEVEN@GMAIL.COM
Mathematical Institute
Leiden University
Niels Bohrweg 1, 2300 RA Leiden, The Netherlands

Editor: Mehryar Mohri

Abstract

We provide a new adaptive method for online convex optimization, MetaGrad, that is robust to general convex losses but achieves faster rates for a broad class of special functions, including exp-concave and strongly convex functions, but also various types of stochastic and non-stochastic functions without any curvature. 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, which are weighted directly proportional to their empirical performance on the data using a new 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 evaluate all versions of MetaGrad on benchmark online classification and regression tasks, on which they consistently outperform both online gradient descent and AdaGrad.

Keywords: online convex optimization, adaptivity

[*] An earlier conference version of this paper appeared at NeurIPS 2016 (Van Erven and Koolen, 2016).

1. Introduction

Methods for *online convex optimization* (OCO) (Shalev-Shwartz, 2012; Hazan, 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 $f_1, f_2, \dots, f_T$ 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* $f_t(\boldsymbol{w}) = \max\{0, 1 - y_t \boldsymbol{w}^\intercal \boldsymbol{x}_t\}$ or the *logistic loss* $f_t(\boldsymbol{w}) = \ln\left(1 + e^{-y_t \boldsymbol{w}^\intercal \boldsymbol{x}_t}\right)$, 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* (OGD). OGD 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*

$$R_T^{\boldsymbol{u}} = \sum_{t=1}^{T} f_t(\boldsymbol{w}_t) - \sum_{t=1}^{T} f_t(\boldsymbol{u})$$

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 $\eta_t \propto 1/\sqrt{t}$, OGD guarantees that the regret for general convex Lipschitz 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 $O(\ln T)$ can be achieved by OGD with much smaller learning rates $\eta_t \propto 1/t$ (Hazan et al., 2007), and, if they are *exp-concave*, then logarithmic regret $O(d \ln T)$ 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 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 2 in Section 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 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 OGD 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 by 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. Assuming that the radius 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, Theorem 7 and Corollary 8 imply that MetaGrad's regret is simultaneously bounded by

$$R_T^{\boldsymbol{u}} = O(\sqrt{T \ln\ln T}) \tag{1}$$

and by

$$R_T^{\boldsymbol{u}} \;\leq\; \sum_{t=1}^{T} (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t \;=\; O\left(\sqrt{V_T^{\boldsymbol{u}}\, d \ln(T/d)} + d\ln(T/d)\right) \tag{2}$$

for any $\boldsymbol{u} \in \mathcal{W}$, where

$$V_T^{\boldsymbol{u}} := \sum_{t=1}^{T} \left((\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t\right)^2 .$$

The inequality $R_T^{\boldsymbol{u}} \leq \tilde{R}_T^{\boldsymbol{u}} := \sum_{t=1}^{T} (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t$ is a direct consequence of convexity of the loss and holds for any learning algorithm, so the important part of (2) is that it bounds $\tilde{R}_T^{\boldsymbol{u}}$ 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 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 9 we discuss a recent extension of MetaGrad by Wang et al. (2020) that removes this limitation.

The main difficulty in achieving the regret guarantee in (2) 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, one for each candidate 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 additive regret overhead for learning the best expert is not of the usual order $O(\sqrt{T})$, which would dwarf all desired benefits, but only costs a negligible $O(\ln \ln T)$.

The experts are instances of exponential weights on the continuous parameters $\boldsymbol{w}$ with a quadratic surrogate loss function, which in particular causes the exponential weights distributions to be multivariate Gaussian. The resulting updates are closely related to the ONS algorithm on the original losses, with the twist that here each expert receives the controller's gradients instead of its own, so only a single gradient (for the controller) needs to be calculated per round. We start and stop experts on the fly using a dynamic grid of exponentially spaced η-values, which guarantees that at most $\lceil \log_2 T \rceil$ experts are active at any given time. Since $\lceil \log_2 T \rceil \leq 30$ as long as $T \leq 10^9$, this seems computationally acceptable. If not, then the number of experts can be further reduced at the cost of slightly worse constants in the bound by spacing the η in the grid further apart.

The version of MetaGrad described so far maintains a full covariance matrix $\boldsymbol{F}_T = \sum_{t=1}^T \boldsymbol{g}_t \boldsymbol{g}_t^\intercal$ of size $d \times d$, where d is the parameter dimension. This requires $O(d^2)$ computation steps per round to update, which is prohibitive for large d. We therefore also present two extensions that require less computation: one based on sketching and one that works coordinatewise. The sketching extension applies the matrix sketching approach of Luo et al. (2017) to approximate $\boldsymbol{F}_T$ by a sketch of its top $m-1$ eigenvectors, and requires $O(md)$ amortised update time per round. As shown in Theorem 12, the price we pay for the improved run-time is that (2) is replaced by

$$\tilde{R}_T^{\boldsymbol{u}} = O\left(\sqrt{(V_T^{\boldsymbol{u}} + \Omega_{m-1})\; d \ln(T/d)} + d\ln(T/d)\right),$$

which includes an extra term $\Omega_{m-1} = \sum_{i=m}^d \lambda_i$ to account for the remaining eigenvalues in $\boldsymbol{F}_T$ that are not captured by the sketch. Thus the hyperparameter m provides a trade-off between regret and run-time.

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 OGD and AdaGrad. To avoid interactions between coordinates, we restrict attention to rectangular domains. Whether this restriction can be lifted is not clear, as discussed in Section 7.2. The main regret bound for the coordinatewise extension is obtained by summing the regret bound (2) over the coordinates:

$$\tilde{R}_T^{\boldsymbol{u}} = O\left(\sum_{i=1}^d \sqrt{V_{T,i}^{u_i} \ln(T)} + d\ln(T)\right), \tag{3}$$

where $V_{T,i}^{u_i} = \sum_{t=1}^{T}(u_i - w_{t,i})^2 g_{t,i}^2$ is the coordinatewise variance. This is established by Theorem 13 and Corollary 14, which also show that the coordinate extension simultaneously guarantees regret of order

$$\tilde{R}_T^{\boldsymbol{u}} \;=\; O\left(\sum_{i=1}^{d} \|g_{1:T,i}\|_2\sqrt{\ln\ln T} + \sqrt{d}\ln\ln T\right) \;=\; O\left(\sqrt{dT\ln\ln T}\right), \tag{4}$$

where $g_{1:T,i} := (g_{i,1}, \ldots, g_{i,T})$. 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 (i.e., dense parameter vectors). Since the coordinate version does not keep track of a full covariance matrix, we cannot expect it to exploit the Bernstein condition for stochastic gradients in all cases. Section 7.2.2 therefore introduces a more stringent coordinate Bernstein condition, under which (3) does always imply fast rates, and Theorem 16 gives sufficient conditions under which the general Bernstein condition implies the coordinate Bernstein condition. It is appealing that the coordinatewise MetaGrad extension simultaneously satisfies (4), because (up to the $\sqrt{\ln\ln T}$ factor) this recovers the diagonal AdaGrad bound of $O(\sum_{i=1}^{d}\|g_{1:T,i}\|_2)$, which can take advantage of sparse gradients (Duchi et al., 2011).

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 Lipschitz-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 $\max_{\boldsymbol{w}\in\mathcal{W}} |(\boldsymbol{w}-\boldsymbol{w}_t)^\intercal \boldsymbol{g}_t|$ instead of the larger $\max_{\boldsymbol{w}\in\mathcal{W}} \|\boldsymbol{w}-\boldsymbol{w}_t\|_2\|\boldsymbol{g}_t\|_2$. The difference is crucial in Section 5.1, where we consider a time-varying domain introduced by Luo et al. (2017) in the context of sketching: this domain is bounded only in the direction of the gradients, so our norms are under control, but has infinite diameter in all orthogonal directions.

We conclude the paper with an empirical evaluation in which we compare our new algorithms (the Full, Sketching and Coordinatewise versions of MetaGrad) with AdaGrad and OGD on 17 real-world LIBSVM regression and classification data sets. Our experiments show that the full-matrix version of MetaGrad beats previous methods in all but one of our experiments and delivers competitive performance throughout. Moreover, we see that the sketching extension provides a controlled trade-off between regret and run-time, while the fastest, coordinatewise version of MetaGrad still works surprisingly well in the majority of experiments.

1.1 Related Work

If we disregard computational efficiency and omit Lipschitz-adaptivity, then the guarantee from (2) 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.

Luo et al. (2017) show a lower bound on the regret of $\Theta(\sqrt{dT})$ for the time-varying domain mentioned above, and obtain a nearly matching upper bound of $O(\sqrt{dT}\ln T)$ using a variant of ONS. Our Theorem 7, which implies (2) when the radius of the domain is bounded, is actually more general and also covers the time-varying domain. For this domain it improves on the upper bound of Luo et al. by replacing the dependence on T by $V_T^{\boldsymbol{u}}$ and by moving the log-factor into the square root. Section 6.3 provides a detailed comparison.

As already mentioned, Wang et al. (2020) extend MetaGrad to adapt to strongly convex functions. Zhang et al. (2019) 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 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 σ 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 works 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.

1.2 Outline

We start with the main definitions in the next section. Then Section 3 contains an extensive set of examples where the guarantee from (2) leads to fast rates, Section 4 presents the Full Matrix version of the MetaGrad algorithm, and Section 5 describes the faster sketching and coordinatewise extensions. Section 6 provides the analysis leading to Theorem 7 for the Full Matrix version of MetaGrad, which is a more detailed statement of (2) with several quantities replaced by data-dependent versions and with exact constants. Section 7 extends this analysis to the two other versions of MetaGrad. Then, in Section 8, we compare all versions of MetaGrad to OGD and to AdaGrad in experiments with several benchmark classification and regression data sets. We conclude with a discussion of possible further

Protocol 1: Online Convex Optimization with First-order Information

```
for $t = 1, 2, \dots$ 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
```

extensions of MetaGrad in Section 9. Finally, most details of the proofs are postponed to the appendix.

2. Setup

We consider algorithms for OCO, which operate according to the protocol displayed in Protocol 1. In each round, the environment reveals a closed convex domain $\mathcal{W}_t \subset \mathbb{R}^d$, 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 5.1). 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.[1] 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 $f_t(\boldsymbol{w}_t) - f_t(\boldsymbol{u}) \le (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t$, which implies the first inequality in (2): $R_T^{\boldsymbol{u}} \le \tilde{R}_T^{\boldsymbol{u}}$. Finally, wherever possible we measure the size of the gradient $\boldsymbol{g}_t$ in the following (semi-)norm:

$$\|\boldsymbol{g}\|_t = \max_{\boldsymbol{w} \in \mathcal{W}_t} |(\boldsymbol{w} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}|,$$

which takes into account the shape of the domain, and is centered at the learner's predictions $\boldsymbol{w}_t$. This is a norm in the typical case that $\mathcal{W}_t$ has full dimension d, and it is still a

[1] 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. Since f_t is convex, there always exists at least one subgradient when $\boldsymbol{w}_t$ is in the interior of its domain. Existence of subgradients on the boundary of $\mathcal{W}_t$ is guaranteed, for instance, if there exists a finite convex extension of f_t to $\mathbb{R}^d$.

semi-norm in general. We note that this norm is smaller than the usual upper bounds based on Hölder's inequality: $\|\boldsymbol{g}\|_t \leq \|\boldsymbol{g}\|_* \max_{\boldsymbol{w}\in\mathcal{W}_t} \|\boldsymbol{w} - \boldsymbol{w}_t\|$ for any dual norms $\|\cdot\|$ and $\|\cdot\|_*$. The difference becomes essential in Section 5.1, where we consider a domain $\mathcal{W}_t$ that has an infinite radius $\max_{\boldsymbol{w}\in\mathcal{W}_t} \|\boldsymbol{w} - \boldsymbol{w}_t\|$ 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:

$$b_t \geq \|\boldsymbol{g}_t\|_t, \qquad\qquad B_t = \max_{s\leq t} b_s, \tag{5}$$

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 $b_t = \|\boldsymbol{g}_t\|_2 \max_{\boldsymbol{u},\boldsymbol{w}\in\mathcal{W}_t} \|\boldsymbol{u} - \boldsymbol{w}\|_2$. We assume throughout that $B_T > 0$; otherwise the regret is trivially bounded by zero.

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

3. Fast Rates Examples

In this section, we motivate our interest in the adaptive bound (2) by giving a series of examples in which it provides fast rates. For simplicity, we will in this section assume that the domain is fixed: $\mathcal{W}_t = \mathcal{W}$, with bounded radius $D_2 \geq \max_{\boldsymbol{u}\in\mathcal{W}} \|\boldsymbol{u}\|_2$, and that all gradients have length at most $G_2 \geq \|\boldsymbol{g}_t\|_2$. 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. In Appendix A.1 we provide simple simulations illustrating these conditions, which are exploited by MetaGrad but not by AdaGrad. Proofs are also postponed to Appendix A.

3.1 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 1 *Suppose, for a given $\boldsymbol{u} \in \mathcal{W}$, there exist constants $a, b > 0$ such that the functions f_t all satisfy*

$$f_t(\boldsymbol{u}) \geq f_t(\boldsymbol{w}) + a(\boldsymbol{u} - \boldsymbol{w})^\intercal \nabla f_t(\boldsymbol{w}) + b\left((\boldsymbol{u} - \boldsymbol{w})^\intercal \nabla f_t(\boldsymbol{w})\right)^2 \qquad \textit{for all } \boldsymbol{w} \in \mathcal{W}. \tag{6}$$

Then any method with regret bound (2) *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 $O(d \ln T)$ 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 general $a > 0$ 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 $(\boldsymbol{u}^* - \boldsymbol{w})^\intercal \nabla f(\boldsymbol{w}) \in [-2D_2G_2, 0]$, so that

$$f(\boldsymbol{u}^*) - f(\boldsymbol{w}) \geq (\boldsymbol{u}^* - \boldsymbol{w})^\intercal \nabla f(\boldsymbol{w}) \geq 2(\boldsymbol{u}^* - \boldsymbol{w})^\intercal \nabla f(\boldsymbol{w}) + \frac{1}{2D_2G_2}\left((\boldsymbol{u}^* - \boldsymbol{w})^\intercal \nabla f(\boldsymbol{w})\right)^2,$$

where the first inequality uses only convexity of f. Thus condition (6) is satisfied by *any fixed convex function*, even if it does not have any curvature at all, with $a = 2$ and $b = 1/(2D_2G_2)$.

At first sight this may appear to contradict the lower bound of order $1/\sqrt{T}$ for convergence of the iterates by Nesterov (2004) (see also Tibshirani, 2014), which implies a lower bound of order $\sqrt{T}$ on the regret. Yet there is no contradiction, as Nesterov's example requires large dimension $d \geq T$, in which case $O(d \ln T)$ is vacuous. In Nesterov's example, MetaGrad still gets the $\sqrt{T}$ rate up to a $\ln \ln T$ factor, however, because it satisfies (1).

3.2 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 (2) 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

$$(\boldsymbol{w} - \boldsymbol{u}^*)^\intercal \mathop{\mathbb{E}}_f \left[\nabla f(\boldsymbol{w}) \nabla f(\boldsymbol{w})^\intercal\right] (\boldsymbol{w} - \boldsymbol{u}^*) \;\leq\; B\big((\boldsymbol{w} - \boldsymbol{u}^*)^\intercal \mathop{\mathbb{E}}_f \left[\nabla f(\boldsymbol{w})\right]\big)^\beta \qquad \text{for all } \boldsymbol{w} \in \mathcal{W}. \tag{7}$$

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 Appendix A.5, 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 2 *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* (2) *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 $O(d \ln T)$ rate, albeit with a B that depends on the distribution of the data. (See Appendix A.4.) In stochastic optimization for support vector machines, the hinge loss is combined with an additional ℓ_2-regularization term. It is sometimes argued that this term also gives fast rates, because it makes the loss strongly convex, but the amount of regularization used in practice is typically too small to get any significant improvements. The present example shows that, even without adding regularization to the loss, it is possible to get logarithmic regret.

4. Full Matrix Version of the MetaGrad Algorithm

In this section, we explain the full matrix version of the MetaGrad algorithm: MetaGrad Full. Computationally more efficient extensions follow in Section 5. MetaGrad Full will be defined by means of the following *surrogate loss* $\ell_t^\eta(\boldsymbol{u})$:

$$\ell_t^\eta(\boldsymbol{u}) \;:=\; \eta(\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t + \big(\eta(\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t\big)^2. \tag{8}$$

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 (2). They may be viewed as causing a "time-varying regularizer" (Orabona et al., 2015) or "temporal adaptation of the proximal function" (Duchi et al., 2011).

MetaGrad Full is a two-level hierarchical construction: at the top is a main controller, shown in Algorithm 1, which manages multiple η-experts, shown in Algorithm 2. 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.

4.1 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

Algorithm 1: MetaGrad Full: Controller

```
1:  for $t = 1, 2, \dots$ do
2:    Receive domain $\mathcal{W}_t$
3:    Start and stop $\eta$-experts to manage active set $\mathcal{A}_t$ (Equation 9). Give newly
      started $\eta$-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 $\eta$-experts project onto $\mathcal{W}_t$
8:      Collect prediction $\boldsymbol{w}_t^\eta$ for every active $\eta$-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 $b_t$ (Equation 5)
12:   Update every active $\eta$-expert with unclipped surrogate loss $\ell_t^\eta$
13:   if No reset needed after round $t$ (Equation 11) then
14:     Update based on the clipped surrogate losses (Equation 12):
        $p_{t+1}(\eta) = \frac{p_t(\eta) \exp\left(-\bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)\right)}{\sum_{\eta \in \mathcal{A}_t} p_t(\eta) \exp\left(-\bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)\right)} \left(\sum_{\eta \in \mathcal{A}_t} p_t(\eta)\right)$    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
```

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. Thus we never tune the controller's weights on learning rates based on any bounds, but always directly in terms of their empirical performance.

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. We will take exponentially spaced learning rates from the infinite grid

$$\mathcal{G} := \{2^i \mid i \in \mathbb{Z}\},$$

and the following learning rates are active in round t:

$$\mathcal{A}_t := \begin{cases} \emptyset & \text{while } B_{t-1} = 0, \\ \mathcal{G} \cap \left(\frac{1}{2\left(\sum_{s=1}^{t-1} b_s \frac{B_{s-1}}{B_s} + B_{t-1}\right)}, \frac{1}{2B_{t-1}} \right] & \text{afterwards.} \end{cases} \tag{9}$$

This means that a^η, the first round in which an η-expert is active, is

$$a^\eta = \min\left\{ t \in \{1, 2, \ldots\} \,\middle|\, \eta > \frac{1}{2\left(\sum_{s=1}^{t-1} b_s \frac{B_{s-1}}{B_s} + B_{t-1}\right)} \right\}. \tag{10}$$

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: $\boldsymbol{w}_t = \boldsymbol{0}$.

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

$$B_t > B_\tau \sum_{s=1}^{t} \frac{b_s}{B_s}. \tag{11}$$

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:

$$\bar{\ell}_t^\eta(\boldsymbol{u}) \;:=\; \eta(\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \bar{\boldsymbol{g}}_t + \big(\eta(\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \bar{\boldsymbol{g}}_t\big)^2, \tag{12}$$

which are based on the clipped gradients

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

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

4.2 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. 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 $\boldsymbol{\Sigma}_t^\eta = \left(\frac{1}{\sigma^2}\boldsymbol{I} + 2\eta^2 \sum_{s=a}^{t} \boldsymbol{g}_s \boldsymbol{g}_s^\intercal\right)^{-1}$.

```
Algorithm 2: MetaGrad Full: $\eta$-Expert
  Input: Learning rate $\eta > 0$, estimate $\sigma > 0$ of comparator norm $\|\boldsymbol{u}\|_2$, first active
         round $a \equiv a^\eta$
  1: Initialize $\check{\boldsymbol{w}}_a^\eta = \boldsymbol{0}$ and $\boldsymbol{\Lambda}_a^\eta = \frac{1}{\sigma^2}\boldsymbol{I}$      ▷ Invariant: $\boldsymbol{\Lambda}_{t+1}^\eta = \frac{1}{\sigma^2}\boldsymbol{I} + 2\eta^2 \sum_{s=a}^t \boldsymbol{g}_s \boldsymbol{g}_s^\intercal$
  2: Initialize $\boldsymbol{\Sigma}_a^\eta = \sigma^2 \boldsymbol{I}$      ▷ Invariant: $\boldsymbol{\Sigma}_t^\eta = (\boldsymbol{\Lambda}_t^\eta)^{-1}$
  3: for $t = a, a+1, \ldots$ do
  4:     Project $\boldsymbol{w}_t^\eta = \arg\min_{\boldsymbol{u} \in \mathcal{W}_t} (\boldsymbol{u} - \check{\boldsymbol{w}}_t^\eta)^\intercal \boldsymbol{\Lambda}_t^\eta (\boldsymbol{u} - \check{\boldsymbol{w}}_t^\eta)$
  5:     Predict $\boldsymbol{w}_t^\eta$
  6:     Observe gradient $\boldsymbol{g}_t = \nabla f_t(\boldsymbol{w}_t)$      ▷ Gradient at controller prediction $\boldsymbol{w}_t$
  7:     Update:
             $\boldsymbol{\Sigma}_{t+1}^\eta = \boldsymbol{\Sigma}_t^\eta - \frac{2\eta^2 (\boldsymbol{\Sigma}_t^\eta \boldsymbol{g}_t)(\boldsymbol{g}_t^\intercal \boldsymbol{\Sigma}_t^\eta)}{1 + 2\eta^2 \boldsymbol{g}_t^\intercal \boldsymbol{\Sigma}_t^\eta \boldsymbol{g}_t}$      ▷ Sherman-Morrison
             $\boldsymbol{\Lambda}_{t+1}^\eta = \boldsymbol{\Lambda}_t^\eta + 2\eta^2 \boldsymbol{g}_t \boldsymbol{g}_t^\intercal$
             $\check{\boldsymbol{w}}_{t+1}^\eta = \boldsymbol{w}_t^\eta - \left(1 + 2\eta(\boldsymbol{w}_t^\eta - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t\right) \eta \boldsymbol{\Sigma}_{t+1}^\eta \boldsymbol{g}_t$
  8: end for
```

Algorithm 2 presents the update equations in a computationally efficient form. To avoid inverting $\boldsymbol{\Sigma}_t^\eta$, it maintains its inverse $\boldsymbol{\Lambda}_t^\eta = (\boldsymbol{\Sigma}_t^\eta)^{-1}$ 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 factor $(1 + 2\eta(\boldsymbol{w}_t^\eta - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t)$ in line 7 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. 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.

4.3 Practical Considerations

Although MetaGrad 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 σ, which is an estimate of the ℓ_2-norm of the comparator $\boldsymbol{u}$. Theorem 7 quantifies the trade-off between underestimating and overestimating this parameter. As discussed below the theorem, underestimating σ always harms the rate. But, for low-dimensional settings, overestimating σ only incurs a logarithmic penalty, so it is much less expensive to use a too large value than to use a too small value. For high-dimensional settings the dependence on σ is similar to the usual dependence of Online Gradient Descent on a guess for $\|\boldsymbol{u}\|_2$, so the rate deteriorates linearly when taking σ too large.

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 7 already scales linearly with the size of the gradients. In fact, the MetaGrad Full algorithm itself is almost invariant under such rescaling, except for the grid $\{2^i \mid i \in \mathbb{Z}\}$ in the definition of $\mathcal{A}_t$. If one wants to make the algorithm fully invariant under rescaling, the grid 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 $t \geq \tau$. Since we do not expect any noticeable difference in performance from this modification, we have left it out.

4.3.1 RUN TIME

The run time of MetaGrad 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 $O(kd^2)$, 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 $\boldsymbol{\Sigma}_t^\eta$ 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 $\boldsymbol{\Sigma}_t^\eta$ 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 5.1), 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. Faster Extension Algorithms

As discussed above, MetaGrad 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 $\boldsymbol{\Sigma}_t^\eta$ used in each η-expert. This reduces the computation per round to $O(kd)$, where k is a hyperparameter 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.

5.1 Sketched MetaGrad with Closed-form Projections

Algorithm 3: Sketched η-Expert

```
Input: Learning rate $\eta > 0$, estimate $\sigma > 0$ of comparator norm $\|\boldsymbol{u}\|_2$, first active
       round $a \equiv a^\eta$
Initialize $\check{\boldsymbol{w}}_a^\eta = \mathbf{0}$
Get $\boldsymbol{S}_{a-1}^\eta$ and $\boldsymbol{H}_{a-1}^\eta$ from initialisation of Frequent Directions Sketching
Algorithm 4
for $t = a, a+1, \ldots$ do
    Observe feature vector $\boldsymbol{x}_t$
    Obtain $\boldsymbol{w}_t^\eta$ by projection (16)
    Issue prediction $\boldsymbol{w}_t^\eta$
    Observe gradient $\boldsymbol{g}_t = \nabla f_t(\boldsymbol{w}_t)$        ▷ Gradient at controller prediction $\boldsymbol{w}_t$
    Send $\boldsymbol{g}_t$ to Frequent Directions Sketching Algorithm 4 and receive $\boldsymbol{S}_t^\eta$ and $\boldsymbol{H}_t^\eta$
    Update $\check{\boldsymbol{w}}_{t+1}^\eta$ as per (17)
end for
```

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.

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 $h_t : \mathbb{R} \to \mathbb{R}$ become available only after the prediction has been made. They then choose the domain to be

$$\mathcal{W}_t = \{\boldsymbol{w} : |\boldsymbol{w}^\intercal \boldsymbol{x}_t| \le C\} \qquad \text{for a fixed constant } C. \tag{13}$$

Let a^η be the round in which the η-expert is first activated and define $\boldsymbol{G}_t^\eta = (\boldsymbol{g}_{a^\eta}, \ldots, \boldsymbol{g}_t)^\intercal \in \mathbb{R}^{(t-a^\eta+1)\times d}$, such that $\boldsymbol{\Sigma}_{t+1}^\eta = (\frac{1}{\sigma^2}\boldsymbol{I}_d + 2\eta^2(\boldsymbol{G}_t^\eta)^\intercal \boldsymbol{G}_t^\eta)^{-1}$. The idea of sketching is to replace $\boldsymbol{\Sigma}_{t+1}^\eta \in \mathbb{R}^{d\times d}$ by an approximation

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

where $\boldsymbol{S}_t^\eta \in \mathbb{R}^{k \times d}$ for a given *sketch size* k that can be much smaller than d, so that $(\boldsymbol{S}_t^\eta)^\intercal \boldsymbol{S}_t^\eta$ has rank at most k. Abbreviating

$$\boldsymbol{g}_t^\eta = (1 + 2\eta(\boldsymbol{w}_t^\eta - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t)\,\eta \boldsymbol{g}_t, \tag{14}$$

we then need to compute

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

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

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

where we have introduced the abbreviation

$$\boldsymbol{H}_t^\eta = (\tfrac{1}{\sigma^2}\boldsymbol{I}_k + 2\eta^2 \boldsymbol{S}_t^\eta (\boldsymbol{S}_t^\eta)^\intercal)^{-1}. \tag{15}$$

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

$$\boldsymbol{w}_t^\eta = \breve{\boldsymbol{w}}_t^\eta - \frac{\mathrm{s}_C(\boldsymbol{x}_t^\intercal \breve{\boldsymbol{w}}_t^\eta)}{(\boldsymbol{x}_t^\intercal \boldsymbol{x}_t - 2\eta^2 \boldsymbol{x}_t^\intercal (\boldsymbol{S}_{t-1}^\eta)^\intercal \boldsymbol{H}_{t-1}^\eta \boldsymbol{S}_{t-1}^\eta \boldsymbol{x}_t)} (\boldsymbol{x}_t - 2\eta^2 (\boldsymbol{S}_{t-1}^\eta)^\intercal \boldsymbol{H}_{t-1}^\eta \boldsymbol{S}_{t-1}^\eta \boldsymbol{x}_t), \tag{16}$$

and the update step can be written (with $\boldsymbol{g}_t^\eta$ as in Equation 14) as

$$\breve{\boldsymbol{w}}_{t+1}^\eta = \boldsymbol{w}_t^\eta - \sigma^2(\boldsymbol{g}_t^\eta - 2\eta^2(\boldsymbol{S}_t^\eta)^\intercal \boldsymbol{H}_t^\eta \boldsymbol{S}_t^\eta \boldsymbol{g}_t^\eta). \tag{17}$$

Assuming that $\boldsymbol{S}_t^\eta$ and $\boldsymbol{H}_t^\eta$ can be efficiently maintained, the operations involving $\boldsymbol{S}_t^\eta \boldsymbol{x}_t$ or $\boldsymbol{S}_t^\eta \boldsymbol{g}_t^\eta$ 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 $\boldsymbol{S}_t^\eta$ and $\boldsymbol{H}_t^\eta$, each requiring $O(kd)$ storage and $O(kd)$ amortised 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 in the literature is the Robust Frequent Directions sketching method of Luo et al. (2019).

5.1.1 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 $\boldsymbol{S}_t^\eta$ every round at the cost of $O(k^2 d)$

Algorithm 4: Frequent Directions Sketching

```
Input: Sketch rank $m$, first active round $a \equiv a^\eta$
Initialize $\boldsymbol{S}_{a-1}^\eta = \boldsymbol{0}_{2m \times d}$, and $\boldsymbol{H}_{a-1}^\eta = \sigma^2 \boldsymbol{I}_{2m}$.
for $t = a, a+1, \ldots$ do
    Receive $\boldsymbol{g}_t$
    Let $\tau = (t-a) \bmod (m+1)$ and write $\boldsymbol{g}_t^\intercal$ to row $(m+\tau)$ of $\boldsymbol{S}_{t-1}^\eta$ to obtain $\tilde{\boldsymbol{S}}$
    if $\tau < m$ then
        Set $\boldsymbol{S}_t^\eta = \tilde{\boldsymbol{S}}$
        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 $\boldsymbol{S}_t^\eta = \begin{pmatrix} \operatorname{diag}(\sigma_1^2 - \sigma_m^2, \ldots, \sigma_m^2 - \sigma_m^2)^{1/2}\boldsymbol{V}^\intercal \\ \boldsymbol{0}_{m \times d} \end{pmatrix}$
        Set $\boldsymbol{H}_t^\eta = \operatorname{diag}(\frac{1}{\sigma^{-2}+2\eta^2(\sigma_1^2-\sigma_m^2)}, \ldots, \frac{1}{\sigma^{-2}+2\eta^2(\sigma_m^2-\sigma_m^2)}, \frac{1}{\sigma^{-2}}, \ldots, \frac{1}{\sigma^{-2}})$
    end if
end for
```

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 amortised 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 4.

Recall that $(\boldsymbol{S}_t^\eta)^\intercal \boldsymbol{S}_t^\eta$ is an approximation of $(\boldsymbol{G}_t^\eta)^\intercal \boldsymbol{G}_t^\eta$. At the start of each epoch, we have the invariant that only the first $m-1$ rows of $\boldsymbol{S}_t^\eta$ contribute to this approximation and the remaining $m+1$ rows are filled with zeros. During the τ-th round in any epoch we first write the incoming gradient $\boldsymbol{g}_t^\intercal$ to row $m+\tau$ of $\boldsymbol{S}_{t-1}^\eta$ 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 $\boldsymbol{S}_t^\eta = \tilde{\boldsymbol{S}}$, and use (15) to see 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 $\boldsymbol{S}_t^\eta = \operatorname{diag}(\lambda_1 - \lambda_m, \ldots, \lambda_m - \lambda_m, 0, \ldots, 0)^{1/2} \boldsymbol{W}^\intercal$. Since the rows of $\boldsymbol{S}_t^\eta$ are now orthogonal,

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

is a diagonal matrix.

5.1.2 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 $\boldsymbol{S}_t^\eta = \operatorname{diag}(\lambda_1 - \lambda_m, \ldots, \lambda_m - \lambda_m, 0, \ldots, 0)^{1/2} \boldsymbol{W}^\intercal = \operatorname{diag}(\sigma_1^2 - \sigma_m^2, \ldots, \sigma_m^2 - \sigma_m^2)^{1/2} \boldsymbol{V}^\intercal$.

5.1.3 Practical Considerations

Sketching introduces an extra hyperparameter $k = 2m$, which controls the sketch size. The sketch keeps track of $m - 1$ dimensions, so in theory we expect that larger k provides a better approximation of the full version of MetaGrad, at the cost of more computation. We indeed observe this in practice in the experiments in Section 8.

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 $f_{t,i}(w) = w g_{t,i}$, 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:

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

with our only freedom consisting of choosing the side lengths $D_{t,i}$.

5.2.1 PRACTICAL CONSIDERATIONS

The bounds b_t on the gradients now become a separate bound per dimension:

$$b_{t,i} := \max_{w_i \in [-D_{t,i}, D_{t,i}]} |(w_i - w_{t,i}) g_{t,i}| = (D_{t,i} + |w_{t,i}|)|g_{t,i}|, \qquad B_{t,i} = \max_{s \leq t} b_{s,i}.$$

Running a copy of MetaGrad per dimension potentially introduces a separate hyperparameter σ_i per dimension i. Like Duchi et al. (2011), we reduce the complexity of hyperparameter tuning by letting $\sigma_i = \sigma$ be the same for all dimensions. In line with the discussion in Section 4.3, the recommended setting for σ then becomes (an overestimate of) the ℓ_∞-norm of the comparator $\boldsymbol{u}$. 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 $D_{t,i} = D_\infty$ for a fixed parameter D_∞.

6. Analysis of the Full Matrix Version of MetaGrad

Recall that MetaGrad runs multiple instances of a baseline "η-expert" algorithm, each with a different candidate tuning of the learning rate η. A controller then aggregates the predictions of these η-experts and manages their lifetimes to always have the required tuning present. The MetaGrad Full η-experts are Exponentially Weighted Average forecasters starting from a Gaussian prior and taking in our quadratic surrogate losses. In turn, the controller is a specialists (aka sleeping experts) algorithm to deal with the starting and retiring of η-experts. When measured in the surrogate loss, the controller ensures a uniform regret bound w.r.t. each η-expert. Yet in the original loss, which is not scaled by η, this results in 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.

6.1 Controller

Let us introduce the concept of expiration to capture when η-experts become inactive and are never used again:

Definition 3 *We say that $\eta \in \mathcal{G}$ is* expired *after T rounds (or, equivalently, after round T) if $\eta > \frac{1}{2B_{T-1}}$.*

Note that expiration can be checked *before* the round happens (it is "predictable"). All learning rates used by Algorithm 1 by means of the active set $\mathcal{A}_t$ (9) 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 its behavior approximates that of any η-expert not expired, when measured in the η surrogate loss (8).

Lemma 4 (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 Full ensures*

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

where we interpret the last sum as 0 if $a^\eta > T$.

The proof is in Appendix B. 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 tiny. 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 with $\eta \in \mathcal{G}$ as our controller algorithm. Of note here is our use of a uniform prior on $\mathcal{G}$, which is improper in the sense that it does not sum to one. Improperness does not cause any problems, because the prior is automatically renormalized on the sets of active learning rates $\mathcal{A}_1, \mathcal{A}_2, \ldots$ 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.

6.2 Full η-Experts

Next we establish an $O(d \ln T)$ regret bound in terms of the surrogate loss for each MetaGrad Full η-expert. The η-experts implement the exponentially weighted average forecaster for the quadratic losses ℓ_t^η starting from a Gaussian prior. Alternatively, they may be viewed as instances of mirror descent with a time-varying quadratic regularizer. The exponentially weighted average forecaster was previously used for a different quadratic loss arising in linear regression by Vovk (2001). Mirror descent for the general quadratic case goes back (at least) to Hazan et al. (2007). Although they do not separate the analysis for general quadratic losses from the reduction of exp-concave losses to quadratics, the 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 5 (Surrogate regret bound) *Consider the MetaGrad Full η-expert in Algorithm 2 with learning rate $\eta \leq \frac{1}{2B_T}$ 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

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

We note that the condition on η in the lemma is slightly stricter than not being expired (Definition 3), which only requires $\eta \leq \frac{1}{2B_{T-1}}$. The reason is that the η-expert operates off the *unclipped* surrogate loss and gradients.

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

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

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 $(\boldsymbol{\Sigma}_{t+1}^\eta)^{-1} \;=\; \frac{1}{\sigma^2}\boldsymbol{I} + 2\eta^2\sum_{s=a^\eta}^t \boldsymbol{g}_s\boldsymbol{g}_s^\intercal$. Expanding, we obtain

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

Now we may use that

$$\frac{1}{2}\left(1 + 2\eta\left\langle \boldsymbol{w}_t - \boldsymbol{w}_t^\eta, \boldsymbol{g}_t\right\rangle\right)^2 \leq \frac{1}{2}(1+2\eta b_t)^2 \leq \frac{1}{2}(1+1)^2 = 2 \tag{18}$$

by the assumed upper bound on η. Moreover, abbreviating $\boldsymbol{A} = \frac{1}{\sigma^2}\boldsymbol{I} + 2\eta^2\sum_{s=a^\eta}^t \boldsymbol{g}_s\boldsymbol{g}_s^\intercal$ and $\boldsymbol{B} = \frac{1}{\sigma^2}\boldsymbol{I} + 2\eta^2\sum_{s=a^\eta}^{t-1} \boldsymbol{g}_s\boldsymbol{g}_s^\intercal$, concavity of the log determinant implies that

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

(Lemma 12 of Hazan et al. (2007) provides a detailed proof of this inequality.) Summing over rounds and telescoping, we find

$$\frac{1}{2}\sum_{t=a^\eta}^T \boldsymbol{g}_t'^\intercal \boldsymbol{\Sigma}_{t+1}^\eta \boldsymbol{g}_t' \;\leq\; \ln\det\left(\boldsymbol{I} + 2\eta^2\sigma^2 \sum_{t=a^\eta}^T \boldsymbol{g}_t\boldsymbol{g}_t^\intercal\right)$$

and obtain the result. ■

6.3 Composition (bounding the actual regret)

To complete the analysis of MetaGrad 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 *gradient covariance matrix* and *essential horizon* by

$$\boldsymbol{F}_T \;:=\; \sum_{t=1}^{T} \boldsymbol{g}_t \boldsymbol{g}_t^\intercal \qquad \text{and} \qquad Q_T \;:=\; \sum_{t=1}^{T-1} \frac{b_t}{B_t} + 1. \tag{19}$$

Theorem 6 (Grid point regret) *MetaGrad Full guarantees that the linearized regret w.r.t. any comparator $\boldsymbol{u} \in \bigcap_{t=1}^{T} \mathcal{W}_t$ is at most*

$$\tilde{R}_T^{\boldsymbol{u}} \;\le\; \eta V_T^{\boldsymbol{u}} + \frac{\ln\det\left(\boldsymbol{I} + 2\eta^2\sigma^2\boldsymbol{F}_T\right) + \frac{1}{2\sigma^2}\|\boldsymbol{u}\|_2^2 + 2\ln\lceil 2\log_2 Q_T\rceil_+ + \frac{1}{2}}{\eta} + 2B_T,$$

simultaneously for all $\eta \in \mathcal{G}$ such that $\eta \le \frac{1}{2B_T}$.

Proof Combining the controller and η-expert surrogate regret bounds from Lemma 4 and Lemma 5, we obtain

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

The definition of the surrogate loss (8) gives $\ell_t^\eta(\boldsymbol{w}_t) - \ell_t^\eta(\boldsymbol{u}) = \eta(\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t - \big(\eta(\boldsymbol{u} - \boldsymbol{w}_t)^\intercal \boldsymbol{g}_t\big)^2$ 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 6. This leads to our main result. The proof is in Appendix C.

Theorem 7 (MetaGrad Full Regret Bound) *For all $\boldsymbol{u} \in \bigcap_{t=1}^{T} \mathcal{W}_t$ the linearized regret of MetaGrad Full is simultaneously bounded by*

$$\tilde{R}_T^{\boldsymbol{u}} \;\le\; \frac{5}{2}\sqrt{V_T^{\boldsymbol{u}}\left(\tfrac{1}{2\sigma^2}\|\boldsymbol{u}\|_2^2 + Z_T\right)} + 5B_T\left(\tfrac{1}{2\sigma^2}\|\boldsymbol{u}\|_2^2 + Z_T\right) + 2B_T, \tag{20}$$

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

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

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

Here the rank $\mathrm{rk}(\boldsymbol{F}_T) \leq d$ plays the role of an effective dimension. If the eigenvalues of $\boldsymbol{F}_T$ satisfy a decay condition, then a more refined bound on Z_T is possible, as can be seen from the proof. The recommended tuning is to set σ to (an upper bound on) $\|\boldsymbol{u}\|_2$. For this case, we obtain the following corollary, which is proved in Appendix D:

Corollary 8 *Suppose the domain $\mathcal{W}_t = \mathcal{W}$ is fixed with finite radius $D_2 := \max_{\boldsymbol{u}\in\mathcal{W}} \|\boldsymbol{u}\|_2$, and we tune $\sigma = D_2$. Then, if all gradients are uniformly bounded by $\|\boldsymbol{g}_t\|_2 \leq G_2$, the linearized regret of MetaGrad Full with respect to any $\boldsymbol{u} \in \mathcal{W}$ is bounded by*

$$\tilde{R}_T^{\boldsymbol{u}} \;=\; O\left(\sqrt{V_T^{\boldsymbol{u}}\, d \ln \frac{D_2 G_2 T}{d}} + D_2 G_2 d \ln\left(\frac{D_2 G_2 T}{d}\right)\right), \tag{21}$$

and it is simultaneously bounded by

$$\tilde{R}_T^{\boldsymbol{u}} \;=\; O\left(D_2 G_2 \sqrt{T \ln\ln T}\right).$$

6.3.1 SENSITIVITY TO σ-TUNING

The recommended tuning for σ is to set it to (an upper bound on) $\|\boldsymbol{u}\|_2$. In the first result of Theorem 7, which covers the regime that $\mathrm{rk}(\boldsymbol{F}_T)$ is relatively small compared to T, the effect of overestimating $\|\boldsymbol{u}\|_2$ is minor, because Z_T depends only logarithmically on σ. In the second result of the Theorem, however, which covers the high-dimensional setting, the effect of σ is similar to the usual dependence of Online Gradient Descent on a guess for $\|\boldsymbol{u}\|_2$ (Zinkevich, 2003; Shalev-Shwartz, 2012) and taking σ much larger affects the rate linearly. In both regimes, underestimating $\|\boldsymbol{u}\|_2$ when tuning σ may degrade performance.

6.3.2 AN UNDESIRABLE REPARAMETRIZATION

If underestimating $\|\boldsymbol{u}\|_2$ is a major concern, then it is possible to reparametrize in terms of a new tuning parameter $\alpha > 0$ by setting $\sigma = \frac{1}{\sqrt{\alpha\eta}}$, as done by Luo et al. (2017). This means that each η-expert is now using a different choice for σ, but all our results up to Theorem 6 still go through. Optimizing η then leads to the following variant of Theorem 7, proved in Appendix C:

Theorem 9 (The Road Not Taken) *Suppose we tune each η-expert in MetaGrad Full with $\sigma = 1/\sqrt{\alpha\eta}$ for a given $\alpha > 0$. Then for all $\boldsymbol{u} \in \bigcap_{t=1}^T \mathcal{W}_t$ its linearized regret is simultaneously bounded by*

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \frac{5}{2}\sqrt{V_T^{\boldsymbol{u}} Z_T} + 5B_T Z_T + \frac{\alpha}{2}\|\boldsymbol{u}\|_2^2 + 2B_T,$$

where $Z_T = \mathrm{rk}(\boldsymbol{F}_T)\ln(1 + \frac{\sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2}{B_T \alpha\, \mathrm{rk}(\boldsymbol{F}_T)}) + 2\ln\lceil 2\log_2 T\rceil_+ + \frac{1}{2}$, and by

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \frac{5}{2}\sqrt{V_T^{\boldsymbol{u}} Z'_T} + 5B_T Z'_T + \frac{2}{\alpha}\sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2 + \frac{\alpha}{2}\|\boldsymbol{u}\|_2^2 + 2B_T,$$

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

This result is of a similar flavor as Theorem 7 if we set $\alpha = 1/\|\boldsymbol{u}\|_2^2$ in the first inequality and $\alpha = 2\sqrt{\sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2}/\|\boldsymbol{u}\|_2$ in the second inequality. A potential gain is that tuning α may be easier in case of the first inequality: the term $\frac{\alpha}{2}\|\boldsymbol{u}\|_2^2$ may not be dominant even if our choice of α is significantly off from the optimal tuning. But we pay significantly for this convenience, because there no longer exists a single choice for α that works both for the first and the second inequality simultaneously, which is why we do not advocate this reparametrization in terms of α.

6.3.3 THE COMPUTATIONALLY EFFICIENT DOMAIN FROM SECTION 5.1

A further important case to consider is when $\mathcal{W}_t$ is the computationally efficient domain from (13), for which the diameter is not bounded. This domain presumes that losses take the form $f_t(\boldsymbol{w}) = h_t(\boldsymbol{w}^\intercal \boldsymbol{x}_t)$ for a convex function h_t. Under the Lipschitz assumption that $|h_t'(z)| \leq L$ for all $|z| \leq C$, Luo et al. (2017) show a lower bound of $\Theta(\sqrt{dT})$ on the worst-case regret. They further obtain a (nearly) matching upper bound of

$$R_T^{\boldsymbol{u}} = O\left(\sqrt{dT} \ln \frac{\sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2}{\alpha} + \alpha \|\boldsymbol{u}\|_2^2\right) \qquad \text{for all } \boldsymbol{u} \in \bigcap_{t=1}^T \mathcal{W}_t$$

with a variant of ONS, where $\alpha > 0$ is a tuning parameter similar to the α in Theorem 9. The first results of Theorems 7 and 9 improve on this in that they improve the dependence on T to $V_T^{\boldsymbol{u}} \leq L^2 C^2 T$, they only depend on the effective dimension via $\mathrm{rk}(\boldsymbol{F}_T) \leq d$ and the logarithmic factor is moved inside the square root.

7. Analysis of the Faster Extension Algorithms

In this section analyse the sketched and coordinate-wise versions of MetaGrad.

7.1 Sketching: Analysis

We will refer to the Frequent Directions sketching version of MetaGrad as MetaGrad Sketch. Its analysis 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 $\boldsymbol{F}_T = \sum_{t=1}^T \boldsymbol{g}_t \boldsymbol{g}_t^\intercal$. Let λ_i be the i-th eigenvalue of $\boldsymbol{F}_T$ 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 10 *Consider the sketching version of the MetaGrad η-expert algorithm with sketch size parameter m, learning rate $\eta \leq \frac{1}{2B_T}$, and 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*

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

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

Compared to Lemma 5, we see that $\sum_{t=a^\eta}^T \boldsymbol{g}_t \boldsymbol{g}_t^\intercal = (\boldsymbol{G}_T^\eta)^\intercal \boldsymbol{G}_T^\eta$ in the logarithmic term has been replaced by its sketching approximation $(\boldsymbol{S}_T^\eta)^\intercal \boldsymbol{S}_T^\eta$. 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 10 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 three minor remarks: the first is that Deswarte imposes a slightly stricter upper bound on η, which allows him to bound $\frac{1}{2}\left(1 + 2\eta \langle \boldsymbol{w}_t - \boldsymbol{w}_t^\eta, \boldsymbol{g}_t \rangle\right)^2 \leq 1$, whereas we get an upper bound of 2 from (18) and therefore obtain a final result that is a factor of 2 larger. The second remark is that our η-expert algorithm is started in round a^η instead of round 1, leading to a bound involving $\Omega_q^\eta = \sum_{i=q+1}^d \lambda_i^\eta$, where λ_i^η is the i-th eigenvalue of $\sum_{t=a^\eta}^T \boldsymbol{g}_t \boldsymbol{g}_t^\intercal$. For simplicity, we immediately use Weyl's inequality to bound $\Omega_q^\eta \leq \Omega_q$, because the difference is minor. Finally, 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 6, we find:

Theorem 11 (Sketching Grid Point Regret) *Let $\eta \in \mathcal{G}$ be such that $\eta \leq \frac{1}{2B_T}$. Then MetaGrad Sketch with sketch size parameter m guarantees that the linearized regret w.r.t. any comparator $\boldsymbol{u} \in \bigcap_{t=1}^T \mathcal{W}_t$ is at most*

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

for any $q = 0, \ldots, m-1$. Recall that Q_T is defined in (19).

As shown in Appendix C, optimizing η and bounding $(\boldsymbol{S}_T^\eta)^\intercal \boldsymbol{S}_T^\eta$ appropriately leads to the following final result:

Theorem 12 (MetaGrad Sketching Regret Bound) *For all $\boldsymbol{u} \in \bigcap_{t=1}^T \mathcal{W}_t$ the linearized regret of MetaGrad Sketch with sketch size parameter m is simultaneously bounded by*

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

where $Z_T = 2m\ln\left(1 + \frac{\sigma^2\sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2}{4B_T^2 m}\right) + 2\ln\lceil 2\log_2 T\rceil_+ + \frac{1}{2} = O(m\ln T)$, and by

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

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

Compared to Theorem 7, we see the additional term involving Ω_q, which corresponds to the directions not captured by the sketch. We also see that $\mathrm{rk}(\boldsymbol{F}_T) \le d$ got replaced by $2m$ in the definition of Z_T. This comes from the analogous upper bound $\mathrm{rk}((\boldsymbol{S}_T^\eta)^\intercal \boldsymbol{S}_T^\eta) \le 2m$.

7.2 Coordinate MetaGrad: Analysis

The analysis of the coordinate version of MetaGrad, which we call MetaGrad Coord, is straightforward as we can simply apply the regret bound of MetaGrad Full to each dimension and add up the bounds:

Theorem 13 *Let* $V_{T,i}^{u_i} = \sum_{t=1}^T (u_i - w_{t,i})^2 g_{t,i}^2$*. For any* $\boldsymbol{u} \in \bigcap_{t=1}^T \mathcal{W}_t^{\text{rect}}$*, the linearized regret of MetaGrad Coord is simultaneously bounded by*

$$\tilde{R}_T^{\boldsymbol{u}} \le \sum_{i=1}^d \Big\{ \frac{5}{2}\sqrt{V_{T,i}^{u_i}(\tfrac{1}{2\sigma^2}u_i^2 + Z_{T,i})} + 5B_{T,i}(\tfrac{1}{2\sigma^2}u_i^2 + Z_{T,i}) + 2B_{T,i} \Big\}, \tag{22}$$

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

$$\tilde{R}_T^{\boldsymbol{u}} \;\le\; \sum_{i=1}^d \Big\{ \frac{5}{2}\sqrt{\Big(V_{T,i}^{u_i} + 2\sigma^2 \sum_{t=1}^T g_{t,i}^2\Big)\Big(\tfrac{1}{2\sigma^2}u_i^2 + Z'_T\Big)} + 5B_{T,i}\Big(\tfrac{1}{2\sigma^2}u_i^2 + Z'_T\Big) + 2B_{T,i} \Big\}, \tag{23}$$

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

The recommended tuning is to set σ to (an upper bound on) $\|\boldsymbol{u}\|_\infty$. For this case, we obtain the following corollary, which is proved in Appendix D:

Corollary 14 *Suppose the domain is a fixed rectangle:* $\mathcal{W}_t = \mathcal{W}^{\text{rect}}$*, and we tune* $\sigma = D_\infty := \max_{\boldsymbol{u}\in\mathcal{W}^{\text{rect}}} \|\boldsymbol{u}\|_\infty$ *based on the size of the domain. Let* $g_{1:T,i} := (g_{i,1}, \dots, g_{i,T})$*. Then the linearized regret of MetaGrad Coord with respect to any* $\boldsymbol{u} \in \mathcal{W}^{\text{rect}}$ *is bounded by*

$$\tilde{R}_T^{\boldsymbol{u}} \;=\; O\left(\sum_{i=1}^d \sqrt{V_{T,i}^{u_i}\ln(D_\infty G_\infty T)} + D_\infty G_\infty d \ln(D_\infty G_\infty T)\right), \tag{24}$$

provided that $\|\boldsymbol{g}_t\|_\infty \le G_\infty$ *for all* t*, and it is simultaneously bounded by*

$$\tilde{R}_T^{\boldsymbol{u}} \;=\; O\left(D_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2 \sqrt{\ln\ln T} + D_\infty G_2 \sqrt{d}\ln\ln T\right) \;=\; O\left(D_\infty G_2 \sqrt{dT\ln\ln T}\right), \tag{25}$$

provided that $\|\boldsymbol{g}_t\|_2 \le G_2$ *for all* t.

The first result of the corollary, (24), is sufficient to obtain fast rates under a coordinate version of the Bernstein condition, which is discussed below. The second result, (25), shows that we simultaneously recover the regret bound of order $\tilde{O}\left(\sum_{i=1}^d D_\infty \sum_{i=1}^d \|g_{1:T,i}\|_2\right)$ that is the main feature of the diagonal version of AdaGrad. As pointed out by Duchi et al. (2011), the norms $\|g_{1:T,i}\|_2$ can be significantly smaller than T when the gradients are sparse, and the dependence on D_∞ is appropriate when the optimal parameters $\boldsymbol{u}$ form a dense vector. When the gradients are not sparse the bound degrades to $\tilde{O}\left(D_\infty\sqrt{dT}\right)$, which is optimal over rectangular domains under an ℓ_2 or even ℓ_1 bound on the gradients: if we encounter T/d axis-aligned gradients per dimension, then each dimension can contribute $\Omega(\sqrt{T/d})$ to the regret, which gives $\Omega(\sqrt{dT})$ regret in total.

7.2.1 OPEN PROBLEM: RESTRICTED DOMAINS

Most online learning methods can deal with arbitrary convex domains using projections, but we have presented Coordinate MetaGrad only for rectangular domains. Can it be extended to other domains, preferably without incurring any significant computational overhead? One approach we have tried is to apply the black-box reduction of Cutkosky and Orabona (2018), which would run MetaGrad Coord with fake gradients $\tilde{\boldsymbol{g}}_t$ to obtain iterates $\tilde{\boldsymbol{w}}_t$ from a rectangular domain, which are then projected onto the true domain $\mathcal{W}_t$ to obtain final iterates $\boldsymbol{w}_t$. Formally, this reduction goes through, but it leads to a regret bound in which the terms $V_{T,i}^{u_i}$ are replaced by unsatisfactory surrogates $\tilde{V}_{T,i}^{u_i}$ that are measured in terms of the fake gradients $\tilde{g}_{t,i}$ and the wrong, unprojected parameters $\tilde{w}_{t,i}$. This can partially be remedied, because the reduction guarantees that

$$\|\tilde{\boldsymbol{g}}_t\|_* \leq \|\boldsymbol{g}_t\|_*, \tag{26}$$

where $\|\cdot\|_*$ is the dual norm of the norm $\|\cdot\|$ that is used to project $\tilde{\boldsymbol{w}}_t$ onto the domain. If $f_t(\boldsymbol{w}) = h_t(\boldsymbol{w}^\intercal \boldsymbol{x}_t)$ for a convex function h_t and $\boldsymbol{x}_t \in \mathbb{R}^d$ is available before we choose $\boldsymbol{w}_t$, then $\nabla f_t(\boldsymbol{w}) = h_t'(\boldsymbol{w}^\intercal \boldsymbol{x}_t)\boldsymbol{x}_t$ and we can project with the norm $\|\boldsymbol{w}\|_{\boldsymbol{x}_t} = \sum_{i=1}^d |x_{t,i}||w_i|$, which leads to the dual norm $\|\boldsymbol{g}\|_{\boldsymbol{x}_t,*} = \max_i \frac{|g_i|}{|x_{t,i}|}$. Plugging this into (26) and simplifying, we then find that

$$|\tilde{g}_{t,i}| \leq |g_{t,i}| \qquad \text{for } i = 1, \ldots, d,$$

which implies that

$$\tilde{V}_{T,i}^{u_i} = \sum_{t=1}^T (u_i - \tilde{w}_{t,i})^2 \tilde{g}_{t,i}^2 \leq \sum_{t=1}^T (u_i - \tilde{w}_{t,i})^2 g_{t,i}^2.$$

We can thus get rid of the dependence on the fake gradients, but the dependence on the wrong iterates $\tilde{\boldsymbol{w}}_t$ remains, so in the end the black-box reduction only gets us half of the way. This is unsatisfying, because the wrong iterates $\tilde{\boldsymbol{w}}_t$ do not lead to fast rates under the coordinate Bernstein condition described below. It is an open problem whether there exists another (computationally efficient) approach that fully preserves the original regret bounds from Corollary 14 for non-rectangular domains, and therefore does achieve these fast rates. In light of this open problem, it is interesting to remark that the black-box reduction can be made to work for MetaGrad Full, as exploited by Mhammedi et al. (2019, Theorem 10).

7.2.2 COORDINATE BERNSTEIN CONDITION

Since the coordinate version of MetaGrad does not keep track of a full covariance matrix Σ_T, we cannot expect it to exploit the Bernstein condition in all cases. An appropriate modification is the following *coordinate* (B, β)*-Bernstein condition*:

$$\sum_{i=1}^{d}(w_i - u_i^*)^2 \mathop{\mathbb{E}}_{f}\left[[\nabla f(\boldsymbol{w})]_i^2\right] \;\leq\; B\Big((\boldsymbol{w}-\boldsymbol{u}^*)^\intercal \mathop{\mathbb{E}}_{f}\left[\nabla f(\boldsymbol{w})\right]\Big)^{\beta} \qquad \text{for all } \boldsymbol{w} \in \mathcal{W}, \tag{27}$$

where we have again assumed that the domain $\mathcal{W}_t = \mathcal{W}$ does not vary between rounds, and that the losses f_t are independent, identically distributed. The following theorem, proved in Appendix A, is analogous to Theorem 2: it shows that, under the coordinate Bernstein condition, the coordinate version of MetaGrad achieves fast rates:

Theorem 15 *If the gradients satisfy the coordinate* (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* (24) *incurs expected regret*

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

So when can we expect the coordinate Bernstein condition to hold? If the covariances between the coordinates of the gradients are zero, then the ordinary Bernstein condition reduces to the coordinate Bernstein condition with the same B and β, but this is a very strong assumption that seems of limited practical use. The following result captures how this assumption may be significantly relaxed while still obtaining the same dependence on β, albeit at the cost of a worse constant B. It considers the case that the losses are of the form $f_t(\boldsymbol{w}) = h_t(\boldsymbol{w}^\intercal \boldsymbol{x}_t)$ for a convex function h_t, with $(\boldsymbol{x}_1, h_1), \ldots, (\boldsymbol{x}_T, h_T)$ independent, identically distributed. Let $h_t'(z)$ denote the (sub)derivative of h_t at z.

Theorem 16 *Suppose that* $0 < L_- \leq |h_t'(\boldsymbol{w}^\intercal \boldsymbol{x})| \leq L_+$ *for all* $\boldsymbol{w} \in \mathcal{W}$ *and that* $\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\intercal] \succeq C \operatorname{diag}(\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\intercal])$ *for some* $C > 0$*. Then*

$$\mathop{\mathbb{E}}_{f}\left[\nabla f(\boldsymbol{w})\nabla f(\boldsymbol{w})^\intercal\right] \succeq \frac{C L_-^2}{L_+^2} \mathop{\mathbb{E}}_{f}\left[\operatorname{diag}([\nabla f(\boldsymbol{w})]_1^2, \ldots, [\nabla f(\boldsymbol{w})]_d^2)\right]$$

for all $\boldsymbol{w} \in \mathcal{W}$*, and consequently the* (B, β)*-Bernstein condition* (7) *implies the coordinate* (B', β)*-Bernstein condition* (27) *with a constant* $B' = \frac{L_+^2}{C L_-^2} B$ *instead of* B*.*

The condition $\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\intercal] \succeq C \operatorname{diag}(\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\intercal])$ expresses that the (uncentered) covariances between features should be weak.[2] In particular, it is satisfied with $C = 1$ if all pairs

[2] The highest $C \geq 0$ satisfying the condition is given by the smallest eigenvalue of the (uncentered) correlation matrix, i.e. $\lambda_{\min}\left(\operatorname{diag}(\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\intercal])^{-1/2}\, \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\intercal] \operatorname{diag}(\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\intercal])^{-1/2}\right)$.

of features have covariance zero. The conditions on h_t are satisfied by the logistic loss $h_t(z) = \ln(1 + e^{-y_t z})$ when the margins $y_t \boldsymbol{w}^\intercal \boldsymbol{x}_t$ are uniformly bounded. For the hinge loss $h_t(z) = \max\{1 - y_t z, 0\}$ they are also satisfied if the margins are strictly less than 1, but we get $L_- = 0$ for $y_t \boldsymbol{w}^\intercal \boldsymbol{x}_t > 1$.

Proof (Theorem 16) The main inequality is established as follows:

$$\begin{aligned}\mathop{\mathbb{E}}_f\left[\nabla f(\boldsymbol{w})\nabla f(\boldsymbol{w})^\intercal\right] &= \mathop{\mathbb{E}}_f\left[h_t'(\boldsymbol{w}^\intercal\boldsymbol{x})^2\boldsymbol{x}\boldsymbol{x}^\intercal\right] \succeq \mathop{\mathbb{E}}_f\left[L_-^2\boldsymbol{x}\boldsymbol{x}^\intercal\right] \succeq CL_-^2 \operatorname{diag}(\mathop{\mathbb{E}}_f\left[\boldsymbol{x}\boldsymbol{x}^\intercal\right]) \\ &\succeq \frac{CL_-^2}{L_+^2}\operatorname{diag}(\mathop{\mathbb{E}}_f\left[h_t'(\boldsymbol{w}^\intercal\boldsymbol{x})^2\boldsymbol{x}\boldsymbol{x}^\intercal\right]) = \frac{CL_-^2}{L_+^2}\mathop{\mathbb{E}}_f\left[\operatorname{diag}([\nabla f(\boldsymbol{w})]_1^2, \ldots, [\nabla f(\boldsymbol{w})]_d^2)\right].\end{aligned}$$

Multiplying both sides of the inequality by $(\boldsymbol{w} - \boldsymbol{u}^*)^\intercal$ on the left and $(\boldsymbol{w} - \boldsymbol{u}^*)$ on the right, we see that the left-hand side of (7) dominates the left-hand side of (27) up to a factor of $\frac{CL_-^2}{L_+^2}$, which is responsible for the difference between B and B'. ■

8. Experiments

The goal of this experiments section is to quantify the performance of the proposed MetaGrad variants in comparison with existing algorithms for Online Convex Optimization. The corresponding Python code is available from GitHub (van Erven et al., 2021). We set things up as follows.

8.1 Setup

We describe the data we used, the specific prediction task we considered, and the algorithms we evaluated. We also discuss the choice of domain and hyper-parameters.

8.1.1 DATA

We evaluate on real-world regression and binary classification data sets from the standard LIBSVM library (Chang and Lin, 2011). A summary of the data sets can be found in Table 2 in Appendix E. We have included data sets of dimension up to 300, so that MetaGrad Full is tractable. We exclude the `mushrooms` data set, because it is linearly separable. This makes the best offline parameters $\boldsymbol{u}^*$ non-unique for the hinge loss and have infinite norm for the logistic loss, which is incompatible with our parameter tuning below. The resulting seventeen data sets have sample sizes T ranging from 252 to 581 012 and dimensions d ranging from 6 to 300. When available we used the normalised version of the data set with features in $[-1, 1]$.

8.1.2 TASK

Each data set is a sequence of labelled examples $(\boldsymbol{x}_t, y_t)$ for $t = 1, \ldots, T$. We define our task to be sequential prediction of the labels y_t from the features $\boldsymbol{x}_t$ using a linear model

$\hat{y}_t = \boldsymbol{w}_t^\intercal \boldsymbol{x}_t$. We use a linear model with intercept, which we implement by appending a constant 1 to each feature vector. For the classification data with $y_t \in \{-1, 1\}$, we consider both the hinge loss $f_t(\boldsymbol{w}) = \max\{0, 1 - y_t \boldsymbol{w}^\intercal \boldsymbol{x}_t\}$ and the logistic loss $f_t(\boldsymbol{w}) = \ln\left(1 + e^{-y_t \boldsymbol{w}^\intercal \boldsymbol{x}_t}\right)$. For the regression data with $y_t \in \mathbb{R}$, we consider the absolute loss $f_t(\boldsymbol{w}) = |y_t - \boldsymbol{w}^\intercal \boldsymbol{x}_t|$ and the squared loss $f_t(\boldsymbol{w}) = (y_t - \boldsymbol{w}^\intercal \boldsymbol{x}_t)^2$.

8.1.3 METHODS

We compare 9 methods: two popular versions of Online Gradient Descent, the diagonal version of AdaGrad, and six versions of MetaGrad. We include the Online Gradient Descent scheme $\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w} \in \mathcal{W}} \boldsymbol{w}^\intercal \boldsymbol{g}_t + \frac{1}{2\eta_t} \|\boldsymbol{w} - \boldsymbol{w}_t\|^2$ with time-decreasing learning rate $\eta_t = \frac{\sigma}{\sqrt{t} \max_{s \leq t} \|\boldsymbol{g}_s\|_2}$ (abbreviated as OGDt) and with the gradient-norm-adaptive learning rate $\eta_t = \frac{\sigma}{\sqrt{\sum_{s=1}^t \|\boldsymbol{g}_s\|_2^2}}$ (abbreviated as OGDnorm). In both cases, σ is a hyperparameter. Diagonal AdaGrad can be viewed as OGDnorm applied to each coordinate separately, and predicts with iterates $\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w} \in \mathcal{W}} \boldsymbol{w}^\intercal \boldsymbol{g}_t + \sum_i \frac{1}{2\eta_{t,i}} (w_i - w_{t,i})^2$ with separate learning rates per dimension $\eta_{t,i} = \frac{\sigma}{\sqrt{\sum_{s=1}^t g_{s,i}^2}}$. Note that for both AdaGrad and Gradient Descent we use the standard mirror descent version, as opposed to the FTRL/primal-dual version. The six versions of MetaGrad are the full version presented in Section 4 (abbreviated as MGFull), the coordinate version presented in Section 5.2 (abbreviated as MGCo), and the Frequent Directions sketching version presented in Section 5.1 for $m = 2$, $m = \min\{11, d + 1\}$, $m = \min\{26, d + 1\}$, and $m = \min\{51, d + 1\}$ (abbreviated as MGFm). Note that in each case the number of directions maintained is $m - 1$, so this corresponds to effective dimensions $1, 10, 25$ and 50, or d when d is smaller.

8.1.4 DOMAIN

Each of our algorithms requires a choice of domain $\mathcal{W}$. While algorithm and domain are independent in principle, in practice computational convenience is paramount, and only the convenient default domain choice is prevalent for each algorithm. In this sense one may think of the choice of algorithm as importing (additional) regularisation through its associated domain. For the two versions of Gradient Descent we use the ℓ_2-norm ball $\mathcal{W} = \{\boldsymbol{w} : \|\boldsymbol{w}\|_2 \leq D_2\}$. For the two diagonal algorithms, AdaGrad and MGCo, we use the ℓ_∞-norm ball $\mathcal{W} = \{\boldsymbol{w} : \|\boldsymbol{w}\|_\infty \leq D_\infty\}$. For the other versions of MetaGrad we use the time-varying domain $\mathcal{W}_t = \{\boldsymbol{w} : |\boldsymbol{w}^\intercal \boldsymbol{x}_t| \leq C\}$ from (13), recalling its major benefit that projections on $\mathcal{W}_t$ can be computed efficiently (see the discussion in Section 5.1). Having fixed the domain shape, we still have to fix the domain sizes D_2, D_∞ and C. We note that this is part of the art of employing machine learning in practice. To simulate the availability of weak prior knowledge about the appropriate domain bound, we first compute the unconstrained optimizer of the cumulative loss $\boldsymbol{u}^* = \arg\min_{\boldsymbol{u} \in \mathbb{R}^d} \sum_{t=1}^T f_t(\boldsymbol{u})$. We then set the size bounds to fit the comparator up to the small factor 3, i.e. setting $D_2 = 3\|\boldsymbol{u}^*\|_2$, $D_\infty = 3\|\boldsymbol{u}^*\|_\infty$ and $C = 3 \max_t |\boldsymbol{x}_t^\intercal \boldsymbol{u}^*|$. We overprovision the domain by the factor $3 > 1$

Algorithm	# best	# better than OGDt	MedianRatio
AdaGrad	0	0	3.54
OGDnorm	0	4	1.41
OGDt	1	34	1.00
MGCo	12	33	0.32
MGF2	2	31	0.31
MGF11	14	31	0.27
MGF26	15	33	0.27
MGF51	17	33	0.25
MGFull	21	33	0.25

Table 1: Comparison of algorithms with OGDt. The MedianRatio column contains the median ratio of the regret of each algorithm over that of OGDt. Columns "# best" and "# better than OGDt" count cases where the algorithm is at most one regret unit above the best algorithm or OGDt, respectively.

to prevent possibly beneficiary effects that may kick in when the comparator lies on the boundary of the domain (Huang et al., 2016).

8.1.5 Hyperparameter Tuning

We now discuss tuning the hyperparameter σ that is present in all methods. To keep the playing field level and convey the same tuning advantage to all algorithms, we provide the optimal theoretical tuning of σ for all methods, even though $\|\boldsymbol{u}^*\|$ is unknown in practice. Theoretical guidance on this optimal setting comes in two types, depending on the makeup of the algorithm, and in particular control of a telescoping term in its regret bound (see e.g. Zinkevich 2003 or Duchi et al. 2011, Corollary 11). For all versions of MetaGrad, the optimal setting $\sigma = \|\boldsymbol{u}^*\|$ is the norm of the comparator itself (see Sections 4.3 and 5.2.1). For algorithms in the mirror descent family (including OGD and AdaGrad), the optimal theoretical setting is $\sigma = \max_{w \in \mathcal{W}} \|\boldsymbol{w} - \boldsymbol{u}^*\|/\sqrt{2}$ (see e.g. Duchi et al. 2011, Corollary 11, Orabona and Pál 2018, Theorem 2). For our norm-ball domain of radius $3\|\boldsymbol{u}^*\|$ this yields $\sigma = \sqrt{8}\|\boldsymbol{u}^*\|$. We also study the effect of tuning σ with hindsight in Appendix E.1.

8.2 Experimental Results

Table 3 in Appendix E lists the regrets of all 9 algorithms on all 17 data sets measured in 2 loss functions each.

8.2.1 Best Overall Algorithm

We first try to answer the question which algorithm is best. To this end, we present a summary of the results in Table 1. The table shows how often each algorithm achieves the lowest loss on our 17 data sets as measured in either of the two relevant loss functions (for

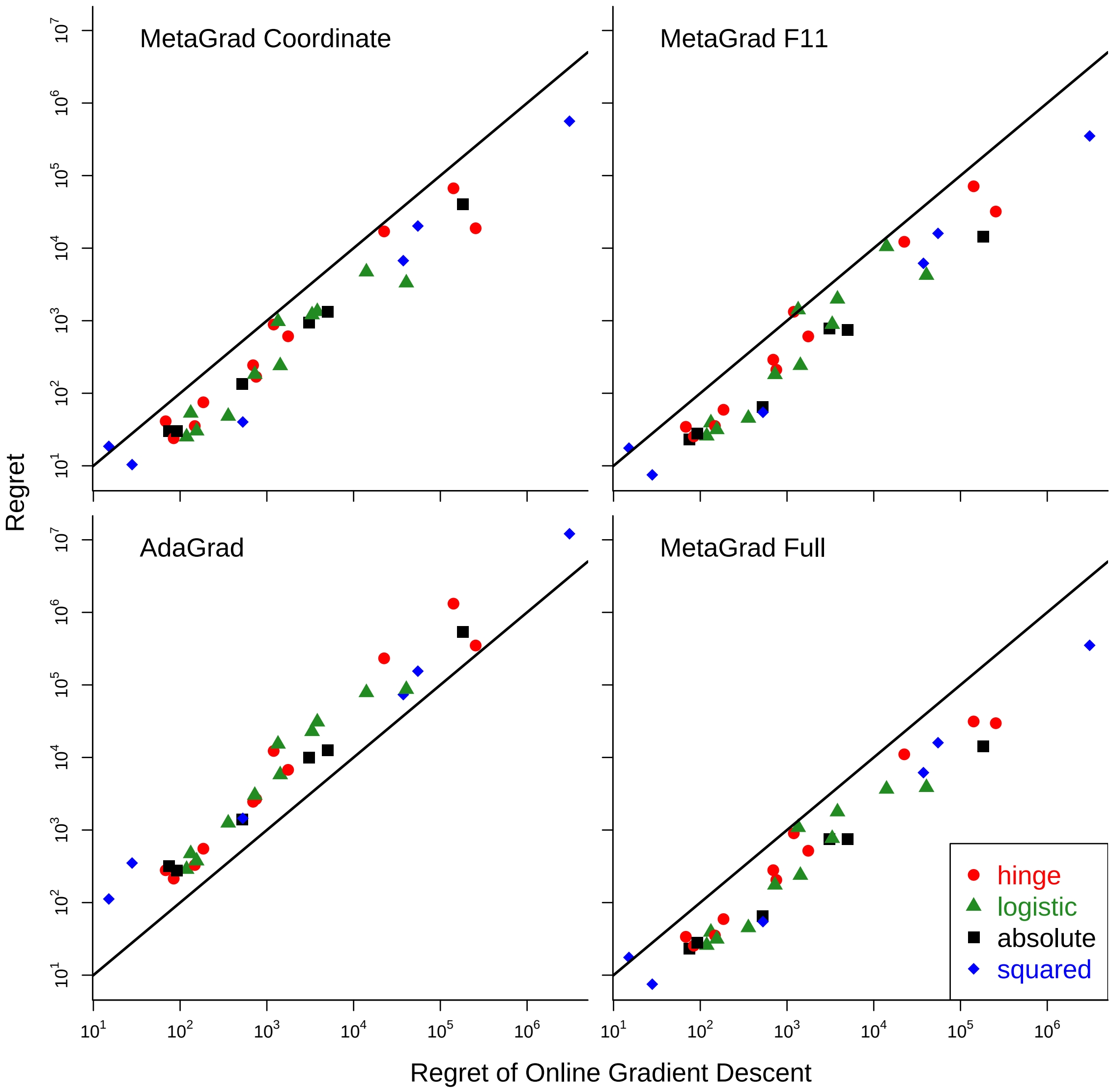


Figure 1: Comparison of the logarithm of the regret of three versions of MetaGrad and AdaGrad and the logarithm of the regret of OGDt. Each marker represents a combination of data set and loss function.

some data sets ties result in several algorithms being counted as best). It further lists how often each algorithm beats OGDt, and what the median ratio of regrets is between each algorithm and OGDt. We choose OGDt as the baseline, as it is empirically best among our competitors AdaGrad, OGDnorm and OGDt. We see that versions of MetaGrad, especially MetaGrad Full, are often the best algorithm overall (there is only 1 of 34 cases where no MetaGrad version is the winner). Moreover, the experiments corroborate the intuition that the performance of the MetaGrad sketching versions improves as more dimensions are retained. The table suggests the recommendation to use sketched MetaGrad in practice, retaining as many dimensions as are computationally affordable.

8.2.2 PREDICTING THE WINNER

The next question is if we can observe any patterns in the circumstances for which each algorithm shines. To gain insight here, we compare in Figure 1 the regrets of MGCo, MGF11, AdaGrad and MGFull each with OGDt. There does not seem to be a visually discernible pattern allowing us to predict the advantage over OGDt (vertical position) from the OGDt regret on the data set (horizontal position) or the loss function (color). We looked into the blue diamond (i.e. square loss) at the far left in each plot, on which OGDt dominates. This data set is `mg`. One feature that stands out for this set is that its optimal unconstrained coefficient vector assigns a large coefficient to the constant 1 feature we added to implement the intercept, while using much smaller (three orders of magnitude) coefficients for all other features. This results in an essentially sparse optimal weight vector. It is not clear to us how OGDt is especially able to exploit this. For sure its regret bound does not give a hint.

Comparing MetaGrad Full to MGF11 in Figure 1, we see that most data set markers are very similarly positioned, except for two (one green, one red) that moved slightly upwards. This is the `covtype` data set. For this set we indeed see the loss rise dramatically when we sketch to smaller sizes or use the coordinatewise approximation. Apparently, for this data set all features are important, and the intrinsic orientation is not the coordinate basis. Overall, the coordinatewise version of MetaGrad is close to the performance of the Full version of MetaGrad (the median regret ratio is 1.09), which suggests that on most data sets the correlations between the features are of little importance.

8.2.3 SURPRISES

To our surprise, AdaGrad has the worst performance of all algorithms. Upon closer review of the literature we observe that the hyperparameter (σ) of AdaGrad is often optimized in hindsight based on the data. In Appendix E.1, we tested by how much we could improve the performance of all methods by tuning σ in hindsight. We find there that the benefits are especially large for AdaGrad. For instance, on `w8a` with the logistic loss, we find that we can tune AdaGrad such that the regret improves from 86921 to 1147, improving upon the theoretically recommended tuning of AdaGrad by an astounding factor 76, as well as beating the best theoretically tuned algorithm (MGCo) by a factor 2.9. See Appendix E.1 for a further study of this phenomenon. However, such post-hoc tuning is not available in sequential decision-making applications. We take this as a clear motivation to study the effects of tuning, and develop algorithms that tune themselves.

A second surprise is that OGDt beats OGDnorm (median regret ratio 1.41). Worst-case regret bounds indicate that the reverse should occur. There exist tighter "luckiness" analyses for stochastic cases in the literature (Gaillard et al., 2014; Koolen et al., 2016; Mourtada and Gaïffas, 2019), but these are not sharp enough to explain the difference between OGDnorm and OGDt. Moreover, these analyses require conditions for which it seems implausible that a large majority of data sets should satisfy them to the same degree, so these analyses cannot explain why the dominance of OGDt is so consistent.

Conclusion Overall, we see that MetaGrad outperforms AdaGrad and Online Gradient Descent consistently across a range of real-world data sets. We conclude that MetaGrad is the best choice for sequential scenarios where safety requirements dictate tuning for theoretical guarantees, as we have studied here.

8.3 Additional Experiments with Hypertuning

In Appendix E.1 we provide additional experiments to investigate the best performance that can be achieved in principle by each of the 9 algorithms, by tuning the hyperparameter σ in hindsight for each data set. For all methods, σ directly influences the effective learning rate η, so this hypertuning provides a loophole to optimize η for the data. Although such post-hoc tuning is impossible in a fully online setting, these experiments give insight into whether the theoretical tunings are good advice in practice for non-adversarial data. In general, all methods gain from hypertuning, but some more than others. A striking difference with the previous experiments is that AdaGrad, OGDnorm and MetaGrad all achieve very similar performance when they use hypertuning. It therefore seems to matter more if the hyperparameters are optimally tuned to the data set, than which algorithm is chosen. This suggests that the empirical superiority of MetaGrad in the primary experiments may be attributed to its ability to better adapt to the optimal learning rate η.

9. Conclusion and Possible Extensions

We provide a new adaptive method, MetaGrad, which is robust to general convex losses but simultaneously can take advantage of special structure in the losses, like curvature in the loss function 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 (8) 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 only the log of the number of experts, and since the number of experts is $O(\ln T)$ this leads to an additional $O(\ln \ln T)$ term that is typically dominated by other terms in the bound. It is therefore also 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 (8) by an upper bound. This possibility is exploited by Wang et al. (2020), who add extra experts with surrogates that contain $\big(\eta G \|\boldsymbol{u} - \boldsymbol{w}_t\|_2\big)^2$ instead, where G is a known upper bound on $\|\boldsymbol{g}_t\|$.[3] 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 adaptive extension of MetaGrad that also gets the optimal rate $O(\ln T)$ for strongly convex losses, without any dependence on d. The price of this extension is that it doubles the number of experts, which adds a negligible constant $\ln 2$ to the regret, but doubles the run-time of the algorithm.

If we are willing to increase the number of experts, then another appealing extension would be to adapt to the σ hyperparameter. This is possible by adding multiple copies of each η-expert with different values for σ. If σ ranges over a set of candidate values $\mathcal{S} := \{\sigma_1, \ldots, \sigma_p\}$, then our overhead compared to the best choice of σ from $\mathcal{S}$ is an additional small constant $\ln p$ in the regret, but our run-time multiplies by p, so we would still need to keep p relatively small. For example, we might take an exponentially spaced grid $\mathcal{S} := \{2^i \in [\sigma_{\text{min}}, \sigma_{\text{max}}] \mid i \in \mathbb{Z}\}$, so that $p \leq \lceil \log_2 \sigma_{\text{max}}/\sigma_{\text{min}} \rceil$.

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., 2019). The same sliding window idea could be used to base the covariance matrix $\boldsymbol{\Sigma}_t^\eta$ in our Algorithm 2 only on the k most recent gradients. This has both computational advantages, because $\boldsymbol{\Sigma}_t^\eta$ 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.

Acknowledgments

The authors would like to thank Raphaël Deswarte and Zakaria Mhammedi for collaborations on earlier versions of MetaGrad (Deswarte, 2018; Mhammedi et al., 2019) and the anonymous reviewers for insightful comments on the manuscript. This research was supported by the Netherlands Organization for Scientific Research (NWO): grant numbers VI.Vidi.192.095 for Van Erven and TOP2EW.15.211 for Van der Hoeven.

Appendix A. Extra Material Related to Fast Rate Conditions

In this section we gather extra material related to the fast rate examples from Sections 3 and 7.2. We first provide simulations. Then we present the proofs of Theorems 1, 2 and

[3] To make a Lipschitz-adaptive version of their approach, we might replace the constant G by the quantity $\|\boldsymbol{g}_t\|$ that it upper bounds.

15. And finally we give an example in which the unregularized hinge loss satisfies the Bernstein condition.

A.1 Simulations: Logarithmic Regret without Curvature

We provide two simple simulation examples to illustrate the sufficient conditions for Theorems 1 and 2, and to show that the resulting 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 coincides with OGD with an adaptive tuning of the learning rate that is applicable to general convex functions. See the description of AdaGrad/OGDnorm in Section 8 for a full description.

In the first example, we consider offline convex optimization of the fixed function $f_t(w) \equiv f(w) = |w - \frac{1}{4}|$, which satisfies the directional derivative condition (6) because it is convex. In the second example, we look at stochastic optimization with convex functions $f_t(w) = |w - x_t|$, where the outcomes $x_t = \pm\frac{1}{2}$ are chosen i.i.d. with probabilities 0.4 and 0.6. These probabilities satisfy (7) with $\beta = 1$. Their values are by no means essential, as long we avoid the worst case where the probabilities are equal. In both examples, the domain is $\mathcal{W} = [-1, 1]$. We tune AdaGrad with hyperparameter $\sigma = \max_{w,u\in\mathcal{W}} |w - u|/\sqrt{2} = \sqrt{2}$ and MetaGrad with $\sigma = \max_{u\in\mathcal{W}} |u| = 1$.

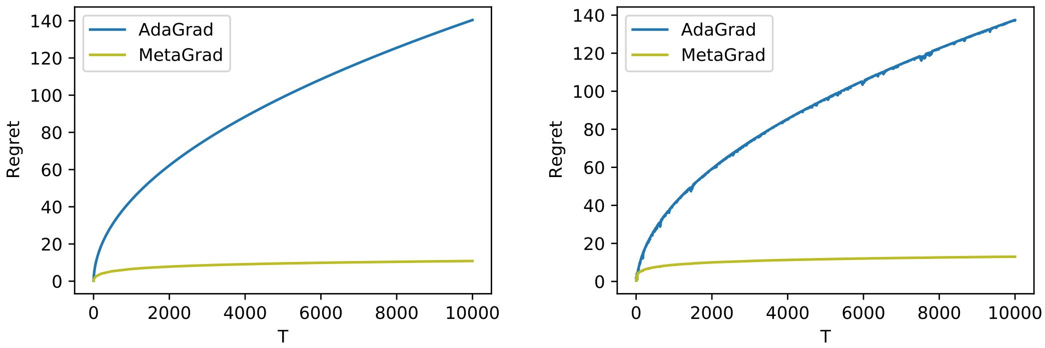
Figure 2 graphs the results. We see that in both cases the regret of AdaGrad follows its $O(\sqrt{T})$ bound, while MetaGrad achieves an $O(\ln T)$ rate, as predicted by Theorems 1 and 2. This shows that MetaGrad achieves a type of adaptivity that is not achieved by AdaGrad.

A.2 Proof of Theorem 1

Proof By (6), applied with $\boldsymbol{w} = \boldsymbol{w}_t$, and (2), 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 $\gamma = \frac{2b}{C}$. ■



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

A.3 Proofs of Theorems 2 and 15

Proof (Theorem 2) By (2) there exists a constant $C > 0$ such that, for all sufficiently large T,

$$\mathbb{E}\left[\tilde{R}_T^{\boldsymbol{u}^*}\right] \le C\,\mathbb{E}\left[\sqrt{V_T^{\boldsymbol{u}^*}\, d\ln T}\right] + Cd\ln T.$$

Abbreviating $\tilde{r}_t^{\boldsymbol{u}} = (\boldsymbol{w}_t - \boldsymbol{u})^\intercal \boldsymbol{g}_t$, we see that $\tilde{R}_T^{\boldsymbol{u}^*} = \sum_{t=1}^T \tilde{r}_t^{\boldsymbol{u}^*}$, $V_T^{\boldsymbol{u}^*} = \sum_{t=1}^T (\tilde{r}_t^{\boldsymbol{u}^*})^2$ and the Bernstein condition with $\boldsymbol{w} = \boldsymbol{w}_t$ becomes

$$\mathbb{E}[(\tilde{r}_t^{\boldsymbol{u}^*})^2 \mid \boldsymbol{w}_t] \le B\,\mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*} \mid \boldsymbol{w}_t]^\beta.$$

Combining the above with multiple applications of Jensen's inequality, the expected linearized regret is at most

$$\begin{aligned}
\mathbb{E}\left[\tilde{R}_T^{\boldsymbol{u}^*}\right] &\le C\sqrt{\mathbb{E}\left[V_T^{\boldsymbol{u}^*}\right] d\ln T} + Cd\ln T \\
&\le C\sqrt{B\sum_{t=1}^T \mathop{\mathbb{E}}_{\boldsymbol{w}_t}\left[\left(\mathbb{E}\left[\tilde{r}_t^{\boldsymbol{u}^*}|\boldsymbol{w}_t\right]\right)^\beta\right] d\ln T + Cd\ln T} \\
&\le C\sqrt{B\sum_{t=1}^T \left(\mathbb{E}\left[\tilde{r}_t^{\boldsymbol{u}^*}\right]\right)^\beta d\ln T} + Cd\ln T. \qquad (28)
\end{aligned}$$

In the following, we will repeatedly use the fact that

$$x^\alpha y^{1-\alpha} = c_\alpha \inf_{\gamma > 0}\left(\frac{x}{\gamma} + \gamma^{\frac{\alpha}{1-\alpha}} y\right) \qquad \text{for any } x, y \ge 0 \text{ and } \alpha \in (0,1), \qquad (29)$$

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

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

If $\beta = 1$, then $\sum_{t=1}^T \big(\mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*}]\big)^\beta = \mathbb{E}[\tilde{R}_T^{\boldsymbol{u}^*}]$ and the result follows by taking $\gamma_1 = \frac{1}{2Cc_{1/2}}$. Alternatively, if $\beta < 1$, then we apply (29) a second time, with $\alpha = \beta$, $x = \mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*}]$ and $y = 1$, to find that, for any $\gamma_2 > 0$,

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

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

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

We may optimize over γ_2 by a third application of (29), now with $x = 4C^2c_{1/2}^2 c_\beta Bd\ln T$, $y = T$ and $\alpha = 1/(2-\beta)$, such that $\alpha/(1-\alpha) = 1/(1-\beta)$:

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

which completes the proof. ■

Proof (Theorem 15) We will show that (28) from the proof of Theorem 2 also holds under the conditions of Theorem 15. The rest of the proof then proceeds in the same way. To this end, we use that (24) implies the existence of a constant $C > 0$ such that, for all sufficiently large T,

$$\tilde{R}_T^{\boldsymbol{u}^*} \leq C\sum_{i=1}^d \sqrt{V_{T,i}^{u_i}\ln(T)} + Cd\ln(T).$$

Multiple applications of Jensen's inequality, together with the coordinate Bernstein condition, then imply that

$$\begin{aligned}
\mathbb{E}\left[\tilde{R}_T^{\boldsymbol{u}^*}\right] &\le C\,\mathbb{E}\left[\sum_{i=1}^d \sqrt{V_{T,i}^{u_i^*}\,\ln(T)}\right] + Cd\ln(T) = Cd\,\mathbb{E}\left[\sum_{i=1}^d \frac{1}{d}\sqrt{V_{T,i}^{u_i^*}\,\ln(T)}\right] + Cd\ln(T)\\
&\le Cd\sqrt{\mathbb{E}\left[\sum_{i=1}^d \frac{1}{d}\left(V_{T,i}^{u_i^*}\,\ln(T)\right)\right]} + Cd\ln(T)\\
&= C\sqrt{\sum_{t=1}^T \mathop{\mathbb{E}}_{\boldsymbol{w}_t}\left[\sum_{i=1}^d (w_{t,i}-u_i^*)^2\,\mathbb{E}[g_{t,i}^2 \mid \boldsymbol{w}_t]\right] d\ln(T)} + Cd\ln(T)\\
&\le C\sqrt{B\sum_{t=1}^T \mathop{\mathbb{E}}_{\boldsymbol{w}_t}\left[\left(\mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*} \mid \boldsymbol{w}_t]\right)^\beta\right] d\ln(T)} + Cd\ln(T)\\
&\le C\sqrt{B\sum_{t=1}^T \left(\mathbb{E}[\tilde{r}_t^{\boldsymbol{u}^*}]\right)^\beta\, d\ln(T)} + Cd\ln(T).
\end{aligned}$$

This establishes the same inequality as in (28), and the remainder of the proof is the same as for Theorem 2. ■

A.4 Unregularized Hinge Loss Example

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

Lemma 17 (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 $f_t(\boldsymbol{u}) = \max\{0, 1 - Y_t\boldsymbol{u}^\intercal \boldsymbol{X}_t\}$ 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 $\mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^\intercal\right]$ 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 $B \le \frac{c}{\sqrt{d}}$ for some absolute constant $c > 0$.

Thus the version of the Bernstein condition that implies an $O(d\ln T)$ 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.

For completeness, we repeat the proof of Lemma 17 from Koolen et al. (2016):

Proof (Lemma 17) Since, by assumption, $\boldsymbol{u}$ and $\boldsymbol{X}$ have length at most 1, the hinge loss simplifies to $f(\boldsymbol{u}) = 1 - Y\boldsymbol{u}^\intercal\boldsymbol{X}$ with gradient $\nabla f(\boldsymbol{u}) = -Y\boldsymbol{X}$. This implies that

$$\boldsymbol{u}^* := \arg\min_{\boldsymbol{u}} \mathbb{E}\left[f(\boldsymbol{u})\right] = \frac{\boldsymbol{\mu}}{\|\boldsymbol{\mu}\|}, \tag{30}$$

and

$$\begin{aligned}(\boldsymbol{w}-\boldsymbol{u}^*)^\intercal \mathbb{E}\left[\nabla f(\boldsymbol{w})\nabla f(\boldsymbol{w})^\intercal\right](\boldsymbol{w}-\boldsymbol{u}^*) &= (\boldsymbol{w}-\boldsymbol{u}^*)^\intercal \mathbb{E}\left[\boldsymbol{X}\boldsymbol{X}^\intercal\right](\boldsymbol{w}-\boldsymbol{u}^*)\\ &\le \lambda_{\text{max}}(\boldsymbol{w}-\boldsymbol{u}^*)^\intercal(\boldsymbol{w}-\boldsymbol{u}^*) \le 2\lambda_{\text{max}}(1-\boldsymbol{w}^\intercal\boldsymbol{u}^*)\\ &= \frac{2\lambda_{\text{max}}}{\|\boldsymbol{\mu}\|}(\boldsymbol{w}-\boldsymbol{u}^*)^\intercal(-\boldsymbol{\mu}) = \frac{2\lambda_{\text{max}}}{\|\boldsymbol{\mu}\|}(\boldsymbol{w}-\boldsymbol{u}^*)^\intercal \mathbb{E}\left[\nabla f(\boldsymbol{w})\right],\end{aligned}$$

which proves the first part of the lemma

For the second part, we first observe that $\lambda_{\text{max}} = 1/d$. Then, to compute $\|\boldsymbol{\mu}\|$, assume without loss of generality that $\|\bar{\boldsymbol{u}}\| = 1$, in which case $\bar{\boldsymbol{u}} = \boldsymbol{u}^*$. Now symmetry of the distribution of $\boldsymbol{X}$ conditional on $\boldsymbol{X}^\intercal\boldsymbol{u}^*$ gives

$$\mathbb{E}\left[Y\boldsymbol{X} \mid \boldsymbol{X}^\intercal\boldsymbol{u}^*\right] = \operatorname{sign}(\boldsymbol{X}^\intercal\boldsymbol{u}^*)\,\mathbb{E}\left[\boldsymbol{X} \mid \boldsymbol{X}^\intercal\boldsymbol{u}^*\right] = \operatorname{sign}(\boldsymbol{X}^\intercal\boldsymbol{u}^*)\boldsymbol{X}^\intercal\boldsymbol{u}^*\boldsymbol{u}^* = |\boldsymbol{X}^\intercal\boldsymbol{u}^*|\boldsymbol{u}^*.$$

By rotational symmetry, we may further assume without loss of generality that $\boldsymbol{u}^* = \boldsymbol{e}_1$ is the first unit vector in the standard basis, and therefore

$$\|\boldsymbol{\mu}\| = \|\,\mathbb{E}\left[|\boldsymbol{X}^\intercal\boldsymbol{u}^*|\right]\boldsymbol{u}^*\| = \mathbb{E}\left[|X_1|\right].$$

If $\boldsymbol{Z} = (Z_1, \ldots, Z_d)$ is multivariate Gaussian $\mathcal{N}(0, I)$. Then $\boldsymbol{X} = \boldsymbol{Z}/\|\boldsymbol{Z}\|$ is uniformly distributed on the sphere, so

$$\mathbb{E}[|X_1|] = \mathbb{E}\left[\frac{|Z_1|}{\|\boldsymbol{Z}\|}\right] \ge \frac{1}{4\sqrt{d}}\Pr\left(|Z_1| \ge \tfrac{1}{2} \wedge \|\boldsymbol{Z}\| \le 2\sqrt{d}\right).$$

Since $\Pr\left(|Z_1| < \frac{1}{2}\right) \le 0.4$ and $\Pr\left(\|\boldsymbol{Z}\| \ge 2\sqrt{d}\right) \le \frac{1}{4d}\mathbb{E}\left[\|\boldsymbol{Z}\|^2\right] = \frac{1}{4}$, we have

$$\Pr\left(|Z_1| \ge \tfrac{1}{2} \wedge \|\boldsymbol{Z}\| \le 2\sqrt{d}\right) \ge 1 - 0.4 - \frac{1}{4} = 0.35,$$

from which the conclusion of the second part follows with $c = 8/0.35$. ■

A.5 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 $Y \ge X$ by convexity of f and $Y \le C := 2D_2G_2$.

Lemma 18 *For $\beta \in (0,1]$, let X be a (B,β)-Bernstein random variable:*

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

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

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

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

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

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

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

$$\begin{aligned}\mathbb{E}[X^2] - B'\,\mathbb{E}[X]^\beta \;&\ge\; \mathbb{E}[X^2] - B'c_\beta\left(\frac{\mathbb{E}[X]}{\gamma} + \gamma^{\frac{\beta}{1-\beta}}\right) \;\ge\; \mathbb{E}[Y^2] - B'c_\beta\left(\frac{\mathbb{E}[Y]}{\gamma} + \gamma^{\frac{\beta}{1-\beta}}\right)\\ &=\; \mathbb{E}[Y^2] - B'\,\mathbb{E}[Y]^\beta, \end{aligned} \tag{31}$$

where the second inequality holds because $x^2 - c_\beta B' x/\gamma$ is a decreasing function of $x \le C$ for $\gamma \le \frac{c_\beta B'}{2C}$, 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 (31). ∎

Appendix B. Controller Regret Bound (Proof of Lemma 4)

We prove the lemma in two parts.

B.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

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

for any $\eta \in \mathcal{G}$ not expired after T rounds (see Definition 3). Note that by definition (8), 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 3). We redefine a^η from (10) as follows:

Definition 19 *We define the wakeup time of learning rate $\eta \in \mathcal{G}$ by*

$$a^\eta \;:=\; \inf\left\{ t \le T \,\middle|\, \eta > \frac{1}{2\left(\sum_{s=1}^{t-1} b_s \frac{B_{s-1}}{B_s} + B_{t-1}\right)} \right\} \wedge (T+1).$$

The difference with (10) is that we now 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 20 *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 tiny. 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 (12) at almost negligible cumulative cost using the clipping technique of Cutkosky (2019).

Lemma 21 (Clipping in the controller is cheap)

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

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), the surrogate loss (8) and its clipped version (12) to find

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

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 22 *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 (12), the range bound b_t in (5) and the wakeup time a_t in Definition 19,

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

■

In the next subsection we deal with the middle sum in Proposition 20. This part only depends on the construction of the controller. We deal with the final sum in the section after that.

B.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 tiny.

Lemma 23 *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*

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

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 $O(\ln \ln T)$.

Proof We first observe that Algorithm 1, 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[4]

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

Algorithm 1 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 (12) of clipped loss and gradient to obtain $(\boldsymbol{w}_t - \boldsymbol{w}_t^\eta)^\intercal \bar{\boldsymbol{g}}_t \geq -B_{t-1}$, and we further use that p_t is supported on $\mathcal{A}_t$, which implies that $\eta \leq \frac{1}{2B_{t-1}}$. Together these license[5] the "prod bound" ($e^{x - x^2} \leq 1 + x$ 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

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

[4] 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.

[5] 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 $\eta \mapsto -\ln(1 + \eta(\boldsymbol{w}_t - \boldsymbol{w}_t^\eta)^\intercal \bar{\boldsymbol{g}}_t)$ (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.

It remains to bound the maximum number of active grid-points during any epoch. Recall from (9) that the active set at any time t is

$$\mathcal{A}_t \;=\; \left(\frac{1}{2\left(\sum_{s=1}^{t-1} b_s \frac{B_{s-1}}{B_s} + B_{t-1}\right)}, \frac{1}{2B_{t-1}}\right] \cap \mathcal{G}.$$

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

$$\max\left\{\eta \in \mathcal{G} \;\middle|\; \eta \le \frac{1}{2B_\tau}\right\}.$$

As we consider the part of the epoch up to time $t \ge \tau + 1$, the smallest η active in the epoch is

$$\min\left\{\eta \in \mathcal{G} \;\middle|\; \eta \ge \frac{1}{2\left(\sum_{s=1}^{t-1} b_s \frac{B_{s-1}}{B_s} + B_{t-1}\right)}\right\}.$$

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

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

where the second inequality holds because of the reset condition (11). All together, we conclude that our prior costs for the improper (uniform on $\mathcal{G}$) prior are upper bounded by

$$\ln \left| \bigcup_{s \in (\tau, t]} \mathcal{A}_s \right| \;\le\; \ln \left\lceil 2\log_2 \left(\sum_{s=1}^{t-1} \frac{b_s}{B_s} + 1\right) \right\rceil_+. \tag{32}$$

■

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

Lemma 24 (Total regret in far past is tiny) *Consider two consecutive epochs, starting after $\tau_1 < \tau_2$, and let η be not expired after τ_1 rounds. Then*

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

Proof

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

where the last inequality is the reset condition (11) at time τ_2. ■

We are now ready to compose the previous two lemmas to obtain the following result:

Lemma 25 (Overall controller specialists regret bound) *Let η be not expired after T rounds. Then*

$$\sum_{t=a^\eta}^{T}\left(\bar{\ell}_t^\eta(\boldsymbol{w}_t)-\bar{\ell}_t^\eta(\boldsymbol{w}_t^\eta)\right) \;\leq\; \eta B_T + 2\ln\left\lceil 2\log_2\left(\sum_{t=1}^{T-1}\frac{b_t}{B_t}+1\right)\right\rceil. \tag{33}$$

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, Lemma 23, to the last two epochs each. Suppose these start after τ_1 and τ_2. For any $\eta \in \mathcal{G}$ that has not expired after T rounds, we find

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

The regret on all epochs except the last two is bounded by Lemma 24. So together we obtain (33). Alternatively, suppose there are 2 epochs. Then, since we get no clipped regret in the 1st epoch (as $B_{t-1}=0$ throughout it, and hence $\bar{\boldsymbol{g}}_t=\mathbf{0}$ and $\bar{\ell}_t^\eta(\cdot)=0$), 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 (33) also holds. Finally, if there is only 1 epoch, then our clipped regret is 0, so (33) also holds. ■

The proof of Lemma 4 is completed by plugging in the upper bounds from Lemmas 21, 22 and 25 into Proposition 20.

Appendix C. Composition Proofs of Theorems 7, 9 and 12

We combine the proofs of Theorems 7 and 12, which are both special cases of the abstract result Theorem 26 below. The proof of Theorem 9 is very similar in spirit, but sufficiently different that we postpone it to the end of the section.

Theorem 26 *Suppose there exist a number $V \geq 0$ and positive semi-definite matrices $\boldsymbol{F}^\eta$ (possibly dependent on η) such that $\operatorname{rk}(\boldsymbol{F}^\eta) \leq r$, $\operatorname{tr}(\boldsymbol{F}^\eta) \leq s$ and the linearized regret is at most*

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \eta V + \frac{\ln\det\left(\boldsymbol{I} + 2\eta^2\sigma^2\boldsymbol{F}^\eta\right) + \frac{1}{2\sigma^2}\|\boldsymbol{u}\|_2^2 + 2\ln\lceil 2\log_2 T\rceil_+ + \frac{1}{2}}{\eta} + 2B_T$$

simultaneously for all $\eta \in \mathcal{G}$ such that $\eta \leq \frac{1}{2B_T}$. Then the linearized regret is both bounded by

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \frac{5}{2}\sqrt{V(\tfrac{1}{2\sigma^2}\|\boldsymbol{u}\|_2^2 + Z_T)} + 5B_T(\tfrac{1}{2\sigma^2}\|\boldsymbol{u}\|_2^2 + Z_T) + 2B_T,$$

where $Z_T = r\ln\left(1 + \frac{\sigma^2 s}{2B_T^2 r}\right) + 2\ln\lceil 2\log_2 T\rceil_+ + \frac{1}{2}$, and by

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \frac{5}{2}\sqrt{\Big(V + 2\sigma^2 s\Big)\Big(\tfrac{1}{2\sigma^2}\|\boldsymbol{u}\|_2^2 + Z_T'\Big)} + 5B_T\Big(\tfrac{1}{2\sigma^2}\|\boldsymbol{u}\|_2^2 + Z_T'\Big) + 2B_T,$$

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

Theorem 7 corresponds to the case $V = V_T^{\boldsymbol{u}}$ and $\boldsymbol{F}^\eta = \boldsymbol{F}_T$, such that $\operatorname{tr}(\boldsymbol{F}^\eta) = \sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2$; Theorem 12 is obtained with $V = V_T^{\boldsymbol{u}} + \frac{2\sigma^2 m\Omega_q}{m-q}$ and $\boldsymbol{F}^\eta = (\boldsymbol{S}_T^\eta)^\intercal \boldsymbol{S}_T^\eta$. To bound $\operatorname{tr}(\boldsymbol{F}^\eta)$ we use that $(\boldsymbol{S}_T^\eta)^\intercal \boldsymbol{S}_T^\eta \preceq (\boldsymbol{G}_T^\eta)^\intercal \boldsymbol{G}_T^\eta = \sum_{t=a^\eta}^T \boldsymbol{g}_t\boldsymbol{g}_t^\intercal \preceq \boldsymbol{F}_T$, where the first inequality holds because $\boldsymbol{S}_T^\eta$ is the Frequent Directions approximation of $\boldsymbol{G}_T^\eta$ (Ghashami et al., 2016). It follows that $\operatorname{tr}(\boldsymbol{F}^\eta) \leq \operatorname{tr}(\boldsymbol{F}_T) = \sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2$. We may further use that $\operatorname{rk}(\boldsymbol{F}^\eta) \leq 2m$, by the dimensionality of $\boldsymbol{S}_T^\eta$. The precondition of Theorem 26 is established by Theorems 6 and 11, respectively, and the observation that $Q_T \leq T$.

To prove Theorem 26 we start with a general lemma about optimizing in η:

Lemma 27 *For any $X, Y > 0$,*

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

Proof Let us denote the unconstrained optimizer of the left-hand side by $\hat{\eta} = \sqrt{Y/X}$. We distinguish two cases: first, when $\hat{\eta} \leq \frac{1}{2B_T}$, 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

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

In the second case, if $\hat{\eta} > \frac{1}{2B_T}$, we plug in the highest available grid point (for which the worst case is $\frac{1}{4B_T}$) to find

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

where the second inequality follows by the assumption that $\hat{\eta} > \frac{1}{2B_T}$. In both cases the conclusion of the lemma follows. ∎

Proof (Theorem 26) We start with the first claim of the theorem. By assumption, for any $\eta \leq \frac{1}{2B_T}$ in the grid $\mathcal{G}$, we have

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \eta V + \frac{A^\eta}{\eta} + 2B_T \;\leq\; \eta V + \frac{A}{\eta} + 2B_T,$$

where

$$\begin{aligned} A^\eta &= \ln\det\big(\boldsymbol{I} + 2\eta^2\sigma^2\boldsymbol{F}^\eta\big) + \frac{1}{2\sigma^2}\|\boldsymbol{u}\|_2^2 + 2\ln\lceil 2\log_2 T\rceil + \frac{1}{2} \\ A &= r\ln\Big(1 + \frac{\sigma^2 s}{2B_T^2 r}\Big) + \frac{1}{2\sigma^2}\|\boldsymbol{u}\|_2^2 + 2\ln\lceil 2\log_2 T\rceil + \frac{1}{2}, \end{aligned}$$

and $A^\eta \leq A$ follows from $\eta \leq 1/(2B_T)$, the first inequality in Lemma 28 below and the fact that the expression $r\ln\big(1+\frac{s}{r}\big)$ is increasing in $r \geq 0$ for all $s \geq 0$. Lemma 27 therefore implies that

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \frac{5}{2}\sqrt{VA} + 5B_T A + 2B_T,$$

which establishes the first claim of the theorem.

For the second claim of the theorem, we upper bound A^η differently, using the second inequality in Lemma 28, to obtain

$$\tilde{R}_T^{\boldsymbol{u}} \;\leq\; \eta V + 2\eta\sigma^2 s + \frac{A'}{\eta} + 2B_T, \qquad \text{where} \qquad A' \;=\; \frac{1}{2\sigma^2}\|\boldsymbol{u}\|_2^2 + 2\ln\lceil 2\log_2 T\rceil + \frac{1}{2}.$$

Using Lemma 27, the second claim follows, which completes the proof of the theorem. ∎

Lemma 28 *For any positive semi-definite matrix $\boldsymbol{M} \in \mathbb{R}^{d\times d}$*

$$\ln\det(\boldsymbol{I}+\boldsymbol{M}) \;\leq\; \operatorname{rk}(\boldsymbol{M})\ln\left(1 + \frac{\operatorname{tr}(\boldsymbol{M})}{\operatorname{rk}(\boldsymbol{M})}\right) \;\leq\; \operatorname{tr}(\boldsymbol{M}),$$

where the middle term is extended by continuity to equal zero at $\boldsymbol{M} = \boldsymbol{0}$.

Proof If $\boldsymbol{M} = \boldsymbol{0}$ then all three equal zero and we are done. Otherwise, let $\lambda_1, \ldots, \lambda_d$ be the eigenvalues of $\boldsymbol{M}$. Then $(1+\lambda_1), \ldots, (1+\lambda_d)$ are the eigenvalues of $\boldsymbol{I}+\boldsymbol{M}$, and Jensen's inequality implies

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

which proves the first inequality. The second inequality follows by $\ln(1+x) \le x$ for all $x \ge 0$. ■

To conclude the section, it remains to prove Theorem 9.

Proof (Theorem 9) Starting from Theorem 6, which still holds even though σ depends on η, we plug in $\sigma = 1/\sqrt{\alpha\eta}$ to obtain

$$\begin{aligned} \tilde{R}_T^{\boldsymbol{u}} &\le \eta V_T^{\boldsymbol{u}} + \frac{A}{\eta} + \frac{\alpha}{2}\|\boldsymbol{u}\|_2^2 + 2B_T, \\ &\qquad \text{where} \qquad A = \ln\det\left(\boldsymbol{I} + \frac{1}{B_T\alpha}\boldsymbol{F}_T\right) + 2\ln\lceil 2\log_2 T\rceil_+ + \frac{1}{2}, \end{aligned}$$

for all $\eta \in \mathcal{G}$ such that $\eta \le 1/(2B_T)$. Lemma 27 therefore implies that

$$\tilde{R}_T^{\boldsymbol{u}} \le \frac{5}{2}\sqrt{V_T^{\boldsymbol{u}} A} + 5B_T A + \frac{\alpha}{2}\|\boldsymbol{u}\|_2^2 + 2B_T,$$

and the first claim of the theorem follows upon applying the first inequality from Lemma 28 with $\boldsymbol{M} = \frac{1}{B_T\alpha}\boldsymbol{F}_T$ and observing that $\operatorname{tr}(\boldsymbol{F}_T) = \sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2$.

For the second claim of the theorem, we again start from Theorem 6 and now apply the second inequality from Lemma 28 for $\boldsymbol{M} = \frac{2\eta}{\alpha}\boldsymbol{F}_T$ to obtain

$$\tilde{R}_T^{\boldsymbol{u}} \le \eta V_T^{\boldsymbol{u}} + \frac{Z_T'}{\eta} + \frac{2}{\alpha}\operatorname{tr}(\boldsymbol{F}_T) + \frac{\alpha}{2}\|\boldsymbol{u}\|_2^2 + 2B_T.$$

Using Lemma 27 and $\operatorname{tr}(\boldsymbol{F}_T) = \sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2$, the second claim follows, which completes the proof of the theorem. ■

Appendix D. Proofs of Corollaries 8 and 14

Proof (Corollary 8) If we ignore the corner case that B_T^2 in the definition of Z_T is exceedingly small, then (21) follows from (20) upon bounding $\|\boldsymbol{g}_t\|_2^2 \le G_2^2$, $B_T \le 2D_2G_2$, and observing that Z_T is increasing in $\operatorname{rk}(\boldsymbol{F}_T) \le d$. To see that (21) holds in general, even for very small B_T^2, we need to verify that $V_T^{\boldsymbol{u}} Z_T = O(V_T^{\boldsymbol{u}} d\ln(D_2G_2T/d))$ and

$B_T Z_T = O(D_2 G_2 \ln(D_2 G_2 T/d))$. To establish the first of these, we reason as follows:

$$\begin{aligned}
V_T^{\boldsymbol{u}} \operatorname{rk}(\boldsymbol{F}_T) \ln \left(1 + \frac{\sigma^2 \sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2}{8B_T^2 \operatorname{rk}(\boldsymbol{F}_T)}\right) \\
&\le V_T^{\boldsymbol{u}} d \ln\left(1 + \frac{D_2^2 G_2^2 T}{8B_T^2 d}\right)\\
&= V_T^{\boldsymbol{u}} d \ln\left(\frac{D_2^2 G_2^2 T^3}{d}\right) + V_T^{\boldsymbol{u}} d \ln\left(\frac{d}{D_2^2 G_2^2 T^3} + \frac{1}{8B_T^2 T^2}\right)\\
&\le V_T^{\boldsymbol{u}} d \ln\left(\frac{D_2^2 G_2^2 T^3}{d}\right) + V_T^{\boldsymbol{u}} \left(\frac{d^2}{D_2^2 G_2^2 T^3} + \frac{d}{8B_T^2 T^2}\right)\\
&\le V_T^{\boldsymbol{u}} d \ln\left(\frac{D_2^2 G_2^2 T^3}{d}\right) + \frac{2d^2}{T^2} + \frac{d}{8T} = O\left(V_T^{\boldsymbol{u}} d \ln \frac{D_2 G_2 T}{d}\right),
\end{aligned}$$

where the last inequality follows from $V_T^{\boldsymbol{u}} \le B_T^2 T \le 2D_2^2 G_2^2 T$. To establish the second case, we observe that

$$\begin{aligned}
B_T \operatorname{rk}(\boldsymbol{F}_T) \ln \left(1 + \frac{\sigma^2 \sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2}{8B_T^2 \operatorname{rk}(\boldsymbol{F}_T)}\right) &\le B_T d \ln\left(1 + \frac{D_2^2 G_2^2 T}{8B_T^2 d}\right)\\
&= B_T d \ln\left(B_T^2 + \frac{D_2^2 G_2^2 T}{8d}\right) - 2dB_T \ln B_T\\
&\le 2D_2 G_2 d \ln(4D_2^2 G_2^2 + \frac{D_2^2 G_2^2 T}{8d}) + \frac{2d}{e} = O\left(D_2 G_2 d \ln\left(\frac{D_2 G_2 T}{d}\right)\right),
\end{aligned}$$

where the last inequality uses that $x \ln x \ge -1/e$. This completes the proof of (21). ■

Proof (Corollary 14) To see that (24) follows from (22), we need to verify that $V_{T,i}^{u_i} Z_{T,i} = O(V_{T,i}^{u_i} \ln(D_\infty G_\infty T))$ and $B_{T,i} Z_{T,i} = O(D_\infty G_\infty \ln(D_\infty G_\infty T))$. These follow as the one-dimensional special cases of the analogous quantities in the proof of Corollary 8.

The first part of (25) then follows from (23) upon observing that $B_{T,i} \le 2D_\infty \max_t \|\boldsymbol{g}_t\|_1 \le 2D_\infty G_2 \sqrt{d}$. The second part follows because

$$\sum_{i=1}^d \|g_{1:T,i}\|_2 = d \sum_{i=1}^d \frac{1}{d} \sqrt{\sum_{t=1}^T g_{t,i}^2} \le d \sqrt{\sum_{i=1}^d \frac{1}{d} \sum_{t=1}^T g_{t,i}^2} = \sqrt{d \sum_{t=1}^T \|\boldsymbol{g}_t\|_2^2} \le G_2 \sqrt{dT}$$

by Jensen's inequality. ■

Appendix E. Experimental Results

Data set	T	d	Outcome	$P(y = 1)$
a9a	32561	123	binary	0.24
australian	690	14	binary	0.44
breast-cancer	683	9	binary	0.35
covtype	581012	54	binary	0.49
diabetes	768	8	binary	0.65
heart	270	13	binary	0.44
ijcnn1	91701	22	binary	0.10
ionosphere	351	34	binary	0.64
phishing	11055	68	binary	0.56
splice	1000	60	binary	0.52
w8a	49479	300	binary	0.03
abalone	4177	8	real	
bodyfat	252	14	real	
cpusmall	8192	12	real	
housing	506	13	real	
mg	1385	6	real	
space_ga	3107	6	real	

Table 2: Summary of the data sets

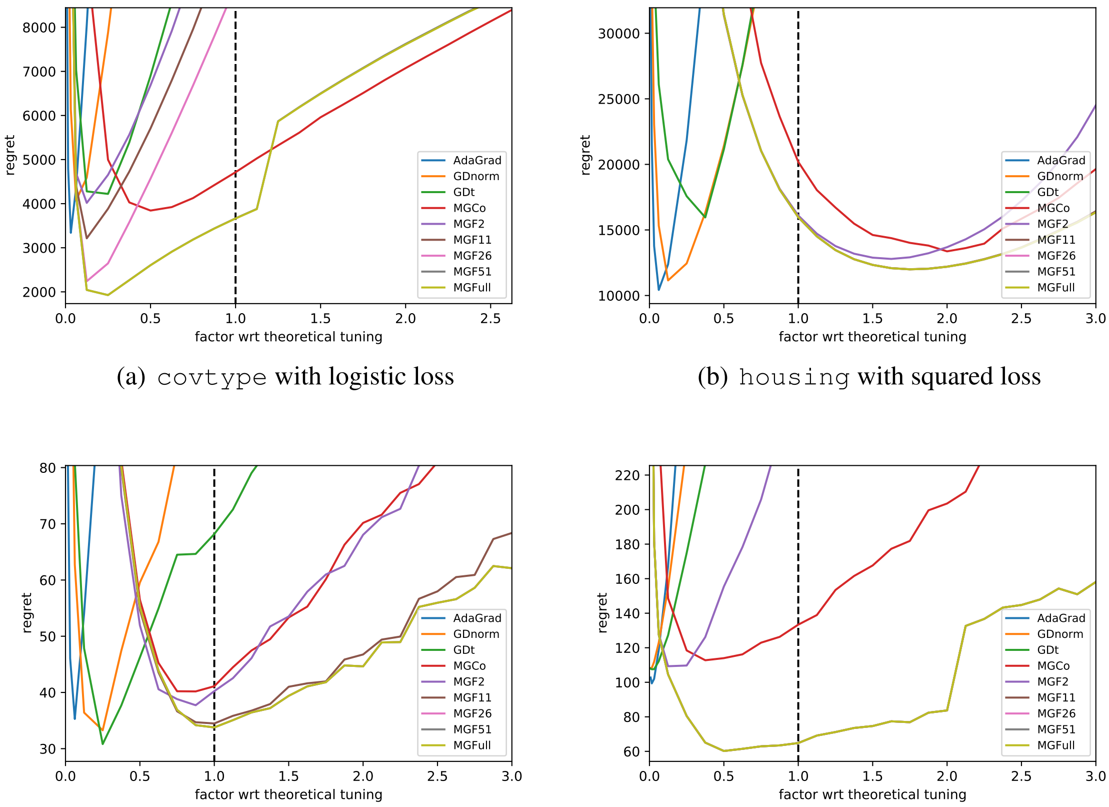
Data set	Loss	AdaGrad	GDnorm	OGDt	MGCo	MGF2	MGF11	MGF26	MGF51	MGFull
a9a	hinge	232414	37708	22472	17012	13754	12230	11671	11160	**11045**
	logistic	30910	7176	3817	**1340**	2249	1990	1910	1813	1783
australian	hinge	279	99	68	41	40	**34**	**34**	**34**	**34**
	logistic	1250	492	359	48	52	**45**	**45**	**45**	**45**
breast-cancer	hinge	214	106	84	**24**	26	25	25	25	25
	logistic	288	147	119	**25**	**26**	**26**	**26**	**26**	**26**
covtype	hinge	1317765	254930	141706	66797	83958	71218	62087	31368	**31355**
	logistic	78430	33935	14042	4713	12214	10516	8941	3668	**3663**
diabetes	hinge	553	306	185	75	63	**59**	**59**	**59**	**59**
	logistic	474	241	133	53	40	**39**	**39**	**39**	**39**
heart	hinge	329	217	148	**35**	42	**35**	**35**	**35**	**35**
	logistic	376	246	155	**30**	35	32	31	31	31
ijcnn1	hinge	12292	3925	1198	**885**	1633	1327	901	901	901
	logistic	15303	4473	1344	**976**	1798	1415	1086	1086	1086
ionosphere	hinge	2672	1102	753	**169**	252	211	206	205	205
	logistic	5786	1897	1426	240	280	242	**238**	**238**	**238**
phishing	hinge	6752	3162	1757	610	635	607	547	**518**	**518**
	logistic	22814	7394	3320	1208	967	890	802	**767**	**767**
splice	hinge	2451	777	694	**243**	303	290	277	288	280
	logistic	3014	819	726	183	182	181	179	177	**175**
w8a	hinge	349174	139920	255346	**18789**	34395	31966	32080	31823	29661
	logistic	86921	21095	40519	**3324**	4546	4230	4049	3977	3865
abalone	absolute	12650	7395	5027	1317	2194	**748**	**748**	**748**	**748**
	squared	73507	44166	37398	6725	7642	**6179**	**6179**	**6179**	**6179**
bodyfat	absolute	319	98	75	30	24	**23**	**23**	**23**	**23**
	squared	351	37	28	10	**7**	**8**	**8**	**8**	**8**
cpusmall	absolute	533948	199595	182464	40537	22251	14301	**14287**	**14287**	**14287**
	squared	12109845	2740512	3082005	561505	353329	**351253**	351257	351257	351257
housing	absolute	9979	3557	3067	946	949	776	**746**	**746**	**746**
	squared	154729	52053	55064	20191	16103	**15973**	15975	15975	15975
mg	absolute	277	110	92	30	40	**28**	**28**	**28**	**28**
	squared	112	32	**15**	19	17	18	18	18	18
space_ga	absolute	1393	908	523	133	259	**65**	**65**	**65**	**65**
	squared	1451	534	528	**40**	75	55	55	55	55

Table 3: The regret of each algorithm for the various data sets and loss functions (rounded to whole numbers). Boldface indicates that the regret is within one unit of the minimum for the row.

E.1 Hypertune Results

In this section we investigate the effect of hyperparameter tuning. Each of the algorithms that we consider has one free parameter, σ, for which the theory advocates tuning it in terms of the (unknown) norm of the comparator, or the maximal distance from the comparator within the domain. This theoretical recommendation is what we employed in our experiments in Section 8. In contrast, we now ask what performance one may reach by optimizing the σ parameter for the data in hand. Our approach will be to evaluate all algorithms on a discrete grid of parameter settings. For convenience of comparison between full and coordinate-wise algorithms, we parameterise our grid by the factor by which we scale the theoretically optimal tuning from Section 8. We include in our grid exponentially small factors 2^j for $j = -7, \dots, -3$, followed by a linear grid running from 2^{-3} to 3 with steps of size $1/8$, resulting in a grand total of 28 grid points. We visualise the entire performance profile for four selected data sets in Figure 3. There we see that the optimal tuning for σ can be either higher or lower than the theoretical recommendation, and whether it should be higher or lower can be different for different algorithms even on the same data set.

We evaluate our algorithms on all data sets. (Recall that a helpful summary of the properties of each data set can be found in Table 2). First, in Table 5, which parallels Table 3, we present the hypertuned regret for each algorithm on each data set. These results are subsequently summarised by Table 4, which is the hypertuned analogue of Table 1. Here we compare all algorithms to AdaGrad instead of OGDt, as it has the best hypertuned performance among prior existing algorithms. (Interestingly, the theoretical prediction that OGDnorm dominates OGDt does materialise for the hypertuned regret, while it did not under the bonafide tuning of Section 8.) We can conclude from Table 4 that AdaGrad, OGDnorm and MetaGrad, in either full or sketched forms, all have very similar performance. As discussed in Section 8.3, this suggests that the empirical superiority of MetaGrad in the experiments from Section 8 may be attributed to its ability to better adapt to the optimal learning rate η. We should also remember that the hypertuned performance is not a-priori indicative of practical results. It is an interesting challenge to develop new methods that achieve as much of this hypertuned performance in practice as possible, but which also come with corresponding theoretical guarantees. In this quest MetaGrad constitutes a solid first step.



(a) `covtype` with logistic loss

(b) `housing` with squared loss

(c) `australian` with hinge loss

(d) `space_ga` with absolute loss

Figure 3: Performance of all algorithms as a function of the tuning parameter σ on four selected data sets. We have parameterised σ by a factor times its theoretically optimal tuning. The dotted line indicates the standard tuning (corresponding to factor 1), at which the results from Table 3 are reproduced. Note that with different σ, the algorithms produce different iterates $\boldsymbol{w}_t$, and as a result see different gradients. For MetaGrad, these further affect the set of active experts that are maintained by the master. These effects make the curves interestingly non-smooth.

Algorithm	# best	# better than AdaGrad	MedianRatio
AdaGrad	14	34	1.00
OGDnorm	8	18	0.99
OGDt	8	14	1.07
MGCo	1	6	1.15
MGF2	1	10	1.09
MGF11	6	18	1.01
MGF26	4	18	1.01
MGF51	7	18	1.01
MGFull	9	18	1.01

Table 4: Comparison of algorithms with AdaGrad, with the σ hyperparameter optimized in hindsight for the data. The MedianRatio column contains the median ratio of the regret of each algorithm over that of AdaGrad. Columns "# best" and "# better than AdaGrad" count cases where the algorithm is at most one regret unit above the best algorithm or AdaGrad, respectively.

Data set	Loss	AdaGrad	OGDnorm	OGDt	MGCo	MGF2	MGF11	MGF26	MGF51	MGFull
a9a	hinge	1512	**484**	504	664	627	592	588	583	585
	logistic	**304**	412	472	527	512	484	478	473	473
australian	hinge	35	33	**31**	40	38	34	34	34	34
	logistic	**25**	26	**24**	33	36	34	34	34	34
breast-cancer	hinge	**20**	**20**	**19**	23	23	22	22	22	22
	logistic	21	**19**	26	23	24	24	24	24	24
covtype	hinge	8070	6382	6205	9095	6811	5648	5067	**4939**	**4939**
	logistic	3339	4077	4222	3844	4017	3214	2240	1927	**1926**
diabetes	hinge	**58**	73	76	71	62	59	59	59	59
	logistic	**36**	50	55	49	40	39	39	39	39
heart	hinge	**34**	35	**33**	35	35	**33**	**34**	**34**	**34**
	logistic	**28**	**28**	30	30	30	29	29	29	29
ijcnn1	hinge	**419**	550	542	597	751	640	502	502	502
	logistic	**500**	663	782	804	1021	823	715	715	715
ionosphere	hinge	106	**103**	110	110	110	108	108	108	108
	logistic	106	**98**	111	106	104	103	103	103	103
phishing	hinge	**290**	471	433	378	326	311	301	303	303
	logistic	**258**	457	492	423	345	335	331	330	330
splice	hinge	210	200	211	179	**175**	177	180	178	177
	logistic	150	147	174	137	139	138	137	**136**	**136**
w8a	hinge	3299	1458	2545	935	875	875	875	**873**	**873**
	logistic	1147	1123	2764	1224	1159	1133	1124	1121	**1117**
abalone	absolute	1038	1040	1033	1211	1131	**692**	**692**	**692**	**692**
	squared	6204	6950	7627	6698	7127	**6179**	**6179**	**6179**	**6179**
bodyfat	absolute	23	**18**	**17**	28	24	23	23	23	23
	squared	6	**3**	**4**	7	6	6	6	6	6
cpusmall	absolute	11379	10608	10577	15284	10489	**9922**	9976	9976	9976
	squared	479645	478014	694804	545240	279054	**278921**	278947	278947	278947
housing	absolute	**666**	794	795	866	894	776	746	746	746
	squared	**10425**	11150	15954	13368	12790	12002	11995	11995	11995
mg	absolute	20	15	**14**	30	36	28	28	28	28
	squared	9	6	**4**	13	13	12	12	12	12
space_ga	absolute	99	108	108	113	109	**60**	**60**	**60**	**60**
	squared	**40**	43	43	**40**	45	45	45	45	45

Table 5: The regret of each algorithm for the various data sets and loss functions, with the σ hyperparameter of each method optimized in hindsight for the data. Boldface indicates the regret differs less than 1 from the minimum regret for the row.

References

Naman Agarwal, Brian Bullins, Xinyi Chen, Elad Hazan, Karan Singh, Cyril Zhang, and Yi Zhang. Efficient full-matrix adaptive regularization. In *Proceedings of the 36th Annual International Conf. on Machine Learning (ICML)*, volume 97, pages 102–110, June 2019.

Peter L. Bartlett and Shahar Mendelson. Empirical minimization. *Probability Theory and Related Fields*, 135(3):311–334, 2006.

Peter L. Bartlett, Elad Hazan, and Alexander Rakhlin. Adaptive online gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS) 20*, pages 65–72, 2007.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

Chao-Kai Chiang, Tianbao Yang, Chia-Jung Le, Mehrdad Mahdavi, Chi-Jen Lu, Rong Jin, and Shenghuo Zhu. Online optimization with gradual variations. In *Proc. of the 25th Annual Conf. on Learning Theory (COLT)*, pages 6.1–6.20, 2012.

Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems (NeurIPS) 22*, pages 414–422, 2009.

Ashok Cutkosky. Artificial constraints and hints for unbounded online learning. In *Proceedings of the 32nd Annual Conference on Learning Theory (COLT)*, volume 99, pages 874–894, June 2019.

Ashok Cutkosky and Kwabena A. Boahen. Online learning without prior information. In *Proceedings of the 30th Annual Conference on Learning Theory (COLT)*, pages 643–677, 2017.

Ashok Cutkosky and Francesco Orabona. Black-box reductions for parameter-free online learning in Banach spaces. In *Proceedings of the 31st Annual Conference on Learning Theory (COLT)*, volume 75, pages 1493–1529, 2018.

Raphaël Deswarte. *Linear regression and learning: contributions to regularization and aggregation methods*. PhD thesis, Université Paris-Saclay, 2018.

Chuong B. Do, Quoc V. Le, and Chuan-Sheng Foo. Proximal regularization for online and batch learning. In *Proc. of the 26th Annual International Conf. on Machine Learning (ICML)*, pages 257–264, 2009.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Tim van Erven and Wouter M. Koolen. MetaGrad: Multiple learning rates in online learning. In *Advances in Neural Information Processing Systems (NeurIPS) 29*, pages 3666–3674, December 2016.

Tim van Erven, Peter D. Grünwald, Nishant A. Mehta, Mark D. Reid, and Robert C. Williamson. Fast rates in statistical and online learning. *Journal of Machine Learning Research*, 16:1793–1861, 2015.

Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. In *Proc. 29th Annual ACM Symposium on Theory of Computing*, pages 334–343. ACM, 1997.

Pierre Gaillard, Gilles Stoltz, and Tim van Erven. A second-order bound with excess losses. In *Proc. of the 27th Annual Conf. on Learning Theory (COLT)*, pages 176–196, 2014.

Mina Ghashami, Edo Liberty, Jeff M. Phillips, and David P. Woodruff. Frequent directions: Simple and deterministic matrix sketching. *SIAM Journal on Computing*, 45(5):1762–1792, 2016.

Gene H. Golub and Charles F. Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

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

Elad Hazan and Satyen Kale. Extracting certainty from uncertainty: Regret bounded by variation in costs. *Machine learning*, 80(2-3):165–188, 2010.

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

Dirk van der Hoeven, Tim van Erven, and Wojciech Kotłowski. The many faces of exponential weights in online learning. In *Annual Conference on Learning Theory (COLT)*, pages 2067–2092, 2018.

Ruitong Huang, Tor Lattimore, András György, and Csaba Szepesvari. Following the leader and fast rates in linear prediction: Curved constraint sets and other regularities. In *Advances in Neural Information Processing Systems 29*, pages 4970–4978, 2016.

Michal Kempka, Wojciech Kotłowski, and Manfred K. Warmuth. Adaptive scale-invariant online algorithms for learning linear models. In *Proceedings of the 36th Annual International Conf. on Machine Learning (ICML)*, pages 3321–3330, 2019.

Wouter M. Koolen and Tim van Erven. Second-order quantile methods for experts and combinatorial games. In *Proc. of the 28th Annual Conf. on Learning Theory (COLT)*, pages 1155–1175, 2015.

Wouter M. Koolen, Tim van Erven, and Peter D. Grünwald. Learning the learning rate for prediction with expert advice. In *Advances in Neural Information Processing Systems (NeurIPS) 27*, pages 2294–2302, 2014.

Wouter M. Koolen, Peter D. Grünwald, and Tim van Erven. Combining adversarial guarantees and stochastic fast rates in online learning. In *Advances in Neural Information Processing Systems (NeurIPS) 29*, pages 4457–4465, 2016.

Haipeng Luo, Alekh Agarwal, Nicolò Cesa-Bianchi, and John Langford. Efficient second order online learning by sketching. In *Advances in Neural Information Processing Systems 29*, pages 902–910, 2016.

Haipeng Luo, Alekh Agarwal, Nicolò Cesa-Bianchi, and John Langford. Efficient second order online learning by sketching. *ArXiv preprint: arXiv:1602.02202v4*, 2017. This is an updated version of Luo et al. (2016) with corrections.

Luo Luo, Cheng Chen, Zhihua Zhang, Wu-Jun Li, and Tong Zhang. Robust frequent directions with application in online learning. *Journal of Machine Learning Research*, 20(45):1–41, 2019.

Brendan McMahan and Matthew Streeter. No-regret algorithms for unconstrained online convex optimization. In *Advances in Neural Information Processing Systems (NeurIPS) 25*, pages 2402–2410, 2012.

H. Brendan McMahan and Matthew J. Streeter. Adaptive bound optimization for online convex optimization. In *Proc. of the 23rd Annual Conf. on Learning Theory (COLT)*, pages 244–256, 2010.

Zakaria Mhammedi and Wouter M. Koolen. Lipschitz and comparator-norm adaptivity in online learning. In *Proceedings of the 33rd Annual Conference on Learning Theory (COLT)*, pages 2858–2887, July 2020.

Zakaria Mhammedi, Wouter M. Koolen, and Tim van Erven. Lipschitz adaptivity with multiple learning rates in online learning. In *Proceedings of the 32nd Annual Conference on Learning Theory (COLT)*, pages 2490–2511, June 2019.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. International Conf. on Learning Representations, 2013. URL `http://arxiv.org/abs/1301.3781`.

Jaouad Mourtada and Stéphane Gaïffas. On the optimality of the Hedge algorithm in the stochastic regime. *Journal of Machine Learning Research*, 20(83):1–28, 2019.

Yurii Nesterov. *Introductory lectures on convex optimization : a basic course*. Applied optimization. Kluwer Academic Publ., 2004.

Yurii Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.

Thom W. H. Neuteboom. Modifying Squint for prediction with expert advice in a changing environment. *Bachelor Thesis*, 2020. To appear at `https://www.universiteitleiden.nl/en/science/mathematics/education/theses#bachelor-theses-mathematics`.

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

Francesco Orabona and Dávid Pál. Scale-free online learning. *Theoretical Computer Science*, 716:50–69, 2018. Special Issue on ALT 2015.

Francesco Orabona, Koby Crammer, and Nicolò Cesa-Bianchi. A generalized online mirror descent with applications to classification and regression. *Machine Learning*, 99(3):411–435, 2015.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2012.

Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical Programming*, 127(1):3–30, 2011.

Nathan Srebro, Karthik Sridharan, and Ambuj Tewari. Smoothness, low noise and fast rates. In *Advances in Neural Information Processing Systems (NeurIPS) 23*, pages 2199–2207, 2010.

Jacob Steinhardt and Percy Liang. Adaptivity and optimism: An improved exponentiated gradient algorithm. In *Proc. of the 31th Annual International Conf. on Machine Learning (ICML)*, pages 1593–1601, 2014.

Ryan Tibshirani. Optimal rates in convex optimization. With Larry Wasserman, 2014. URL `http://www.stat.cmu.edu/~larry/=sml/optrates.pdf`.

Tim van Erven, Wouter M. Koolen, and Dirk van der Hoeven. Code for experiments in the paper "MetaGrad: Adaptation using multiple learning rates in online learning". `https://github.com/DirkvdH/Online-Appendix-MetaGrad`, 2021.

Volodya Vovk. Competitive on-line statistics. *International Statistical Review*, 69(2):213–248, 2001.

Guanghui Wang, Shiyin Lu, and Lijun Zhang. Adaptivity and optimality: A universal algorithm for online convex optimization. In *Proc. of the 35th Uncertainty in Artificial Intelligence Conference*, pages 659–668, 2020.

Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596, 2010.

Lijun Zhang, Guanghui Wang, Weiwei Tu, and Zhi-Hua Zhou. Dual adaptivity: A universal algorithm for minimizing the adaptive regret of convex functions. *CoRR*, abs/1906.10851, 2019. URL `http://arxiv.org/abs/1906.10851`.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proc. of the 20th Annual International Conf. on Machine Learning (ICML)*, pages 928–936, 2003.

Martin Zinkevich. *Theoretical Guarantees for Algorithms in Multi-Agent Settings*. PhD thesis, Carnegie Mellon University, 2004.