Implicit Online Learning

Brian Kulis KULIS@EECS.BERKELEY.EDU
ICSI and Department of EECS, University of California, Berkeley, CA 94720, USA

Peter L. Bartlett BARTLETT@EECS.BERKELEY.EDU
Department of EECS and Department of Statistics, University of California, Berkeley, CA 94720, USA

Abstract

Online learning algorithms have recently risen to prominence due to their strong theoretical guarantees and an increasing number of practical applications for large-scale data analysis problems. In this paper, we analyze a class of online learning algorithms based on fixed potentials and nonlinearized losses, which yields algorithms with implicit update rules. We show how to efficiently compute these updates, and we prove regret bounds for the algorithms. We apply our formulation to several special cases where our approach has benefits over existing online learning methods. In particular, we provide improved algorithms and bounds for the online metric learning problem, and show improved robustness for online linear prediction problems. Results over a variety of data sets demonstrate the advantages of our framework.

1. Introduction

In its most general form, online convex programming (Cesa-Bianchi & Lugosi, 2006; Zinkevich, 2003) can be described as follows: a player chooses some point $\boldsymbol{w}_t$ from a convex set at each iteration. A convex loss function ℓ_t is revealed, and the player pays loss $\ell_t(\boldsymbol{w}_t)$. This is repeated for T timesteps, resuling in a total loss of $\sum_t \ell_t(\boldsymbol{w}_t)$. Examples of problems that fall within this framework include online portfolio management, online linear regression, online classification, and many others (see Cesa-Bianchi & Lugosi (2006) for an overview). Much of the recent success in online learning has been in developing algorithms that are provably competitive with offline counterparts. Typically, we measure the quality of an online learning method in terms of the *regret*: the difference between the sum of the losses $\sum_t \ell_t(\boldsymbol{w}_t)$ and the sum of the losses of the best fixed predictor $\boldsymbol{w}_*$. Regret of $O(\sqrt{T})$ has been shown for general online convex programming and $O(\log T)$ for some special cases.

Online learning updates are often motivated as balancing a tradeoff between "conservativeness" and "correctiveness:" when updating from step to step, we do not want to change our model (i.e., our vector $\boldsymbol{w}_t$) very much, and yet we want to minimize the loss. Such a balance can naturally be modeled using a weighted sum of two terms, one for the conservativeness and one for the correctiveness; one can then compute an update as the minimizer of such a function. Despite this motivation, such updates have proven difficult to analyze in general, so alternative approaches have been proposed and analyzed. One approach *linearizes* the losses via a first-order Taylor expansion, which makes the analysis simpler but looser (e.g. Kivinen & Warmuth (1997) and others). A second approach employs an evolving notion of conservativeness, leading to typically expensive updates (e.g. the method of Azoury & Warmuth (2001)).

Our goal in this paper is to analyze updates obtained by the non-linearized loss with a fixed notion of conservativeness (i.e. a fixed potential, or regularizer), leading to algorithms with implicit update rules. We motivate the study of these updates from practical problems, including non-negative linear regression and online metric learning. For example, a popular regularizer for metric learning is the LogDet divergence, and implicit methods involving this regularizer avoid restrictive assumptions and outperform other methods in practice. Implicit updates are further desirable in special cases where closed-form solutions are available, or where the updates can be computed easily; we discuss several such examples in the paper. To our knowledge, implicit updates, while used ubiquitously to motivate online learning, have not been analyzed in general. As we will see, such updates are slightly more computationally expensive to compute than fixed potential methods with linearized losses (since the resulting implicit updates cannot be computed in closed form), but less expensive than time-varying potential methods. To summarize the main contributions of our work:

- We show $O(\sqrt{T})$ regret for arbitrary Bregman diver-

Appearing in *Proceedings of the 27^{th} International Conference on Machine Learning*, Haifa, Israel, 2010.

gence regularizers for the linear prediction problem, comparable to existing bounds.
- We show $O(\log T)$ regret when the loss functions are strongly convex and the potential is $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$, matching existing bounds up to terms independent of T.
- For LogDet online metric learning, our analysis yields $O(\sqrt{T})$ regret under a variety of constraints; previous bounds for the LogDet regularizer were not $O(\sqrt{T})$.
- We show results indicating that the implicit updates are desirable for linear regression and non-negative linear regression problems.

2. Online Convex Programming: Background and Related Work

We begin with a discussion on the necessary mathematical background and related work for online convex programming (OCP).

2.1. Mathematical Background

Recall that a function f is *convex* if, for any two points $\boldsymbol{x}$ and $\boldsymbol{y}$ in its domain and $\beta \in (0, 1)$:

$$f(\beta\boldsymbol{x} + (1-\beta)\boldsymbol{y}) \leq \beta f(\boldsymbol{x}) + (1-\beta)f(\boldsymbol{y}).$$

The function is strictly convex if the inequality holds strictly. The convex conjugate f^* of a convex function f on a convex domain S is defined as

$$f^*(\boldsymbol{x}^*) = \sup_{\boldsymbol{x}\in S} \left(\langle \boldsymbol{x}^*, \boldsymbol{x}\rangle - f(\boldsymbol{x})\right).$$

We consider functions of *Legendre type* (see Rockafellar (1997)), which implies that the gradient map ∇f is defined on int dom$f$ and is an isomorphism between int dom$f$ and int domf^*. For a function ϕ of Legendre type, denote the *Bregman divergence* with respect to ϕ as

$$D_\phi(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}) - \phi(\boldsymbol{y}) - (\boldsymbol{x}-\boldsymbol{y})^T \nabla\phi(\boldsymbol{y}).$$

Examples of Bregman divergences include the squared Euclidean distance $\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2$, arising from the function $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$. Other examples include the relative entropy (or KL-divergence) arising from $\phi(\boldsymbol{x}) = \sum_i (\boldsymbol{x}(i)\log\boldsymbol{x}(i) - \boldsymbol{x}(i))$, or the LogDet divergence, arising in the matrix domain from $\phi(X) = -\log\det X$.

2.2. OCP Formulation

In online convex programming, the player (algorithm) chooses a point from a fixed convex set F, which we will denote as $\boldsymbol{w}_t$, during each timestep t. After making this choice, the convex loss function ℓ_t is revealed and the player pays loss $\ell_t(\boldsymbol{w}_t)$. The goal is to minimize the total loss $\sum_t \ell_t(\boldsymbol{w}_t)$. We compare against the loss of the best fixed strategy. We denote the regret as

$$R_T = \sum_{t=1}^{T} \ell_t(\boldsymbol{w}_t) - \min_{\boldsymbol{w}\in F} \sum_{t=1}^{T} \ell_t(\boldsymbol{w}).$$

Typically, we denote the optimal fixed strategy from F as $\boldsymbol{w}_*$, and so $R_T = \sum_t (\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{w}_*))$. Here, T is the total number of time steps.

To illustrate with a simple example, consider online linear prediction. The loss function at each step is given by $\ell_t(\boldsymbol{w}_t) = \frac{1}{2}(\boldsymbol{w}_t^T\boldsymbol{x}_t - y_t)^2$. That is, we may view the algorithm as receiving a data point $\boldsymbol{x}_t$ and response y_t at every iteration, with the goal of finding a good vector of weights to minimize the squared loss between a linear combination of the weights over the data points and the response variables. After paying the loss $\ell_t(\boldsymbol{w}_t)$ at each iteration, the algorithm uses information about $\boldsymbol{w}_t$, $\boldsymbol{x}_t$, and y_t (and potentially information from the previous timesteps) to update the vector of weights to $\boldsymbol{w}_{t+1}$.

2.3. Explicit Updates for OCP

Algorithmically, a standard approach to designing online convex programming algorithms has been to update weights as a tradeoff between conservativeness and correctiveness. Loosely speaking, we do not want to change our vector $\boldsymbol{w}_t$ "too much" at each timestep, while at the same time we want the loss over the updated vector to be small. More formally, we can define the following regularized loss function:

$$f_t(\boldsymbol{w}) = D_\phi(\boldsymbol{w}, \boldsymbol{w}_t) + \eta_t \ell_t(\boldsymbol{w}). \tag{1}$$

The first term (the "conservativeness" term) measures the divergence between $\boldsymbol{w}$ and the current vector $\boldsymbol{w}_t$ via a Bregman divergence, while the second term gives the loss incurred by $\boldsymbol{w}$ (and measures "correctiveness"). η_t is called the *learning rate*, and governs the tradeoff between these two terms. The update for computing $\boldsymbol{w}_{t+1}$ is then defined by the minimizer of (1) projected onto F, that is,

$$\begin{aligned} \tilde{\boldsymbol{w}}_{t+1} &= \text{argmin}_{\boldsymbol{w}\in\text{dom}\phi} f_t(\boldsymbol{w}) \\ \boldsymbol{w}_{t+1} &= \text{argmin}_{\boldsymbol{w}\in F} D_\phi(\boldsymbol{w}, \tilde{\boldsymbol{w}}_{t+1}) \end{aligned} \tag{2}$$

It has proven difficult to analyze such updates in general (see, e.g., the discussion in Kivinen & Warmuth (1997)); as a result, two related approaches have been proposed. The first assumes that the loss functions ℓ_t are *linear*, and if they are not, the loss functions are linearized via the first-order Taylor approximation about $\boldsymbol{w}_t$. In particular, we can approximate $\ell_t(\boldsymbol{w})$ as

$$\ell_t(\boldsymbol{w}) \approx \ell_t(\boldsymbol{w}_t) + \nabla\ell_t(\boldsymbol{w}_t)(\boldsymbol{w} - \boldsymbol{w}_t),$$

and since the regret with the original loss functions is less than or equal to the regret with the approximated loss functions (due to convexity of the loss), we can bound the regret with the approximate functions to obtain a bound on

the regret with the original loss functions. See Kivinen & Warmuth (1997); Zinkevich (2003); Hazan et al. (2007) for examples of this approach.

Assuming linear loss functions in this manner greatly simplifies both the mechanics of the algorithm, and its analysis. The updates can generally be computed in closed form by computing the gradient of f_t with the linearized loss functions and solving for $\boldsymbol{w}$. Noting that $\nabla_{\boldsymbol{w}} D_\phi(\boldsymbol{w}, \boldsymbol{w}_t)$

$$\begin{aligned} &= \nabla_{\boldsymbol{w}}(\phi(\boldsymbol{w}) - \phi(\boldsymbol{w}_t) - (\boldsymbol{w} - \boldsymbol{w}_t)^T \nabla\phi(\boldsymbol{w}_t)) \\ &= \nabla\phi(\boldsymbol{w}) - \nabla\phi(\boldsymbol{w}_t), \end{aligned}$$

and $\nabla\ell_t(\boldsymbol{w}) = \nabla\ell_t(\boldsymbol{w}_t)$ for linearized losses, we can solve for $\tilde{\boldsymbol{w}}_{t+1}$ by setting the gradient to 0 as

$$\tilde{\boldsymbol{w}}_{t+1} = \nabla\phi^*(\nabla\phi(\boldsymbol{w}_t) - \eta_t \nabla\ell_t(\boldsymbol{w}_t))$$

when ϕ is of Legendre type. Such updates have been analyzed for arbitrary Bregman divergence regularizers (Cesa-Bianchi & Lugosi, 2006). Assuming only that the loss functions are convex, it has been shown that the regret is bounded by $O(\sqrt{T})$. If the loss functions are strongly convex, the regret for $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$ has been shown to be bounded by $O(\log T)$ (Hazan et al., 2007).

Another approach, which we will not consider in detail in this paper, has been to use a *time-varying* potential approach. We define the function f_t as $f_t(\boldsymbol{w}) = D_{\phi_t}(\boldsymbol{w}, \boldsymbol{w}_t) + \eta_t \ell_t(\boldsymbol{w})$, and so the Bregman divergence, our notion of conservativeness, changes from iteration to iteration (typically $\phi_t = \phi_0 + \sum_t \eta_t \ell_t$). Algorithms in this class include the Follow the Regularized Leader technique (e.g., Kakade & Shalev-Shwartz (2008)) and the elliptic potential method (Azoury & Warmuth, 2001). Such methods have shown logarithmic regret for a wider class of loss functions, including exp-concave loss functions. However, the updates for these methods are more expensive (in the vector case, typically $O(d^2)$ as opposed to $O(d)$ where d is the dimensionality of the data), and the regret bounds have a linear dependence on the dimensionality.

3. Implicit Updates for OCP

Our goal in this work is to study the updates arising from solving f_t directly as in (2). As we will see in the next section, because we do not linearize the loss functions, the resulting updates cannot always be computed in closed form, hence we call them *implicit* updates.

3.1. Motivation

Given that there has been a wide body of existing work on online learning, why even bother with studying implicit updates? Below we sketch out a few reasons.

Non-negative Linear Regression. Consider the case of linear prediction ($\ell_t(\boldsymbol{w}_t) = \frac{1}{2}(\boldsymbol{w}_t^T \boldsymbol{x}_t - y_t)^2$) using the Burg entropy function $\phi(\boldsymbol{x}) = -\sum_i \log \boldsymbol{x}(i)$. A related scenario arises in the matrix case for online metric learning (Jain et al., 2009), where $\phi(X) = -\log\det X$ and the loss is $\frac{1}{2}(\text{tr}(W_t^T X_t) - y_t)^2$; such a regularizer has been shown to be useful for learning positive semi-definite matrices due to LogDet's various intrinsic properties. In the vector case, the Bregman divergence corresponding to the Burg entropy is called the Itakura-Saito divergence: $D_\phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_i (\frac{\boldsymbol{x}(i)}{\boldsymbol{y}(i)} - \log\frac{\boldsymbol{x}(i)}{\boldsymbol{y}(i)} - 1)$. It is a useful regularizer for non-negative linear regression, as its domain is limited to positive-valued vectors but, unlike the relative entropy, those vectors are not typically restricted to lie over a unit simplex.

It is straightforward to show that the updates using the linearized loss in the case of Burg entropy regularization are given by

$$\tilde{\boldsymbol{w}}_{t+1}(i) = \frac{\boldsymbol{w}_t(i)}{1 + \eta_t(\boldsymbol{w}_t^T \boldsymbol{x}_t - y_t)\boldsymbol{w}_t(i)\boldsymbol{x}(i)}.$$

As noted in Jain et al. (2009), this update can cause elements of $\tilde{\boldsymbol{w}}_{t+1}$ to be negative (when $\eta_t(\boldsymbol{w}_t^T \boldsymbol{x}_t - y_t)\boldsymbol{w}_t(i)\boldsymbol{x}_t(i) < -1$), which is outside of the domain of ϕ. Hence, the analysis breaks down, and restrictive assumptions must be placed over the domain of the inputs and/or the step sizes. In contrast, Jain et al. (2009) derived an implicit update in closed form in the matrix domain (i.e. $\phi(X) = -\log\det X$) for a class of losses that arise in online metric learning. Unlike the linearized case, updates using non-linearized losses always remain in the domain of ϕ, making analysis and computation simpler. Jain et al. (2009) proved a simple regret bound for this case, but it is not $O(\sqrt{T})$; we will refine and generalize their approach to a larger class of online LogDet metric learning algorithms with $O(\sqrt{T})$ regret.

Closed-Form Special Cases. Applying an update with a non-linearized loss is clearly desirable in cases where a closed-form update is possible since the inherent approximation introduced by the linearized loss is removed, without any additional cost for the update. In addition to the case noted above, one can also perform implicit gradient descent updates in closed form. Take the example with $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$, $\ell_t(\boldsymbol{w}_t) = \frac{1}{2}(\boldsymbol{w}_t^T \boldsymbol{x}_t - y_t)^2$, and $F = \mathcal{R}^d$. The update using the linearized loss is the standard gradient descent update:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t(\boldsymbol{w}_t^T \boldsymbol{x}_t - y_t)\boldsymbol{x}_t.$$

Updating using the original loss ℓ_t and solving (2) results in the closed-form update

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{\eta_t}{1 + \eta_t\|\boldsymbol{x}_t\|_2^2}(\boldsymbol{w}_t^T \boldsymbol{x}_t - y_t)\boldsymbol{x}_t.$$

The updates are nearly identical, except that the implicit update can be viewed as using a modified learning rate

$\eta_t/(1+\eta_t\|\boldsymbol{x}_t\|_2^2)$. Terms of this form will appear in our analysis. Other cases when $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$ also yield closed-form updates.

Robustness. Empirically, implicit updates outperform or nearly match the performance of explicit updates in general. Furthermore, the implicit methods appear to be more robust to scaling of the data. Using the gradient descent example from above, the update naturally factors in the scale of the input vector $\boldsymbol{x}_t$ when computing the update, which may help explain such robustness. We will show some examples of this phenomenon in our experiments.

3.2. Computing the Updates

The updates for the implicit online algorithm are obtained by solving (2) directly. Using the definition of the Bregman divergence, the gradient of f_t with respect to $\boldsymbol{w}$ simplifies to

$$\nabla\phi(\boldsymbol{w}) - \nabla\phi(\boldsymbol{w}_t) + \eta_t\nabla\ell_t(\boldsymbol{w}),$$

and so setting this to zero to compute $\tilde{\boldsymbol{w}}_{t+1}$ yields the expression $\nabla\phi(\tilde{\boldsymbol{w}}_{t+1}) = \nabla\phi(\boldsymbol{w}_t) - \eta_t\nabla\ell_t(\tilde{\boldsymbol{w}}_{t+1})$. Since ϕ is of Legendre type, $\tilde{\boldsymbol{w}}_{t+1} = \nabla\phi^*(\nabla\phi(\boldsymbol{w}_t) - \eta_t\nabla\ell_t(\tilde{\boldsymbol{w}}_{t+1}))$ (Rockafellar, 1997). This is an *implicit* update since $\tilde{\boldsymbol{w}}_{t+1}$ appears on both sides. For cases when the update cannot be computed in closed form, a root finding method must be employed, but in many practical cases, fast root finding can be performed with only a few functions evaluations. For example, in the case of linear prediction ($\ell_t(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{w}^T\boldsymbol{x}_t - y_t)^2$), we can use a root finder to solve for the inner product $\bar{y}_t = \tilde{\boldsymbol{w}}_{t+1}^T\boldsymbol{x}_t$, resulting in the following equation:

$$\nabla\phi^*(\nabla\phi(\boldsymbol{w}_t) - \eta_t(\bar{y}_t - y_t)\boldsymbol{x}_t)^T\boldsymbol{x}_t - \bar{y}_t = 0.$$

The resulting computation is similar to computing a Bregman projection (Censor & Zenios, 1997). Typically, good implementations require only a few choices for $\bar{y}_t$ (using a bisection method or more sophisticated approach); for example, for the relative entropy and von Neumann divergence, one can adapt the root finding method discussed in Kulis et al. (2009) for computing Bregman projections. In these cases, the typical running time of the algorithm is maintained at $O(d)$, making the updates competitive with the explicit updates.

3.3. A General Stepwise Lemma and Regret Bound

Let us bound the regret of our implicit online algorithm by analyzing how the online solution compares with the optimal solution from step to step.

Lemma 3.1. ***[Stepwise Lemma]*** *Using updates defined by* (1) *and* (2)*, at each step t,*

$$\alpha_t\eta_t\ell_t(\boldsymbol{w}_t) - \eta_t\ell_t(\boldsymbol{w}_*) \leq D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_{t+1}),$$

if α_t satisfies $\alpha_t \leq \frac{f_t(\tilde{\boldsymbol{w}}_{t+1})}{f_t(\boldsymbol{w}_t)}$ and $\boldsymbol{w}_*$ is the optimal offline solution.

Proof. We will prove that the stepwise lemma holds with $\tilde{\boldsymbol{w}}_{t+1}$ in place of $\boldsymbol{w}_{t+1}$. Then, noting that $D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_{t+1}) = [D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \tilde{\boldsymbol{w}}_{t+1})] + [D_\phi(\boldsymbol{w}_*, \tilde{\boldsymbol{w}}_{t+1}) - D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_{t+1})] \geq D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \tilde{\boldsymbol{w}}_{t+1}) + D_\phi(\boldsymbol{w}_{t+1}, \tilde{\boldsymbol{w}}_{t+1}) \geq D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \tilde{\boldsymbol{w}}_{t+1})$, where the first inequality follows by the generalized Pythagorous inequality (see Censor & Zenios (1997), Thm. 2.4.1), the stepwise lemma will hold for $\boldsymbol{w}_{t+1}$ as well.

Using the definition of Bregman divergences and the fact that $\nabla\phi(\tilde{\boldsymbol{w}}_{t+1}) = \nabla\phi(\boldsymbol{w}_t) - \eta_t\nabla\ell_t(\tilde{\boldsymbol{w}}_{t+1})$, straightforward algebra verifies that

$$\begin{aligned} & D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \tilde{\boldsymbol{w}}_{t+1}) \\ = \; & D_\phi(\tilde{\boldsymbol{w}}_{t+1}, \boldsymbol{w}_t) + \eta_t\nabla\ell_t(\tilde{\boldsymbol{w}}_{t+1})^T(\tilde{\boldsymbol{w}}_{t+1} - \boldsymbol{w}_*). \end{aligned}$$

Therefore, we will prove the lemma by showing that for appropriate α_t, the following holds:

$$\begin{aligned} & \alpha_t\eta_t\ell_t(\boldsymbol{w}_t) - \eta_t\ell_t(\boldsymbol{w}_*) - D_\phi(\tilde{\boldsymbol{w}}_{t+1}, \boldsymbol{w}_t) \\ & -\eta_t\nabla\ell_t(\tilde{\boldsymbol{w}}_{t+1})^T(\tilde{\boldsymbol{w}}_{t+1} - \boldsymbol{w}_*) \leq 0. \end{aligned}$$

The convexity of ℓ_t implies the following:

$$\ell_t(\boldsymbol{w}_*) \geq \ell_t(\tilde{\boldsymbol{w}}_{t+1}) + (\boldsymbol{w}_* - \tilde{\boldsymbol{w}}_{t+1})^T\nabla\ell_t(\tilde{\boldsymbol{w}}_{t+1}). \quad (3)$$

We rearrange and plug into the above equation (after multiplying by η_t), and from this we know that the lemma is true if

$$\alpha_t\eta_t\ell_t(\boldsymbol{w}_t) - \eta_t\ell_t(\tilde{\boldsymbol{w}}_{t+1}) - D_\phi(\tilde{\boldsymbol{w}}_{t+1}, \boldsymbol{w}_t) \leq 0, \quad (4)$$

or equivalently, $\alpha_t f_t(\boldsymbol{w}_t) - f_t(\tilde{\boldsymbol{w}}_{t+1}) \leq 0$. The lemma follows.

□

We can use this lemma to prove the following regret bound. It has two pieces. The second, the almost-telescoping sum of divergences, is standard. The first is more unusual: it is small whenever the relative drop in value of the criterion f_t after the implicit update is not too large. In the next section, we shall see that the following theorem implies $O(\sqrt{T})$ and $O(\log T)$ regret for several specific cases.

Theorem 3.2. *Let $R_T = \sum_t \ell_t(\boldsymbol{w}_t) - \sum_t \ell_t(\boldsymbol{w}_*)$ be the regret of the implicit online learning algorithm. Then, for $\alpha_t \leq f_t(\tilde{\boldsymbol{w}}_{t+1})/f_t(\boldsymbol{w}_t)$, we have $R_T \leq$*

$$\sum_t \frac{1}{\eta_t}\Big((1-\alpha_t)\eta_t\ell_t(\boldsymbol{w}_t) + D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_{t+1})\Big).$$

Proof. We add $(1-\alpha_t)\eta_t\ell_t(\boldsymbol{w}_t)$ to both sides of the stepwise lemma, and then divide the inequality by η_t. Finally, sum over all T timesteps. □

4. Special Cases

In the following, we will look in depth at some special cases of interest—the case of linear prediction, and the case when the loss functions are strongly convex. For linear prediction, we give a general result over all Bregman divergence regularizers that is comparable to known regret bounds, and discuss specific examples in detail. We show for strongly convex loss functions how to achieve logarithmic regret, nearly matching existing bounds for this case.

4.1. Linear Prediction

For linear prediction, our loss function is given by $\ell_t(\boldsymbol{w}_t) = \frac{1}{2}(\boldsymbol{w}_t^T\boldsymbol{x}_t - y_t)^2$. Note that $\nabla^2\ell_t(\boldsymbol{w}_t) = \boldsymbol{x}_t\boldsymbol{x}_t^T$, and is not strongly convex.

First we simplify the expression for α_t:

Lemma 4.1. *For $\ell_t(\boldsymbol{w}_t) = \frac{1}{2}(\boldsymbol{w}_t^T\boldsymbol{x}_t - y_t)^2$, and regularization with an arbitrary strictly convex function ϕ of Legendre type, choosing*

$$\alpha_t = \frac{1}{1+\eta_t\gamma_t}$$

satisfies the conditions of Lemma 3.1, where $\gamma_t = \boldsymbol{x}_t^T[\nabla^2\phi(\tilde{\boldsymbol{w}}_{t+1})]^{-1}\boldsymbol{x}_t$.

The proof appears in the appendix. To prove a regret bound, we sum up the stepwise lemma over all T timesteps using the above choice of α_t, and note the telescoping sum. The proof is similar to Zinkevich (2003) but is included here to highlight differences in the bounds. For notational simplicity, let $\|\boldsymbol{x}\|_{S_t}^2 = \boldsymbol{x}^T S_t \boldsymbol{x}$, where $S_t = [\nabla^2\phi(\tilde{\boldsymbol{w}}_{t+1})]^{-1}$.

Theorem 4.2. *Let $R_T = \sum_{t=1}^T \ell_t(\boldsymbol{w}_t) - \sum_{t=1}^T \ell_t(\boldsymbol{w}_*)$ be the regret of the implicit online learning algorithm with a strictly convex function ϕ of Legendre type. Suppose there are constants G and D such that for all $\boldsymbol{w}_t$ and S_t, $\|\nabla\ell_t(\boldsymbol{w}_t)\|_{S_t}^2 \le G$ and $D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) \le D$. Then, choosing $\eta_t = O(t^{-1/2})$, we have $R_T = O(\sqrt{T})$.*

Proof. We take the stepwise lemma result, and add $(1-\alpha_t)\eta_t\ell_t(\boldsymbol{w}_t)$ to both sides, then divide by η_t. This yields

$$\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{w}_*) \le \frac{\eta_t\gamma_t}{1+\eta_t\gamma_t}\ell_t(\boldsymbol{w}_t) + \frac{1}{\eta_t}(D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_{t+1})).$$

Noting that $(\eta_t\gamma_t)/(1+\eta_t\gamma_t) \le \eta_t\gamma_t$, we have $\ell_t(\boldsymbol{w}_t) - \ell_t(\boldsymbol{w}_*)$

$$\begin{aligned} &\le \eta_t\gamma_t\ell_t(\boldsymbol{w}_t) + \frac{1}{\eta_t}(D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_{t+1})) \\ &= \frac{\eta_t}{2}\|\nabla\ell_t(\boldsymbol{w}_t)\|_{S_t}^2 + \frac{1}{\eta_t}(D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_{t+1})), \end{aligned}$$

and summing over all timesteps yields R_T

$$\begin{aligned} &\le \sum_t \frac{\eta_t}{2}\|\nabla\ell_t(\boldsymbol{w}_t)\|_{S_t}^2 \\ &\quad + \sum_t \frac{1}{\eta_t}(D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_{t+1})) \\ &\le \frac{G}{2}\sum_t \eta_t + \sum_t \frac{1}{\eta_t}(D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_{t+1})). \end{aligned}$$

Setting $\eta_t = \sqrt{D/(tG)}$, we can simplify the second summation to $\sqrt{GDT}$ since the sum telescopes. The first sum simplifies using $\sum_{t=1}^T \frac{1}{\sqrt{t}} \le 2\sqrt{T} - 1$ to obtain the result $R_T \le 2\sqrt{DGT}$. □

Note that the simpler $\eta_t = t^{-1/2}$ (as in Zinkevich (2003)) also yields $O(\sqrt{T})$ regret, but the result above is tighter. Moreover, we can improve the analysis by not simplifying via $(\eta_t\gamma_t)/(1+\eta_t\gamma_t) \le \eta_t\gamma_t$ as is done in the proof. In this case, we achieve regret of the form $\sqrt{T} - \log T$ when choosing $\eta_t = O(t^{-1/2})$.

4.2. Linear Prediction Examples

We discuss three particular special cases in this section: the squared Euclidean distance, the relative entropy, and the Itakura-Saito divergence. Since γ_t is indirectly a function of η_t, and therefore t, we also show bounds on γ_t for each of these cases (alternatively, we can say that γ_t is the maximum over all vectors $\boldsymbol{x}, \boldsymbol{w}$ of $\boldsymbol{x}^T[\nabla^2\phi(\boldsymbol{w})]^{-1}\boldsymbol{x}$ in the domain of ϕ, but this may be a weak estimate of γ_t).

Squared Euclidean Distance: The squared Euclidean distance is generated by the function $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$; this divergence is particularly desirable because the exact gradient update can be computed in closed form for linear prediction. Furthermore, we have $\nabla\phi(\boldsymbol{x}) = \boldsymbol{x}$, and $\nabla^2\phi = I$. Therefore, $\gamma_t = \|\boldsymbol{x}_t\|_2^2$. Given that $\gamma_t = \|\boldsymbol{x}_t\|_2^2$, we have $\gamma_t\ell_t(\boldsymbol{w}_t) = \frac{1}{2}\|\nabla\ell_t(\boldsymbol{w}_t)\|_2^2$.

Relative Entropy: Now consider the Bregman divergence obtained by the generating function $\phi(\boldsymbol{x}) = \sum_i \boldsymbol{x}(i)\log\boldsymbol{x}(i)$. In this case, $\gamma_t = \sum_i \tilde{\boldsymbol{w}}_{t+1}(i)\boldsymbol{x}_t(i)^2 \le 3\|\boldsymbol{x}_t\|_\infty^2$. The equality follows by noting that $\nabla^2\phi(\tilde{\boldsymbol{w}}_{t+1}) = \mathrm{diag}(1/\tilde{\boldsymbol{w}}_{t+1})$, where the division is elementwise. The proof of the inequality is omitted due to lack of space.

Itakura-Saito and LogDet: A third special case arises in the case of non-negative linear regression. In the vector case, the potential is called the Burg entropy and is defined by $\phi(\boldsymbol{x}) = -\sum_i \log\boldsymbol{x}(i)$; in the matrix case, the potential is defined as $\phi(X) = -\log\det X$. It is straightforward to show that $\gamma_t = \sum_i \tilde{\boldsymbol{w}}_{t+1}(i)^2\boldsymbol{x}_t(i)^2$ for the vector case, and $\gamma_t = \mathrm{tr}((X_t\tilde{W}_{t+1})^2)$ for the matrix case.

To bound γ_t for the vector case, we let F be the set of vectors whose elements are less than or equal to 1. Let R be the maximum value of $\|\boldsymbol{x}_t\|_1$; we will also assume that $y_t \in [-R, R]$ (since $\boldsymbol{w}_*^T \boldsymbol{x}_t \in [-R, R]$ by Hölder's inequality). The projection step onto F guarantees that $\hat{y}_t = \boldsymbol{w}_t^T \boldsymbol{x}_t \in [-R, R]$ for all t. Since $|\bar{y}_t - y_t| \leq |\hat{y}_t - y_t|$ (as the $\tilde{\boldsymbol{w}}_{t+1}$ update minimizes (2)), we can further conclude that $\bar{y}_t \in [-3R, 3R]$. Finally, let $\boldsymbol{v}(i) = \tilde{\boldsymbol{w}}_{t+1}(i)\boldsymbol{x}_t(i)$. Then the following holds:

$$\gamma_t = \|\boldsymbol{v}\|_2^2 \leq \|\boldsymbol{v}\|_1^2 \leq 9R^2.$$

Thus, $\gamma_t \leq 9R^2$. An analogous result holds for the matrix case.

4.3. Logarithmic Regret with Strongly Convex Losses

A second class of loss functions that have been widely studied assumes strong convexity of the loss; that is $\nabla^2 \ell_t \succeq HI$, for some scalar H. An example would be the loss function $\ell_t(\boldsymbol{w}_t) = \frac{1}{2}\|\boldsymbol{w}_t - \boldsymbol{x}_t\|^2$, whose second derivative is simply the identity (i.e., $H = 1$).

Strongly convex losses are useful and widely studied because they have been shown to yield online learning algorithms with logarithmic regret bounds (e.g. Hazan et al. (2007); Kakade & Shalev-Shwartz (2008)). Here we will show that when choosing $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$, implicit updates achieve logarithmic regret for any strongly convex loss function. Our resulting bounds will be comparable to existing bounds; the first step is to prove a stronger version of the stepwise lemma, stated below.

Lemma 4.3. ***[Stepwise Lemma for H-Strongly Convex Losses]*** *At each step t,*

$$\alpha_t \eta_t \ell_t(\boldsymbol{w}_t) - \eta_t \ell_t(\boldsymbol{w}_*) \leq$$
$$D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_t) - D_\phi(\boldsymbol{w}_*, \boldsymbol{w}_{t+1}) - \frac{\eta_t H}{2}\|\boldsymbol{w}_* - \boldsymbol{w}_{t+1}\|_2^2,$$

where $\alpha_t \leq \frac{f_t(\tilde{\boldsymbol{w}}_{t+1})}{f_t(\boldsymbol{w}_t)}$ and $\boldsymbol{w}_*$ is the optimal offline solution.

Proof. The proof follows the stepwise lemma proof, except in place of (3), we use the following inequality arising from strong convexity (see Boyd & Vandenberghe (2004), Section 9.1.2):

$$\ell_t(\boldsymbol{w}_*) \geq \ell_t(\tilde{\boldsymbol{w}}_{t+1}) + (\boldsymbol{w}_* - \tilde{\boldsymbol{w}}_{t+1})^T \nabla \ell_t(\tilde{\boldsymbol{w}}_{t+1}) + \frac{H}{2}\|\boldsymbol{w}_* - \tilde{\boldsymbol{w}}_{t+1}\|_2^2.$$

We also use the generalized Pythagorous theorem for the additional $\frac{\eta_t H}{2}\|\boldsymbol{w}_* - \boldsymbol{w}_{t+1}\|_2^2$ term, analogous to the stepwise lemma. The remainder of the proof is the same. □

Next, we bound the difference in f_t obtained from updating $\boldsymbol{w}_t$ to $\boldsymbol{w}_{t+1}$ using an *upper bound* on the Hessian of ℓ_t:

Lemma 4.4. *For $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$ and ℓ_t such that $hI \succeq \nabla^2 \ell_t$ is an upper bound on the Hessian of ℓ_t, the following holds:*

$$f_t(\boldsymbol{w}_t) - f_t(\tilde{\boldsymbol{w}}_{t+1}) \leq \frac{1 + \eta_t h}{2}\|\tilde{\boldsymbol{w}}_{t+1} - \boldsymbol{w}_t\|_2^2.$$

See the proof in the appendix. The stepwise lemma ensures that the second term in the regret cancels when choosing $\eta_t = 1/(Ht)$, and the lemma above ensures that the first term of the regret is bounded logarithmically.

Theorem 4.5. *Let $R_T = \sum_t \ell_t(\boldsymbol{w}_t) - \sum_t \ell_t(\boldsymbol{w}_*)$ be the regret of the implicit online learning algorithm. Given that $\phi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$ and ℓ_t is H-strongly convex, then choosing $\eta_t = 1/(Ht)$ yields $R_T = O(\log T)$ when running for a total of T timesteps.*

The proof appears in the appendix. Comparing with the existing proof for explicit updates with strongly convex functions of Hazan et al. (2007), we see that the existing bound is $R_T \leq \frac{G}{2H}(1 + \log T)$, where $\|\nabla \ell_t\|^2 \leq G$. Our bound is identical for the terms dependent on T, but has an additional additive term dependent on the maximum condition number of the Hessian of ℓ_t.

5. Application: Online Metric Learning

So far, we have seen that implicit online learning updates can be applied in a variety of settings, yielding competitive regret bounds as compared to state-of-the-art explicit methods. To highlight an application of implicit updates for a practical problem, we now focus on a domain where implicit updates are particularly useful—online metric learning. As we discuss below, and as the empirical results indicate, the LogDet divergence as a regularizer has shown superior performance in this domain; further, as discussed in Section 3.1, an implicit update scheme is necessary. Our analysis yields better bounds and a more general algorithm.

The goal in online Mahalanobis metric learning is to learn a positive semi-definite matrix W that paramaterizes the so-called Mahalanobis distance: $d_W(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} - \boldsymbol{y})^T W (\boldsymbol{x} - \boldsymbol{y})$. We will update W in an online manner; each iteration, we receive some information about the desired Mahalanobis distance, which is encoded in the loss function $\ell_t(W_t) = \frac{1}{2}(\text{tr}(W_t X_t) - y_t)^2$. For example, when $X_t = (\boldsymbol{x}_t - \boldsymbol{y}_t)(\boldsymbol{x}_t - \boldsymbol{y}_t)^T$, then $\text{tr}(W_t X_t)$ is simply the Mahalanobis distance between $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ using W_t, and so the loss encodes how close the Mahalanobis distance is to the target distance y_t. Note that in this case, X_t is a rank-one matrix. Another possibility, used frequently in the offline setting, is that of relative distance constraints.

Various regularizers are possible in the matrix domain. Examples include the squared Frobenius norm $D_\phi(X, Y) = \frac{1}{2}\|X - Y\|_F^2$, the von Neumann divergence $D_\phi(X, Y) = \text{tr}(X \log X - X \log Y - X + Y)$, or the LogDet divergence,

discussed earlier. Because the constraints for online metric learning are typically low-rank (i.e., the matrices X_t are low-rank), the implicit LogDet updates can be computed in closed form (see Jain et al. (2009) for the case of similarity and dissimilarity constraints) in $O(d^2)$ time, where d is the number of rows/columns of W_t. The von Neumann divergence updates require an eigendecomposition in practice, resulting in updates that cost $O(d^3)$, and because the W_t matrices are restricted to the positive semi-definite cone, the squared Frobenius norm updates must project onto the PSD cone, which may be potentially expensive. Combined with the fact that LogDet has produced good empirical results for online metric learning, the LogDet divergence is a natural regularizer for the online metric learning problem.

The previous work of Jain et al. (2009) considered only the case of similarity and dissimilarity constraints, and the analysis depended on properties of rank-one matrices. In contrast, the analysis in this paper applies in general to other constraints and regularizers for the online metric learning problem. Furthermore, our bounds for the case considered in Jain et al. (2009) are stronger; their bounds are not of the form $O(\sqrt{T})$, even if the stepsize for η_t is defined to be $O(1/\sqrt{t})$.

6. Experimental Results

Synthetic Results. We begin with some simple synthetic experiments to demonstrate advantages of the implicit approach. First, we compare standard gradient descent with the implicit gradient descent algorithm for linear prediction. An optimal 20-dimensional vector $\boldsymbol{w}_*$ was chosen randomly (with uniform weights in $[0, 1]$). For each timestep, we chose random $\boldsymbol{x}_t$ vectors with weights uniformly random from $[-.5, .5]$ and computed a y_t value as the inner product between the optimal $\boldsymbol{w}_*$ and $\boldsymbol{x}_t$, with added Gaussian noise (variance .1). We compared the methods (using $\eta_t = 1/\sqrt{t}$), and then repeated the experiment by scaling the $\boldsymbol{x}_t$ vectors (and the noise) by 2 and 2.5 (but keeping the same learning rate). The results in the top row of Figure 1 show that as the scale increases, the standard gradient descent makes larger mistakes during early timesteps, whereas the implicit method appears more robust during these timesteps. As a result, the difference in the total accumulated loss between the methods grows with the scale of the data.

Next, we compared exponentiated gradient descent, exponentiated gradient descent with implicit updates, and the Itakura-Saito based descent method. As before, we construct $\boldsymbol{x}_t$ vectors with weights chosen uniformly at random from $[-.5, .5]$. When the optimal weight vector is normalized to sum to 1, we found that the exponentiated gradient descent methods and the Itakura-Saito method all gave nearly identical results (plots not shown). We also tested cases where the optimal weight vector was not normalized to sum to 1. For this case, the loss for the EGD-based methods was always significantly higher due to the restriction that the weights lie on the unit simplex. See the final plot of Figure 1 for an example; note that both EGD-based methods performed similarly in this experiment and it is difficult to distinguish them on the plot.

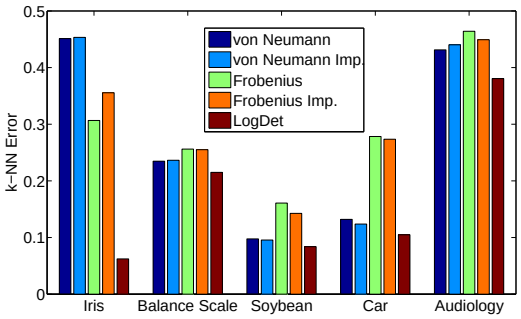

Figure 2. Comparison of metric learning methods on standard UCI data sets.

Online Metric Learning. We applied our method to the online metric learning problem over a set of UCI data sets. We compared various regularizers for metric learning, using similarity and dissimilarity constraints (see Section 5 for a description of the problem). For this experiment, we ran the following methods: a von Neumann regularized explicit method, a von Neumann regularized implicit method, a squared Frobenius regularized explicit method, a squared Frobenius regularized implicit method, and the LogDet regularized implicit method. For each algorithm, we ran 10,000 online timesteps and constraints were generated randomly over a 70 percent training set. Each constraint is a random pair of points from the training set, and target distances are set to the 5th and 95th percentile of the training set for same-class and different-class constraints, respectively. Results are averaged over 10 runs. We see from the results in Figure 2 that the LogDet method clearly outperforms the other online regularizers and algorithms on these data sets. Furthermore, given the additional cost of the von Neumann-based methods (each iteration is $O(d^3)$ whereas the cost of an iteration for the LogDet-based method is $O(d^2)$), the requirement that the Frobenius-based methods project onto the positive semi-definite cone, and the fact that LogDet cannot be effectively analyzed in the explicit case, these results further validate our analysis of implicit methods.

References

Azoury, K. S. and Warmuth, M. Relative loss bounds for on-line density estimation with the exponential family of distributions. *Machine Learning*, 43(3):211–246, 2001.

Boyd, S. and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, 2004.

Censor, Y. and Zenios, S. *Parallel Optimization*. Oxford University Press, 1997.

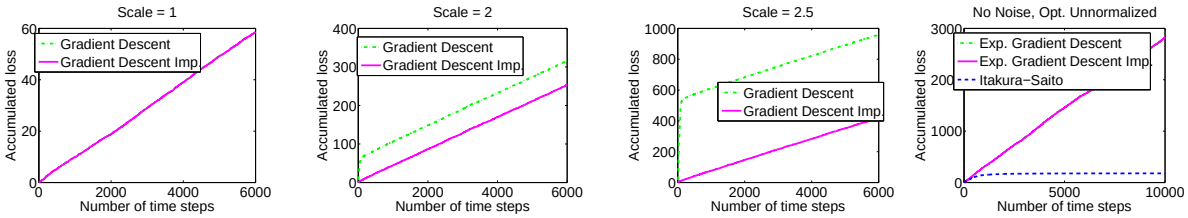

Figure 1. Synthetic results. The first 3 plots show how the implicit gradient descent algorithm is more robust as the scale of the data is increased; the last plot shows that, for non-negative linear regression, the Itakura-Saito methods (which require implicit analysis) are more suited than exponentiated gradient descent for problems where the optimal weight vector is not normalized. See text for details.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

Hazan, E., Agarwal, A., and Kale, S. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69: 169–192, 2007.

Jain, P., Kulis, B., Dhillon, I., and Grauman, K. Online metric learning and fast similarity search. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

Kakade, S. and Shalev-Shwartz, S. Mind the duality gap: Logarithmic regret algorithms for online optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

Kivinen, J. and Warmuth, M. K. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–64, 1997.

Kulis, B., Sustik, M., and Dhillon, I. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research (JMLR)*, 10, 2009.

Rockafellar, R. T. *Convex Analysis*. Princeton University Press, 1997.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning (ICML)*, 2003.

Appendix: Proofs

Proof: [Lemma 4.1] Recall from the stepwise lemma that α_t must satisfy $\alpha_t f_t(\boldsymbol{w}_t) \leq f_t(\tilde{\boldsymbol{w}}_{t+1})$ or, equivalently, $\alpha_t f_t(\boldsymbol{w}_t) - f_t(\tilde{\boldsymbol{w}}_{t+1}) \leq 0$. Define the left-hand side using (4) as a function of y_t; that is $h(y_t) =$

$$\alpha_t \frac{\eta_t}{2}(\boldsymbol{w}_t^T \boldsymbol{x}_t - y_t)^2 - \frac{\eta_t}{2}(\tilde{\boldsymbol{w}}_{t+1}^T \boldsymbol{x}_t - y_t)^2 - D_\phi(\tilde{\boldsymbol{w}}_{t+1}, \boldsymbol{w}_t).$$

We will show that the derivative of h is zero somewhere (and $h = 0$ at that point), and that the second derivative is always negative for a specific choice of α_t, implying that $h \leq 0$ everywhere. Since $\partial f_t / \partial \tilde{\boldsymbol{w}}_{t+1} = 0$, we know from the multivariate chain rule that when taking the derivative of h with respect to y_t, we can treat $\tilde{\boldsymbol{w}}_{t+1}$ as a constant. That leads to $(dh)/(dy_t) = -\alpha_t \eta_t (\hat{y}_t - y_t) + \eta_t(\bar{y}_t - y_t)$, where $\hat{y}_t = \boldsymbol{w}_t^T \boldsymbol{x}_t$ and $\bar{y}_t = \tilde{\boldsymbol{w}}_{t+1}^T \boldsymbol{x}_t$. When $\hat{y}_t = y_t$, then loss is zero and so $\tilde{\boldsymbol{w}}_{t+1} = \boldsymbol{w}_t$, $h = 0$, and $h' = 0$. Now we must compute the second derivative of h with respect to α_t and show that this is always negative. The 2nd derivative is computed as (noting that $\bar{y}_t$ is a function of y_t)

$$\alpha_t \eta_t + \eta_t \left(\frac{d\bar{y}_t}{dy_t} - 1 \right),$$

which results in the condition $\alpha_t \leq 1 - (d\bar{y}_t)/(dy_t)$. From Section 3.2, we know that $\tilde{\boldsymbol{w}}_{t+1} = \nabla\phi^*(\nabla\phi(\boldsymbol{w}_t) - \eta(\tilde{\boldsymbol{w}}_{t+1}^T \boldsymbol{x}_t - y_t)\boldsymbol{x}_t)$ and so therefore since $\bar{y}_t = \tilde{\boldsymbol{w}}_{t+1}^T \boldsymbol{x}_t$, we have $\bar{y}_t = \nabla\phi^*(\nabla\phi(\boldsymbol{w}_t) - \eta(\bar{y}_t - y_t)\boldsymbol{x}_t)^T \boldsymbol{x}_t$. Simple application of the chain rule yields

$$\frac{d\bar{y}_t}{dy_t} = -\eta_t \boldsymbol{x}_t^T [\nabla^2 \phi^*(\nabla\phi(\boldsymbol{w}_t) - \eta_t(\bar{y}_t - y_t)\boldsymbol{x}_t)]\boldsymbol{x}_t \left(\frac{d\bar{y}_t}{dy_t} - 1 \right).$$

Using $\nabla^2\phi^*(\nabla\phi(\boldsymbol{x})) = [\nabla^2\phi(\boldsymbol{x})]^{-1}$ and $\gamma_t = \boldsymbol{x}_t^T [\nabla^2 \phi(\tilde{\boldsymbol{w}}_{t+1})]^{-1} \boldsymbol{x}_t$, we simplify to

$$\frac{d\bar{y}_t}{dy_t} = \frac{\eta_t \gamma_t}{1 + \eta_t \gamma_t},$$

and simplification of α_t yields the lemma.

Proof: [Lemma 4.4] An upper bound of hI on $\nabla^2 \ell_t$ (Boyd & Vandenberghe (2004), Section 9.1.2) yields $\ell_t(\boldsymbol{w}_t) \leq$

$$\ell_t(\tilde{\boldsymbol{w}}_{t+1}) + (\boldsymbol{w}_t - \tilde{\boldsymbol{w}}_{t+1})^T \nabla \ell_t(\tilde{\boldsymbol{w}}_{t+1}) + \frac{h}{2}\|\boldsymbol{w}_t - \tilde{\boldsymbol{w}}_{t+1}\|_2^2.$$

Since $\tilde{\boldsymbol{w}}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla \ell_t(\tilde{\boldsymbol{w}}_{t+1})$, the term $(\boldsymbol{w}_t - \tilde{\boldsymbol{w}}_{t+1})^T \nabla \ell_t(\tilde{\boldsymbol{w}}_{t+1})$ simplifies to $\frac{1}{\eta_t}\|\tilde{\boldsymbol{w}}_{t+1} - \boldsymbol{w}_t\|_2^2$. Multiplying the whole equation by η_t and simplifying via $f_t(\tilde{\boldsymbol{w}}_{t+1}) = \frac{1}{2}\|\tilde{\boldsymbol{w}}_{t+1} - \boldsymbol{w}_t\|_2^2 + \eta_t \ell_t(\tilde{\boldsymbol{w}}_{t+1})$, the result follows.

Proof: [Theorem 4.5] By summing over all T timesteps as in Theorem 3.2, using Lemma 4.3 we obtain

$$R_T \leq \sum_t \frac{1}{\eta_t}(f_t(\boldsymbol{w}_t) - f_t(\tilde{\boldsymbol{w}}_{t+1})) +$$

$$\sum_t \frac{1}{2\eta_t}(\|\boldsymbol{w}_* - \boldsymbol{w}_t\|_2^2 - \|\boldsymbol{w}_* - \boldsymbol{w}_{t+1}\|_2^2 - \eta_t H \|\boldsymbol{w}_* - \boldsymbol{w}_{t+1}\|_2^2).$$

By choosing $\eta_t = 1/(Ht)$, the second sum simplifies to $\frac{1}{2\eta_1}\|\boldsymbol{w}_* - \boldsymbol{w}_1\|_2^2$. For the first sum, we use Lemma 4.4 and the fact that $\|\tilde{\boldsymbol{w}}_{t+1} - \boldsymbol{w}_t\|_2^2 = \eta_t^2 \|\nabla \ell_t(\tilde{\boldsymbol{w}}_{t+1})\|_2^2 \leq \eta_t^2 G$, where $\|\nabla \ell_t\|^2 \leq G$. This gives us

$$\begin{aligned} R_T &\leq G \sum_t \frac{\eta_t + \eta_t^2 h}{2} + \frac{HD}{2} \\ &\leq \frac{G}{2H}\left(1 + \log T\right) + \frac{Gh}{H^2} + \frac{HD}{2}. \end{aligned}$$