Follow-the-Regularized-Leader and Mirror Descent: Equivalence Theorems and L1 Regularization

H. Brendan McMahan
Google, Inc.

Abstract

We prove that many mirror descent algorithms for online convex optimization (such as online gradient descent) have an equivalent interpretation as follow-the-regularized-leader (FTRL) algorithms. This observation makes the relationships between many commonly used algorithms explicit, and provides theoretical insight on previous experimental observations. In particular, even though the FOBOS composite mirror descent algorithm handles L_1 regularization explicitly, it has been observed that the FTRL-style Regularized Dual Averaging (RDA) algorithm is even more effective at producing sparsity. Our results demonstrate that the key difference between these algorithms is how they handle the cumulative L_1 penalty. While FOBOS handles the L_1 term exactly on any given update, we show that it is effectively using subgradient approximations to the L_1 penalty from previous rounds, leading to less sparsity than RDA, which handles the cumulative penalty in closed form. The FTRL-Proximal algorithm, which we introduce, can be seen as a hybrid of these two algorithms, and significantly outperforms both on a large, real-world dataset.

1 INTRODUCTION

We consider the problem of online convex optimization and its application to online learning. On each round $t = 1, \ldots, T$, we pick a point $x_t \in \mathbb{R}^n$. A convex loss function f_t is then revealed, and we incur loss $f_t(x_t)$.

In this work, we investigate the relationship between two of the most important and successful families of low-regret algorithms for online convex optimization. On the surface, follow-the-regularized-leader algorithms like Regularized Dual Averaging [Xiao, 2009] appear quite different from gradient descent (and more generally, mirror descent) style algorithms like FOBOS [Duchi and Singer, 2009]. However, here we show that in the case of quadratic stabilizing regularization there are essentially only two differences between the algorithms:

- How they choose to center the additional strong convexity used to guarantee low regret: RDA centers this regularization at the origin, while FOBOS centers it at the current feasible point.
- How they handle an arbitrary non-smooth regularization function Ψ. This includes the mechanism of projection onto a feasible set and how L_1 regularization is handled.

To make these differences precise while also illustrating that these families are actually closely related, we consider a third algorithm, FTRL-Proximal. When the non-smooth term Ψ is omitted, this algorithm is in fact identical to FOBOS. On the other hand, its update is essentially the same as that of dual averaging, except that additional strong convexity is centered at the current feasible point (see Table 1).

Previous work has shown experimentally that Dual Averaging with L_1 regularization is much more effective at introducing sparsity than FOBOS [Xiao, 2009, Duchi et al., 2010a]. Our equivalence theorems provide a theoretical explanation for this: while RDA considers the cumulative L_1 penalty $t\lambda\|x\|_1$ on round t, FOBOS (when viewed as a global optimization using our equivalence theorem) considers $\phi_{1:t-1} \cdot x + \lambda\|x\|_1$, where ϕ_s is a certain subgradient approximation of $\lambda\|x_s\|_1$ (we use $\phi_{1:t-1}$ as shorthand for $\sum_{s=1}^{t-1} \phi_s$, and extend the notation to sums over matrices and functions as needed).

In Section 2, we consider general formulations of mirror descent and follow-the-regularized-leader, and prove theorems relating the two. In Section 3, we compare FOBOS, RDA, and FTRL-Proximal experimen-

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

Table 1: Summary of algorithms expressed as global optimizations against functions $f_t(x) = \ell_t(x) + \Psi(x)$, where $\Psi(x)$ is an arbitrary and typically non-smooth convex function, for example $\Psi(x) = \lambda\|x\|_1$. Each algorithm's objective has three components: (A) an approximation to $\ell_{1:t}$ based on the gradients $g_t = \nabla\ell_t(x_t)$, (B) terms for the non-smooth portion Ψ (the ϕ_t are certain subgradients of Ψ), and (C) additional strong convexity to stabilize the algorithm, needed to guarantee low regret (the matrices Q_s are generalized learning rates). These four algorithms are the cross product of 2 design decisions: how the Ψ function is handled, and where additional strong convexity is centered. See Section 1 for details and references.

			(A)		(B)	(C)
FOBOS	x_{t+1}	$= \arg\min_x$	$g_{1:t} \cdot x$	$+$	$\phi_{1:t-1} \cdot x + \Psi(x)$	$+\frac{1}{2}\sum_{s=1}^{t} \|Q_s^{\frac{1}{2}}(x - x_s)\|_2^2$
AOGD	x_{t+1}	$= \arg\min_x$	$g_{1:t} \cdot x$	$+$	$\phi_{1:t-1} \cdot x + \Psi(x)$	$+\frac{1}{2}\sum_{s=1}^{t} \|Q_s^{\frac{1}{2}}(x - 0)\|_2^2$
RDA	x_{t+1}	$= \arg\min_x$	$g_{1:t} \cdot x$	$+$	$t\Psi(x)$	$+\frac{1}{2}\sum_{s=1}^{t} \|Q_s^{\frac{1}{2}}(x - 0)\|_2^2$
FTRL-Proximal	x_{t+1}	$= \arg\min_x$	$g_{1:t} \cdot x$	$+$	$t\Psi(x)$	$+\frac{1}{2}\sum_{s=1}^{t} \|Q_s^{\frac{1}{2}}(x - x_s)\|_2^2$

tally. The FTRL-Proximal algorithm behaves very similarly to RDA in terms of sparsity, confirming that it is the cumulative subgradient approximation to the L_1 penalty that causes decreased sparsity in FOBOS.

In recent years, online gradient descent and stochastic gradient descent (its batch analogue) have proven themselves to be excellent algorithms for large-scale machine learning. In the simplest case FTRL-Proximal is identical, but when L_1 or other non-smooth regularization is needed, FTRL-Proximal significantly outperforms FOBOS, and can outperform RDA as well. Since the implementations of FTRL-Proximal and RDA only differ by a few lines of code, we recommend trying both and picking the one with the best performance in practice.

Algorithms We begin by establishing notation and introducing more formally the algorithms we consider. While our theorems apply to more general versions of these algorithms, here we focus on the specific instances we use in our experiments. We consider loss functions $f_t(x) = \ell_t(x) + \Psi(x)$, where Ψ is a fixed (typically non-smooth) regularization function. In a typical online learning setting, given an example (θ_t, y_t) where $\theta_t \in \mathbb{R}^n$ is a feature vector and $y_t \in \{-1, 1\}$ is a label, we take $\ell_t(x) = \text{loss}(\theta_t \cdot x, y_t)$. For example, for logistic regression we use log-loss, $\text{loss}(\theta_t \cdot x, y_t) = \log(1 + \exp(-y_t\theta_t \cdot x))$. We use the standard reduction to linear functions, letting $g_t = \nabla\ell_t(x_t)$. All of the algorithms we consider support composite updates (consideration of Ψ explicitly rather than through a gradient $\nabla f_t(x_t)$) as well as positive semi-definite matrix learning rates Q which can be chosen adaptively (the interpretation of these matrices as learning rates will be clarified in Section 2).

The first algorithm we consider is from the gradient-descent family, namely FOBOS, which plays

$$x_{t+1} = \arg\min_x g_t \cdot x + \lambda\|x\|_1 + \frac{1}{2}\|Q_{1:t}^{\frac{1}{2}}(x - x_t)\|_2^2.$$

We state this algorithm implicitly as an optimization, but a gradient-descent style closed-form update can also be given [Duchi and Singer, 2009]. The algorithm was described in this form as a specific composite-objective mirror descent (COMID) algorithm by Duchi et al. [2010b].

The Regularized Dual Averaging (RDA) algorithm of Xiao [2009] plays

$$x_{t+1} = \arg\min_x g_{1:t} \cdot x + t\lambda\|x\|_1 + \frac{1}{2}\sum_{s=1}^{t} \|Q_s^{\frac{1}{2}}(x - 0)\|_2^2.$$

In contrast to FOBOS, the optimization is over the sum $g_{1:t}$ rather than just the most recent gradient g_t. We will show (in Theorem 4) that when $\lambda = 0$ and the ℓ_t are not strongly convex, this algorithm is in fact equivalent to the Adaptive Online Gradient Descent (AOGD) algorithm [Bartlett et al., 2007].

The FTRL-Proximal algorithm plays

$$x_{t+1} = \arg\min_x g_{1:t} \cdot x + t\lambda\|x\|_1 + \frac{1}{2}\sum_{s=1}^{t} \|Q_s^{\frac{1}{2}}(x - x_s)\|_2^2.$$

This algorithm was introduced in [McMahan and Streeter, 2010], but without support for an explicit Ψ. Regret bounds for the more general algorithm that handles a fixed Ψ function are proved in [McMahan, 2011].

One of our principle contributions is showing the close connection between all four of these algorithms; Table 1 summarizes the key results from Theorems 2 and 4, writing AOGD and FOBOS in a form that makes the relationship to RDA and FTRL-Proximal explicit.

In our analysis, we will consider arbitrary convex functions R_t and $\tilde{R}_t$ in place of the $\frac{1}{2}\|Q_t^{\frac{1}{2}}x\|_2^2$ and $\frac{1}{2}\|Q_t^{\frac{1}{2}}(x-x_t)\|_2^2$ that appear here, as well as arbitrary convex $\Psi(x)$ in place of $\lambda\|x\|_1$. For all these algorithms, the matrices Q_t are chosen adaptively. In the experiments, we use per-coordinate adaptation where the Q_t are diagonal such that $Q_{1:t} = \text{diag}(\bar{\sigma}_{t,1}, \ldots, \bar{\sigma}_{t,n})$ with $\bar{\sigma}_{t,i} = \frac{1}{\gamma}\sqrt{\sum_{s=1}^{t} g_{t,i}^2}$. See McMahan and Streeter [2010] and Duchi et al. [2010a] for details. Since all of the algorithms benefit from this approach, we use the more familiar names of the original algorithms, even though in most cases they were introduced with scalar learning rates. The γ is learning-rate scale parameter, which we tune in experiments.

Efficient Implementations All of these algorithms can be implemented efficiently, in that the update for a g_t with K nonzeros can be done in $\mathcal{O}(K)$ time. Both FTRL-Proximal and RDA can be implemented (for diagonal learning rates) by storing two floating-point values for each attribute, a quadratic term and a linear term. When $x_{t,i}$ is needed, it can be solved for lazily in closed form (see for example [Xiao, 2009]).

For FOBOS, the presence of $\lambda\|x\|_1$ in the update implies all coefficients $x_{t,i}$ needs to be updated even when $g_{t,i} = 0$. However, by storing the index t of the last round on which $g_{t,i}$ was nonzero, the L_1 part of the update can be made lazy [Duchi and Singer, 2009].

Feasible Sets In some applications, we may be restricted to only play points from a restricted convex feasible set $\mathcal{F} \subseteq \mathbb{R}^n$, for example, the set of (fractional) paths between two nodes in a graph. Since all the algorithms we consider support composite updates, this is accomplished for free by choosing Ψ to be the indicator function $\Psi_{\mathcal{F}}$ on $\mathcal{F}$, that is $\Psi_{\mathcal{F}}(x) = 0$ if $x \in \mathcal{F}$, and ∞ otherwise. It is straightforward to verify that $\arg\min_{x\in\mathbb{R}^n} g_{1:t}\cdot x + R_{1:t}(x) + \Psi_{\mathcal{F}}(x)$ is equivalent to $\arg\min_{x\in\mathcal{F}} g_{1:t}\cdot x + R_{1:t}(x)$, and so in this work we can generalize (for example) the results of [McMahan and Streeter, 2010] for specific feasible sets without explicitly discussing $\mathcal{F}$, and instead considering arbitrary extended convex functions Ψ.

Notation and Technical Background We write $x^\top y$ or $x \cdot y$ for the inner product between $x, y \in \mathbb{R}^n$. The ith entry in a vector x is denoted $x_i \in \mathbb{R}$; when we have a sequence of vectors $x_t \in \mathbb{R}^n$ indexed by time, the ith entry is $x_{t,i} \in \mathbb{R}$. For positive semi-definite B, we write $B^{1/2}$ for the square root of B, the unique $X \in S_+^n$ such that $XX = B$, so $\|B^{\frac{1}{2}}x\|_2^2 = x^\top Bx$. Unless otherwise stated, convex functions are assumed to be extended, with domain $\mathbb{R}^n$ and range $\mathbb{R}\cup\{\infty\}$ (see, for example [Boyd and Vandenberghe, 2004, 3.1.2]). For a convex function f, we let $\partial f(x)$ denote the set of subgradients of f at x (the subdifferential of f at x). By definition, $g \in \partial f(x)$ means $f(y) \geq f(x) + g^\top(y-x)$ for all y. When f is differentiable, we write $\triangledown f(x)$ for the gradient of f at x. In this case, $\partial f(x) = \{\triangledown f(x)\}$. All mins and argmins are over $\mathbb{R}^n$ unless otherwise noted. We make frequent use of the following standard results, summarized as follows:

Theorem 1. *Let $R : \mathbb{R}^n \to \mathbb{R}$ be strongly convex with continuous first partial derivatives, and let $\Psi : \mathbb{R}^n \to \mathbb{R}\cup\{\infty\}$ be an arbitrary convex function. Define $g(x) = R(x) + \Psi(x)$. Then, there exists a unique pair (x^*, ϕ^*) such that both*

$$\phi^* \in \partial\Psi(x^*)$$

and

$$x^* = \arg\min_x R(x) + \phi^* \cdot x.$$

Further, this x^* is the unique minimizer of g.

Note that an equivalent condition to $x^* = \arg\min_x R(x) + \phi^* \cdot x$ is $\triangledown R(x^*) + \phi^* = 0$.

Proof. Since R is strongly convex, g is strongly convex, and so has a unique minimizer x^* (see for example, [Boyd and Vandenberghe, 2004, 9.1.2]). Let $r = \triangledown R$. Since x^* is a minimizer of g, there must exist a $\phi^* \in \partial\Psi(x^*)$ such that $r(x^*) + \phi^* = 0$, as this is a necessary (and sufficient) condition for $0 \in \partial g(x^*)$. It follows that $x^* = \arg\min_x R(x) + \phi^* \cdot x$, as $r(x^*) + \phi^*$ is the gradient of this objective at x^*. Suppose some other (x', ϕ') satisfies the conditions of the theorem. Then, $r(x') + \phi' = 0$, and so $0 \in \partial g(x')$, and so x' is a minimizer of g. Since this minimizer is unique, $x' = x^*$, and $\phi' = -r(x^*) = \phi^*$. □

2 MIRROR DESCENT FOLLOWS THE LEADER

In this section we consider the relationship between mirror descent algorithms (the simplest example being online gradient descent) and FTRL algorithms. Let $f_t(x) = g_t \cdot x + \Psi(x)$ where $g_t \in \partial\ell_t(x_t)$. Let R_1 be strongly convex, with all the R_t convex. We assume that $\min_x R_1(x) = 0$, and assume that $x = 0$ is the unique minimizer unless otherwise noted.

Follow The Regularized Leader (FTRL) The simplest follow-the-regularized-leader algorithm plays

$$x_{t+1} = \arg\min_x g_{1:t}\cdot x + \frac{\sigma_{1:t}}{2}\|x\|_2^2. \tag{1}$$

For $t = 1$, we typically take $x_1 = 0$. We can generalize $\frac{1}{2}\|x\|_2^2$ to an arbitrary strongly convex R by:

$$x_{t+1} = \arg\min_x g_{1:t}\cdot x + \sigma_{1:t}R(x) \tag{2}$$

We could choose $\sigma_{1:t}$ independently for each t, but we need $\sigma_{1:t}$ to be non-decreasing in t, and so writing it as a sum of the per-round increments $\sigma_t \geq 0$ is reasonable. The most general update is

$$x_{t+1} = \arg\min_x g_{1:t} \cdot x + R_{1:t}(x). \tag{3}$$

where we add an additional convex function R_t on each round. Letting $R_t(x) = \sigma_t R(x)$ recovers the previous formulation.

When $\arg\min_{x \in \mathbb{R}^n} R_t(x) = 0$, we call the functions R_t (and associated algorithms) *origin-centered*. We can also define *proximal* versions of FTRL[1] that center additional regularization at the current point rather than at the origin. In this section, we write $\tilde{R}_t(x) = R_t(x - x_t)$ and reserve the R_t notation for origin-centered functions. Note that $\tilde{R}_t$ is only needed to select x_{t+1}, and x_t is known to the algorithm at this point, ensuring the algorithm only needs access to the first t loss functions when computing x_{t+1} (as required). The general update is

$$x_{t+1} = \arg\min_x g_{1:t} \cdot x + \tilde{R}_{1:t}(x), \tag{4}$$

In the simplest case, this becomes

$$x_{t+1} = \arg\min_x g_{1:t} \cdot x + \sum_{s=1}^{t} \frac{\sigma_s}{2} \|x - x_s\|_2^2. \tag{5}$$

Mirror Descent The simplest version of mirror descent is gradient descent using a constant step size η, which plays

$$x_{t+1} = x_t - \eta g_t = -\eta g_{1:t}. \tag{6}$$

In order to get low regret, T must be known in advance so η can be chosen accordingly (or a doubling trick can be used). But, since there is a closed-form solution for the point x_{t+1} in terms of $g_{1:t}$ and η, we generalize this to a "revisionist" algorithm that on each round plays the point that gradient descent with constant step size would have played if it had used step size η_t on rounds 1 through $t-1$. That is, $x_{t+1} = -\eta_t g_{1:t}$. When $R_t(x) = \frac{\sigma_t}{2}\|x\|_2^2$ and $\eta_t = \frac{1}{\sigma_{1:t}}$, this is equivalent to the FTRL of Equation (1).

In general, we will be more interested in gradient descent algorithms which use an adaptive step size that depends (at least) on the round t. Using a variable step size η_t on each round, gradient descent plays:

$$x_{t+1} = x_t - \eta_t g_t. \tag{7}$$

An intuition for this update comes from the fact it can be re-written as

$$x_{t+1} = \arg\min_x g_t \cdot x + \frac{1}{2\eta_t}\|x - x_t\|_2^2.$$

This version captures the notion (in online learning terms) that we don't want to change our hypothesis x_t too much (for fear of predicting badly on examples we have already seen), but we do want to move in a direction that decreases the loss of our hypothesis on the most recently seen example (which is approximated by the linear function g_t).

Mirror descent algorithms use this intuition, replacing the L_2-squared penalty with an arbitrary Bregman divergence. For a differentiable, strictly convex R, the corresponding Bregman divergence is

$$\mathcal{B}_R(x, y) = R(x) - \big(R(y) + \nabla R(y) \cdot (x - y)\big)$$

for any $x, y \in \mathbb{R}^n$. We then have the update

$$x_{t+1} = \arg\min_x g_t \cdot x + \frac{1}{\eta_t}\mathcal{B}_R(x, x_t), \tag{8}$$

or explicitly (by setting the gradient of (8) to zero),

$$x_{t+1} = r^{-1}(r(x_t) - \eta_t g_t) \tag{9}$$

where $r = \nabla R$. Letting $R(x) = \frac{1}{2}\|x\|_2^2$ so that $\mathcal{B}_R(x, x_t) = \frac{1}{2}\|x - x_t\|_2^2$ recovers the algorithm of Equation (7). One way to see this is to note that $r(x) = r^{-1}(x) = x$ in this case.

We can generalize this even further by adding a new strongly convex function R_t to the Bregman divergence on each round. Namely, let

$$\mathcal{B}_{1:t}(x, y) = \sum_{s=1}^{t} \mathcal{B}_{R_s}(x, y),$$

so the update becomes

$$x_{t+1} = \arg\min_x g_t \cdot x + \mathcal{B}_{1:t}(x, x_t) \tag{10}$$

or equivalently $x_{t+1} = (r_{1:t})^{-1}(r_{1:t}(x_t) - g_t)$ where $r_{1:t} = \sum_{s=1}^{t} \nabla R_t = \nabla R_{1:t}$ and $(r_{1:t})^{-1}$ is the inverse of $r_{1:t}$. The step size η_t is now encoded implicitly in the choice of R_t.

Composite-objective mirror descent (COMID) [Duchi et al., 2010b] handles Ψ functions[2] as part of the objective on each round: $f_t(x) = g_t \cdot x + \Psi(x)$. Using our notation, the COMID update is

$$x_{t+1} = \arg\min_x \eta g_t \cdot x + \mathcal{B}(x, x_t) + \eta\Psi(x),$$

[1] We adapt the name "proximal" from [Do et al., 2009], but note that while similar proximal regularization functions were considered, that paper deals only with gradient descent algorithms, not FTRL.

[2] Our Ψ is denoted r in [Duchi et al., 2010b]

which can be generalized to

$$x_{t+1} = \arg\min_x g_t \cdot x + \Psi(x) + \mathcal{B}_{1:t}(x, x_t), \qquad (11)$$

where the learning rate η has been rolled into the definition of $R_1, \ldots, R_t$. When Ψ is chosen to be the indicator function on a convex set, COMID reduces to standard mirror descent with greedy projection.

2.1 An Equivalence Theorem for Proximal Regularization

In Theorem 2, we show that mirror descent algorithms can be viewed as FTRL algorithms.

Theorem 2. *Let R_t be a sequence of differentiable origin-centered convex functions ($\nabla R_t(0) = 0$), with R_1 strongly convex, and let Ψ be an arbitrary convex function. Let $x_1 = \hat{x}_1 = 0$. For a sequence of loss functions $f_t(x) = g_t \cdot x + \Psi(x)$, let the sequence of points played by the composite-objective mirror descent algorithm be*

$$\hat{x}_{t+1} = \arg\min_x \; g_t \cdot x + \Psi(x) + \tilde{\mathcal{B}}_{1:t}(x, \hat{x}_t), \qquad (12)$$

where $\tilde{R}_t(x) = R_t(x - \hat{x}_t)$, and $\tilde{\mathcal{B}}_t = \mathcal{B}_{\tilde{R}_t}$, so $\tilde{\mathcal{B}}_{1:t}$ is the Bregman divergence with respect to $\tilde{R}_1 + \cdots + \tilde{R}_t$. Consider the alternative sequence of points x_t played by a proximal FTRL algorithm, applied to these same f_t, defined by

$$x_{t+1} = \arg\min_x \; (g_{1:t} + \phi_{1:t-1}) \cdot x + \tilde{R}_{1:t}(x) + \Psi(x) \quad (13)$$

where $\phi_t \in \partial\Psi(x_{t+1})$ such that $g_{1:t} + \phi_{1:t-1} + \nabla\tilde{R}_{1:t}(x_{t+1}) + \phi_t = 0$. Then, these algorithms are equivalent, in that $x_t = \hat{x}_t$ for all $t > 0$.

The Bregman divergences used by mirror descent in the theorem are with respect to the proximal functions $\tilde{R}_{1:t}$, whereas typically (as in Equation (10)) these functions would not depend on the previous points played. We will show when $R_t(x) = \frac{1}{2}\|Q_t^{\frac{1}{2}}x\|_2^2$, this issue disappears. Considering arbitrary Ψ functions also complicates the theorem statement somewhat. The following Corollary sidesteps these complexities, to state a simple direct equivalence result:

Corollary 3. *Let $f_t(x) = g_t \cdot x$. Then, the following algorithms play identical points:*

- *Gradient descent with positive semi-definite learning rates Q_t, defined by:*

$$x_{t+1} = x_t - Q_{1:t}^{-1} g_t.$$

- *FTRL-Proximal with regularization functions $\tilde{R}_t(x) = \frac{1}{2}\|Q_t^{\frac{1}{2}}(x - x_t)\|_2^2$, which plays*

$$x_{t+1} = \arg\min_x g_{1:t} \cdot x + \tilde{R}_{1:t}(x).$$

Proof. Let $R_t(x) = \frac{1}{2}x^\top Q_t x$. It is easy to show that $R_{1:t}$ and $\tilde{R}_{1:t}$ differ by only a linear function, and so (by a standard result) $\mathcal{B}_{1:t}$ and $\tilde{\mathcal{B}}_{1:t}$ are equal, and simple algebra reveals

$$\mathcal{B}_{1:t}(x, y) = \tilde{\mathcal{B}}_{1:t}(x, y) = \frac{1}{2}\|Q_{1:t}^{\frac{1}{2}}(x - y)\|_2^2.$$

Then, it follows from Equation (9) that the first algorithm is a mirror descent algorithm using this Bregman divergence. Taking $\Psi(x) = 0$ and hence $\phi_t = 0$, the result follows from Theorem 2. □

Extending the approach of the corollary to FOBOS, we see the only difference between that algorithm and FTRL-Proximal is that FTRL-Proximal optimizes over $t\Psi(x)$, whereas in Equation (13) we optimize over $\phi_{1:t-1} \cdot x + \Psi(x)$ (see Table 1). Thus, FOBOS is equivalent to FTRL-Proximal, except that FOBOS approximates all but the most recent Ψ function by a subgradient.

The behavior of FTRL-Proximal can thus be different from COMID when a non-trivial Ψ is used. While we are most concerned with the choice $\Psi(x) = \lambda\|x\|_1$, it is also worth considering what happens when Ψ is the indicator function on a feasible set $\mathcal{F}$. Then, Theorem 2 shows that mirror descent on $f_t(x) = g_t \cdot x + \Psi(x)$ (equivalent to COMID in this case) approximates previously seen Ψs by their subgradients, whereas FTRL-Proximal optimizes over Ψ explicitly. In this case, it can be shown that the mirror-descent update corresponds to the standard greedy projection [Zinkevich, 2003], whereas FTRL-Proximal corresponds to a lazy projection [McMahan and Streeter, 2010].[3]

Proof of Theorem 2. The proof is by induction. For the base case, we have $x_1 = \hat{x}_1 = 0$. For the induction step, assume $x_t = \hat{x}_t$. Theorem 1 guarantees the existence of a suitable ϕ_t for use in the update of Equation (13), and so in particular there exists a unique $\phi_{t-1} \in \partial\Psi(x_t)$ such that

$$g_{1:t-1} + \phi_{1:t-2} + \nabla\tilde{R}_{1:t-1}(x_t) + \phi_{t-1} = 0,$$

and so applying the induction hypothesis,

$$-\nabla\tilde{R}_{1:t-1}(\hat{x}_t) = g_{1:t-1} + \phi_{1:t-1}. \qquad (14)$$

[3] Zinkevich [2004, Sec. 5.2.3] describes a different lazy projection algorithm, which requires an appropriately chosen constant step-size to get low regret. FTRL-Proximal does not suffer from this problem, because it always centers the additional regularization R_t at points in $\mathcal{F}$, whereas our results show the algorithm of Zinkevich centers the additional regularization *outside* of $\mathcal{F}$, at the optimum of the unconstrained optimization. This leads to the high regret in the case of standard adaptive step sizes, because the algorithm can get "stuck" too far outside the feasible set to make it back to the other side.

Then, starting from Equation (12),

$$\hat{x}_{t+1} = \arg\min_x g_t \cdot x + \tilde{\mathcal{B}}_{1:t}(x, \hat{x}_t) + \Psi(x).$$

We now manipulate this expression for $\hat{x}_{t+1}$. Applying Theorem 1, for some $\phi'_t \in \partial\Psi(\hat{x}_{t+1})$,

$$\begin{aligned}\hat{x}_{t+1} &= \arg\min_x g_t \cdot x + \tilde{\mathcal{B}}_{1:t}(x, \hat{x}_t) + \phi'_t \cdot x \\ &= \arg\min_x g_t \cdot x + \tilde{R}_{1:t}(x) - \tilde{R}_{1:t}(x_t) \\ &\qquad - \nabla\tilde{R}_{1:t}(\hat{x}_t)(x - x_t) + \phi'_t \cdot x \qquad \text{Defn. of } \tilde{\mathcal{B}}_{1:t}\end{aligned}$$

Dropping terms independent of x,

$$\begin{aligned}&= \arg\min_x g_t \cdot x + \tilde{R}_{1:t}(x) - \nabla\tilde{R}_{1:t}(\hat{x}_t)x + \phi'_t \cdot x \\ &= \arg\min_x g_t \cdot x + \tilde{R}_{1:t}(x) - \nabla\tilde{R}_{1:t-1}(\hat{x}_t)x + \phi'_t \cdot x\end{aligned}$$

since $\nabla\tilde{R}_t(\hat{x}_t) = 0$, and then using Eq (14)

$$= \arg\min_x g_t \cdot x + \tilde{R}_{1:t}(x) + (g_{1:t-1} + \phi_{1:t-1})x + \phi'_t \cdot x.$$

We conclude $\hat{x}_{t+1} = x_{t+1}$, as $(\hat{x}_{t+1}, \phi'_t)$ satisfy the conditions of Theorem 1 with respect to the objective in the optimization defining x_{t+1}. □

2.2 An Equivalence Theorem for Origin-Centered Regularization

For the moment, suppose $\Psi(x) = 0$. So far, we have shown conditions under which gradient descent on $f_t(x) = g_t \cdot x$ with an adaptive step size is equivalent to follow-the-proximally-regularized-leader. In this section, we show that mirror descent on the *regularized* functions $f_t^R(x) = g_t \cdot x + R_t(x)$, with a certain natural step-size, is equivalent to a follow-the-regularized-leader algorithm with origin-centered regularization.

The algorithm we consider was introduced by Bartlett et al. [2007, Theorem 2.1]. Letting $R_t(x) = \frac{\sigma_t}{2}\|x\|_2^2$ and fixing $\eta_t = \frac{1}{\sigma_{1:t}}$, their adaptive online gradient descent algorithm is

$$x_{t+1} = x_t - \eta_t \nabla f_t^R(x_t) = x_t - \eta_t(g_t + \sigma_t x_t)).$$

We show (in Corollary 5) that this algorithm is identical to follow-the-leader on the functions $f_t^R(x) = g_t \cdot x + R_t(x)$, an algorithm that is minimax optimal in terms of regret against quadratic functions like f^R [Abernethy et al., 2008]. As with the previous theorem, the difference between the two is how they handle an arbitrary Ψ. If one uses $\tilde{R}_t(x) = \frac{\sigma_t}{2}\|x - x_t\|_2^2$ in place of $R_t(x)$, this algorithm reduces to standard online gradient descent [Do et al., 2009].

The key observation of [Bartlett et al., 2007] is that if the underlying functions ℓ_t have strong convexity, we can roll that into the R_t functions, and so introduce less additional stabilizing regularization, leading to regret bounds that interpolate between $\sqrt{T}$ for linear functions and $\log T$ for strongly convex functions. Their work did not consider composite objectives (Ψ terms), but our equivalence theorems show their adaptivity techniques can be lifted to algorithms like RDA and FTRL-Proximal that handle such non-smooth functions more effectively than mirror descent formulations.

We will prove our equivalence theorem for a generalized versions of the algorithm. Instead of vanilla gradient descent, we analyze the mirror descent algorithm of Equation (11), but now g_t is replaced by $\nabla f_t^R(x_t)$, and we add the composite term $\Psi(x)$.

Theorem 4. *Let $f_t(x) = g_t \cdot x$, and let $f_t^R(x) = g_t \cdot x + R_t(x)$, where R_t is a differentiable convex function. Let Ψ be an arbitrary convex function. Consider the composite-objective mirror-descent algorithm which plays*

$$\hat{x}_{t+1} = \arg\min_x \nabla f_t^R(\hat{x}_t) \cdot x + \Psi(x) + \mathcal{B}_{1:t}(x, \hat{x}_t), \quad (15)$$

and the FTRL algorithm which plays

$$x_{t+1} = \arg\min_x f_{1:t}^R(x) + \phi_{1:t-1} \cdot x + \Psi(x), \quad (16)$$

for $\phi_t \in \partial\Psi(x_{t+1})$ such that $g_{1:t} + \nabla R_{1:t}(x_{t+1}) + \phi_{1:t-1} + \phi_t = 0$. If both algorithms play $\hat{x}_1 = x_1 = 0$, then they are equivalent, in that $x_t = \hat{x}_t$ for all $t > 0$.

The most important corollary of this result is that it lets us add the Adaptive Online Gradient Descent algorithm to Table 1. It is also instructive to specialize to the simplest case when $\Psi(x) = 0$ and the regularization is quadratic:

Corollary 5. *Let $f_t(x) = g_t \cdot x$ and $f_t^R(x) = g_t \cdot x + \frac{\sigma_t}{2}\|x\|_2^2$. Then following update algorithms play identical points:*

- *FTRL, which plays $x_{t+1} = \arg\min_x f_{1:t}^R(x)$.*
- *Gradient descent on the functions f^R using the step size $\eta_t = \frac{1}{\sigma_{1:t}}$, which plays*

$$x_{t+1} = x_t - \eta_t \nabla f_t^R(x_t)$$

- *Revisionist constant-step size gradient descent with $\eta_t = \frac{1}{\sigma_{1:t}}$, which plays*

$$x_{t+1} = -\eta_t g_{1:t}.$$

The last equivalence in the corollary follows from deriving the closed form for the point played by FTRL. We now proceed to the proof of the general theorem:

Proof of Theorem 4. The proof is by induction, using the induction hypothesis $\hat{x}_t = x_t$. The base case for $t = 1$ follows by inspection. Suppose the induction hypothesis holds for t; we will show it also holds for $t+1$. Again let $r_t = \nabla R_t$ and consider Equation (16). Since R_1 is assumed to be strongly convex, applying Theorem 1 gives us that x_t is the unique solution to $\nabla f^R_{1:t-1}(x_t) + \phi_{1:t-1} = 0$ and so $g_{1:t-1} + r_{1:t-1}(x_t) + \phi_{1:t-1} = 0$. Then, by the induction hypothesis,

$$-r_{1:t-1}(\hat{x}_t) = g_{1:t-1} + \phi_{1:t-1}. \tag{17}$$

Now consider Equation (15). Since R_1 is strongly convex, $\mathcal{B}_{1:t}(x, \hat{x}_t)$ is strongly convex in its first argument, and so by Theorem 1 we have that $\hat{x}_{t+1}$ and some $\phi'_t \in \partial\Psi(\hat{x}_{t+1})$ are the unique solution to

$$\nabla f^R_t(\hat{x}_t) + \phi'_t + r_{1:t}(\hat{x}_{t+1}) - r_{1:t}(\hat{x}_t) = 0,$$

since $\nabla_p \mathcal{B}_R(p, q) = r(p) - r(q)$. Beginning from this equation,

$$\begin{aligned}
0 &= \nabla f^R_t(\hat{x}_t) + \phi'_t + r_{1:t}(\hat{x}_{t+1}) - r_{1:t}(\hat{x}_t) \\
&= g_t + r_t(\hat{x}_t) + \phi'_t + r_{1:t}(\hat{x}_{t+1}) - r_{1:t}(\hat{x}_t) \\
&= g_t + r_{1:t}(\hat{x}_{t+1}) + \phi'_t - r_{1:t-1}(\hat{x}_t) \\
&= g_t + r_{1:t}(\hat{x}_{t+1}) + \phi'_t + g_{1:t-1} + \phi_{1:t-1} \qquad \text{Eq (17)} \\
&= g_{1:t} + r_{1:t}(\hat{x}_{t+1}) + \phi_{1:t-1} + \phi'_t.
\end{aligned}$$

Applying Theorem 1 to Equation (16), (x_{t+1}, ϕ_t) are the unique pair such that

$$g_{1:t} + r_{1:t}(x_{t+1}) + \phi_{1:t-1} + \phi_t = 0$$

and $\phi_t \in \partial\Psi(x_{t+1})$, and so we conclude $\hat{x}_{t+1} = x_{t+1}$ and $\phi'_t = \phi_t$. □

3 EXPERIMENTS

We compare FOBOS, FTRL-Proximal, and RDA on a variety of datasets to illustrate the key differences between the algorithms, from the point of view of introducing sparsity with L_1 regularization. In all experiments we optimize log-loss (see Section 1).

Binary Classification We compare FTRL-Proximal, RDA, and FOBOS on several public datasets. We used four sentiment classification data sets (Books, Dvd, Electronics, and Kitchen), available from [Dredze, 2010], each with 1000 positive examples and 1000 negative examples,[4] as well as the scaled versions of the rcv1.binary (20,242 examples) and news20.binary (19,996 examples) data sets from LIBSVM [Chang and Lin, 2010].

[4]We used the features provided in processed_acl.tar.gz, and scaled each vector of counts to unit length.

All our algorithms use a learning rate scaling parameter γ (see Section 1). The optimal choice of this parameter can vary somewhat from dataset to dataset, and for different settings of the L_1 regularization strength λ. For these experiments, we first selected the best γ for each (dataset, algorithm, λ) combination on a random shuffling of the dataset. We did this by training a model using each possible setting of γ from a reasonable grid (12 points in the range $[0.3, 1.9]$), and choosing the γ with the highest online AUC. We then fixed this value, and report the average AUC over 5 different shufflings of each dataset. We chose the area under the ROC curve (AUC) as our accuracy metric as we found it to be more stable and have less variance than the mistake fraction. However, results for classification accuracy were qualitatively very similar.

Ranking Search Ads by Click-Through-Rate We collected a dataset of about 1,000,000 search ad impressions from a large search engine,[5] corresponding to ads shown on a small set of search queries. We formed examples with a feature vector θ_t for each ad impression, using features based on the text of the ad and the query, as well as where on the page the ad showed. The target label y_t is 1 if the ad was clicked, and -1 otherwise.

Smaller learning-rates worked better on this dataset; for each (algorithm, λ) combination we chose the best γ from 9 points in the range $[0.03, 0.20]$. Rather than shuffling, we report results for a single pass over the data using the best γ, processing the events in the order the queries actually occurred. We also set a lower bound for the stabilizing terms $\bar{\sigma}_t$ of 20.0, (corresponding to a maximum learning rate of 0.05), as we found this improved accuracy somewhat. Again, qualitative results did not depend on this choice.

Results Table 2 reports AUC accuracy (larger numbers are better), followed by the density of the final predictor x_T (number of non-zeros divided by the total number of features present in the training data). We measured accuracy online, recording a prediction for each example before training on it, and then computing the AUC for this set of predictions. For these experiments, we fixed $\lambda = 0.05/T$ (where T is the number of examples in the dataset), which was sufficient to introduce non-trivial sparsity. Overall, there is very little difference between the algorithms in terms of accuracy, with RDA having a slight edge for these choices for λ. Our main point concerns the sparsity numbers. It has been shown before that RDA outperforms FO-

[5]While we report results on a single dataset, we repeated the experiments on two others, producing qualitatively the same results. No user-specific data was used in these experiments.

Table 2: AUC (area under the ROC curve) for online predictions and sparsity in parentheses. The best value for each dataset is bolded. For these experiments, λ was fixed at $0.05/T$.

DATA	FTRL-PROXIMAL	RDA	FOBOS
BOOKS	0.874 (0.081)	**0.878** (**0.079**)	0.877 (0.382)
DVD	0.884 (0.078)	0.886 (**0.075**)	**0.887** (0.354)
ELECTRONICS	0.916 (0.114)	**0.919** (**0.113**)	0.918 (0.399)
KITCHEN	0.931 (**0.129**)	**0.934** (0.130)	0.933 (0.414)
NEWS	0.989 (**0.052**)	**0.991** (0.054)	0.990 (0.194)
RCV1	0.991 (**0.319**)	**0.991** (0.360)	0.991 (0.488)
WEB SEARCH ADS	**0.832** (**0.615**)	0.831 (0.632)	0.832 (0.849)

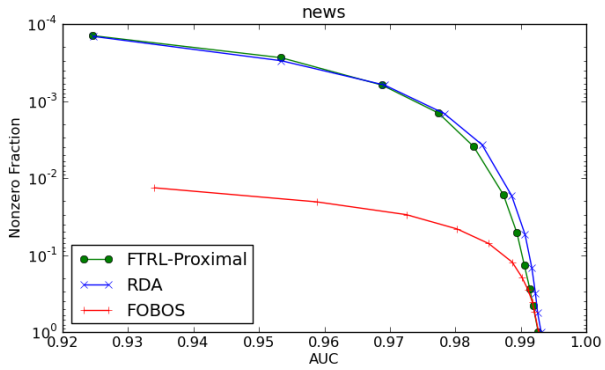

Figure 1: Sparsity versus accuracy tradeoffs on the 20 newsgroups dataset. Sparsity increases on the y-axis, and AUC increases on the x-axis, so the top right corner gets the best of both worlds. FOBOS is pareto-dominated by FTRL-Proximal and RDA.

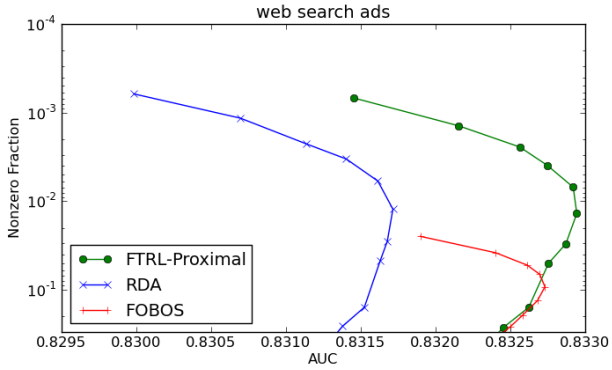

Figure 2: The same comparison as the previous figure, but on a large search ads ranking dataset. On this dataset, FTRL-Proximal significantly outperforms both other algorithms.

BOS in terms of sparsity. The question then is how does FTRL-Proximal perform, as it is a hybrid of the two, selecting additional stabilization R_t in the manner of FOBOS, but handling the L_1 regularization in the manner of RDA. These results make it very clear: it is the treatment of L_1 regularization that makes the key difference for sparsity, as FTRL-Proximal behaves very comparably to RDA in this regard.

Fixing a particular value of λ, however, does not tell the whole story. For all these algorithms, one can trade off accuracy to get more sparsity by increasing the λ parameter. The best choice of this parameter depends on the application as well as the dataset. For example, if storing the model on an embedded device with expensive memory, sparsity might be relatively more important. To show how these algorithms allow different tradeoffs, we plot sparsity versus AUC for the different algorithms over a range of λ values. Figure 1 shows the tradeoffs for the 20 newsgroups dataset, and Figure 2 shows the tradeoffs for web search ads.

In all cases, FOBOS is pareto-dominated by RDA and FTRL-Proximal. These two algorithms are almost indistinguishable in the their tradeoff curves on the newsgroups dataset, but on the ads dataset FTRL-Proximal significantly outperforms RDA as well.[6]

Conclusions We have shown a close relationship between certain mirror descent algorithms like FOBOS, and FTRL-style algorithms like RDA. This was accomplished by expressing the mirror descent update as a global optimization in the style of FTRL. This reformulation provides a clear characterization of the difference in how L_1 regularization (and in general, an arbitrary non-smooth regularizer Ψ) is handled by these algorithms. Experimental results demonstrate that it is this difference that accounts for the superior sparsity produced by RDA. We also introduced the composite-objective FTRL-Proximal algorithm that can be seen as a hybrid between the other two, centering stabilizing regularization in the manner of FOBOS, but handling Ψ (an in particular, L_1 regularization) in the manner of RDA. We showed that this algorithm can outperform both of the others on a large, real-world dataset.

[6]The improvement is more significant than it first appears. A simple model with only features based on where the ads were shown achieves an AUC of nearly 0.80, and the inherent uncertainty in the clicks means that even predicting perfect probabilities would produce an AUC significantly less than 1.0, perhaps 0.85.

Acknowledgments

The author wishes to thank Matt Streeter for numerous helpful discussions and comments, and Fernando Pereira for a conversation that led to the focus on the choice $\Psi(x) = \|x\|_1$.

References

Jacob Abernethy, Peter L. Bartlett, Alexander Rakhlin, and Ambuj Tewari. Optimal strategies and minimax lower bounds for online convex games. In *COLT*, 2008.

Peter Bartlett, Elad Hazan, and Alexander Rakhlin. Adaptive online gradient descent. Technical Report UCB/EECS-2007-82, EECS Department, University of California, Berkeley, Jun 2007.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM data sets. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/, 2010.

Chuong B. Do, Quoc V. Le, and Chuan-Sheng Foo. Proximal regularization for online and batch learning. In *ICML*, 2009.

Mark Dredze. Multi-domain sentiment dataset (v2.0). http://www.cs.jhu.edu/~mdredze/datasets/sentiment/, 2010.

John Duchi and Yoram Singer. Efficient learning using forward-backward splitting. In *NIPS*, 2009.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *COLT*, 2010a.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Ambuj Tewari. Composite objective mirror descent. In *COLT*, 2010b.

H. Brendan McMahan. A unified analysis of regularized dual averaging and composite mirror descent with implicit updates. Submitted, 2011.

H. Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization. In *COLT*, 2010.

Lin Xiao. Dual averaging method for regularized stochastic learning and online optimization. In *NIPS*, 2009.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *ICML*, 2003.

Martin Zinkevich. *Theoretical guarantees for algorithms in multi-agent settings*. PhD thesis, Pittsburgh, PA, USA, 2004.