Proximal regularization for online and batch learning

Chuong B. Do CHUONGDO@CS.STANFORD.EDU
Quoc V. Le QUOCLE@CS.STANFORD.EDU
Computer Science Department, Stanford University, Stanford, CA 94305, USA

Chuan-Sheng Foo CSFOO@CS.STANFORD.EDU
Institute for Infocomm Research, Singapore 138632, Singapore

Abstract

Many learning algorithms rely on the curvature (in particular, strong convexity) of regularized objective functions to provide good theoretical performance guarantees. In practice, the choice of regularization penalty that gives the best testing set performance may result in objective functions with little or even no curvature. In these cases, algorithms designed specifically for regularized objectives often either fail completely or require some modification that involves a substantial compromise in performance.

We present new online and batch algorithms for training a variety of supervised learning models (such as SVMs, logistic regression, structured prediction models, and CRFs) under conditions where the optimal choice of regularization parameter results in functions with low curvature. We employ a technique called *proximal regularization*, in which we solve the original learning problem via a sequence of modified optimization tasks whose objectives are chosen to have greater curvature than the original problem. Theoretically, our algorithms achieve low regret bounds in the online setting and fast convergence in the batch setting. Experimentally, our algorithms improve upon state-of-the-art techniques, including Pegasos and bundle methods, on medium and large-scale SVM and structured learning tasks.

1. Introduction

Consider the task of training a linear SVM:

$$\min_{\mathbf{w}\in\mathbb{R}^n} \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{m}\sum_{i=1}^{m} \max(0, 1 - y^{(i)}\mathbf{w}^\mathsf{T}\mathbf{x}^{(i)}). \quad (1)$$

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

In this optimization problem, the L_2 regularization penalty plays two important roles: not only does the quadratic term prevent overfitting to the empirical loss on the training data, but in fact, it also controls a measure of curvature of the objective function, known as its *strong convexity*.

In the past several years, a number of approaches have been proposed for training linear SVMs, ranging from batch methods such as the cutting plane algorithm (Joachims, 2006) to online methods such as the PEGASOS subgradient algorithm (Shalev-Shwartz et al., 2007). In essentially all of these algorithms (for which the relevant bounds are known), theory indicates that the number of iterations required to obtain an ϵ-accurate solution is roughly $O(1/\lambda\epsilon)$. For example, cutting-plane methods require $O(1/\lambda\epsilon)$ passes through the training set (Smola et al., 2008), whereas PEGASOS must process $\tilde{O}(1/\lambda\epsilon)$ training examples to ensure ϵ accuracy on expectation. In both cases, the theoretical bounds depend largely on the chosen value of the regularization hyperparameter λ.

For many real world problems, however, the ideal choice of λ can be quite small. When this is the case, state-of-the-art cutting plane and subgradient algorithms give unacceptably slow convergence, both in theory and in practice. Recently, (Bartlett et al., 2008) described an *adaptive online gradient descent* algorithm based on the simple intuition that an objective function with low curvature can be stabilized by adding extra terms whose purpose is to increase curvature. In this paper, we extend these ideas to construct new online and batch algorithms suitable for training a wide variety of supervised learning models.

Specifically, we design a sequence of optimization tasks, each of which is a variant of the original problem modified to include an extra *proximal regularization* term. We show how to choose these proximal terms in an adaptive fashion such that the resulting sequence of minimizers (or approximate minimizers) converge to the solution of the original optimization problem. Finally, we describe some simple heuristic modifications to these algorithms that retain all optimality guarantees while resulting in considerable performance improvements in practice. In the online setting, our analysis leads naturally to a stochastic subgradient-style algorithm along the lines of the PEGASOS. In the batch setting, our analysis yields an improved cutting-plane/bundle method. Both in theory and in experiments, our methods exhibit comparable performance for large λ (high curvature) compared to existing methods and dramatic improvements for small λ (low curvature). [1]

2. Preliminaries

Let $\|\cdot\|$ denote the Euclidean norm, $\|\mathbf{x}\| := \sqrt{\mathbf{x}^\mathsf{T}\mathbf{x}}$. Given a point $\mathbf{x} \in \mathbb{R}^n$ and a compact (i.e., closed, bounded) subset $S \subseteq \mathbb{R}^n$, let $\Pi_S[\mathbf{x}] := \arg\min_{\mathbf{y}\in S} \|\mathbf{x} - \mathbf{y}\|$ denote the Euclidean projection of $\mathbf{x}$ onto S. For notational convenience, we use notational shorthand $c_{a:b} := \sum_{i=a}^{b} c_i$ for any sequence of scalars $c_a, c_{a+1}, \ldots, c_{b-1}, c_b \in \mathbb{R}$. For a vector $\mathbf{x} \in \mathbb{R}^n$ and $c \in \mathbb{R}$, let $[\mathbf{x}; c] \in \mathbb{R}^{n+1}$ denote the concatenation of c onto the end of $\mathbf{x}$. For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, let $\mathbf{x} \succeq \mathbf{y}$ denote the component-wise inequalities, $x_i \geq y_i, \forall i$. Let $\mathbf{0}$ and $\mathbf{1}$ denote the vectors of all 0's and all 1's, respectively.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be λ-*strongly convex* if for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and any subgradient $\mathbf{g}$ belonging to the subdifferential[2] $\partial f(\mathbf{x})$ of f at $\mathbf{x}$, $f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{g}^\mathsf{T}(\mathbf{y} - \mathbf{x}) + \frac{\lambda}{2}\|\mathbf{y} - \mathbf{x}\|^2$. Here, we consider learning problems associated with the optimization of λ-strongly convex functions in both the online and batch settings.

In the online setting, we base our analyses on the concept of a *convex repeated game*. A convex repeated game is a two-player game consisting of T rounds. During round t, the first player proposes a vector $\mathbf{w}_t$ belonging to some compact convex set S, the second player responds by choosing a λ_t-strongly convex function of the form $f_t(\mathbf{w}) := \frac{\lambda_t}{2}\|\mathbf{w}\|^2 + \ell_t(\mathbf{w})$ for some convex function ℓ_t, and then the first player suffers loss $f_t(\mathbf{w}_t)$. We assume that $\lambda_t \geq 0$, and that the same set S is used in each round; for simplicity, we assume throughout that S is an origin-centered closed ball of radius R. Here, we seek an algorithm to minimize the first player's *regret*, $\sum_{t=1}^{T} f_t(\mathbf{w}_t) - \min_{\mathbf{u}\in S}\sum_{t=1}^{T} f_t(\mathbf{u})$, i.e., the excess loss suffered compared to the minimum loss possible for any fixed choice of $\mathbf{w} \in S$.

In the batch setting, we are given a λ-strongly convex function of the form $f(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \ell(\mathbf{w})$, where ℓ is again a convex function. Here, we will assume $\lambda > 0$ in order to ensure that the optimization problem is well-posed. If $\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^n} f(\mathbf{w})$, then our goal will be to find an approximate minimizer $\mathbf{w}$ such that $f(\mathbf{w}) - f(\mathbf{w}^*) \leq \epsilon$.

[1]For an extended version of this paper with proofs, see http://ai.stanford.edu/~chuongdo/papers/proximal_proofs.pdf

[2]The *subdifferential* $\partial f(\mathbf{x})$ of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at $\mathbf{x}$ is the set of all vectors $\mathbf{g}$ such that $f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{g}^\mathsf{T}(\mathbf{y} - \mathbf{x})$ for all $\mathbf{y} \in \mathbb{R}^n$; elements belonging to the subdifferential are known as *subgradients*.

Algorithm 1 Projected subgradient descent

```
Initialize $\mathbf{w}_1 \leftarrow \mathbf{0}$.
for $t \leftarrow 1, \ldots, T$ do
    Receive a $\lambda_t$-strongly convex function $f_t$.
    Choose $\mathbf{g}_t \in \partial f_t(\mathbf{w}_t)$.
    Set $\eta_t \leftarrow 1/\lambda_{1:t}$.
    Set $\mathbf{w}_{t+1} \leftarrow \Pi_S[\mathbf{w}_t - \eta_t \mathbf{g}_t]$.
end for
return $\mathbf{w}_{T+1}$.
```

Algorithm 2 Proximal projected subgradient descent

```
Initialize $\mathbf{w}_1 \leftarrow \mathbf{0}$.
for $t \leftarrow 1, \ldots, T$ do
    Receive a $\lambda_t$-strongly convex function $f_t$.
    Choose $\mathbf{g}_t \in \partial f_t(\mathbf{w}_t)$.
    Set $\tau_t \leftarrow \frac{-\lambda_{1:t} - \tau_{1:t-1} + \sqrt{(\lambda_{1:t} + \tau_{1:t-1})^2 + \frac{G_t^2}{R^2}}}{2}$.
    Set $\eta_t \leftarrow 1/(\lambda_{1:t} + \tau_{1:t})$.
    Set $\mathbf{w}_{t+1} \leftarrow \Pi_S[\mathbf{w}_t - \eta_t \mathbf{g}_t]$.
end for
return $\mathbf{w}_{T+1}$.
```

3. Online proximal learning

As a starting point, we recall the projected subgradient algorithm for strongly convex repeated games proposed by (Hazan et al., 2007) and later generalized by (Bartlett et al., 2008), as stated in Algorithm 1. In this algorithm, the first player updates his parameter vector in each round by taking a projected subgradient step, $\mathbf{w}_{t+1} \leftarrow \Pi_S[\mathbf{w}_t - \eta_t \mathbf{g}_t]$. When the step size $\eta_t = 1/\lambda_{1:t}$, we obtain the following regret bound (Bartlett et al., 2008):

Lemma 1. *Suppose that $\lambda_t > 0$ and $\|\mathbf{g}_t\| \leq G_t$ for $t = 1, \ldots, T$. Then, for any $\mathbf{u} \in S$, Algorithm 1 satisfies*

$$\sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq \frac{1}{2}\sum_{t=1}^{T}\frac{G_t^2}{\lambda_{1:t}}. \tag{2}$$

When $\lambda_t = \lambda$ and $G_t = G$ in each round, then the right hand side of the inequality can be further upper-bounded by $\frac{G^2}{2\lambda}(1 + \log T)$. Algorithm 1, thus, is an example of an algorithm with *logarithmic regret*. When λ is small, however, this guaranteed regret can still be large.

3.1. Proximal regret bound

Now, suppose we run Algorithm 1 on the sequence of modified functions,

$$f'_t(\mathbf{w}) := f_t(\mathbf{w}) + \frac{\tau_t}{2}\|\mathbf{w} - \mathbf{w}_t\|^2. \tag{3}$$

for some setting of constants $\tau_1, \ldots, \tau_T \in \mathbb{R}$. We refer to the additional quadratic term in each of our modified functions as a *proximal regularization term*. Whereas each f_t is λ_t-strongly convex, each modified function f'_t is $(\lambda_t + \tau_t)$-strongly convex. Also, since the gradient of the proximal regularization term is zero when evaluated at $\mathbf{w}_t$, it follows immediately that $\partial f'_t(\mathbf{w}_t) = \partial f_t(\mathbf{w}_t)$. Thus, the updates in the proximal regularization case differ from the non-proximal algorithm only in the choice of step sizes, since we can still use the same subgradients.

The idea of adding temporary regularization terms in order to achieve better bounds on the regret of a learning algorithm was first introduced in (Bartlett et al., 2008), who considered modified objective functions of the form

$$f''_t(\mathbf{w}) := f_t(\mathbf{w}) + \frac{\tau_t}{2}\|\mathbf{w}\|^2. \qquad (4)$$

Unlike in the proximal case, $\partial f''_t(\mathbf{w}_t) \neq \partial f_t(\mathbf{w}_t)$. In Section 5, we compare empirically these two choices.

To analyze the proximal regularization method, we apply Lemma 1 to the sequence of functions in (3) to obtain

Corollary 1. *Define*

$$\mathcal{R}_T(\tau_1, \ldots, \tau_T) := \frac{1}{2}\sum_{t=1}^{T}\left[4\tau_t R^2 + \frac{G_t^2}{\lambda_{1:t} + \tau_{1:t}}\right]. \qquad (5)$$

For any fixed $\tau_1, \ldots, \tau_T \geq 0$, running Algorithm 1 on the sequence of functions $f'_1, \ldots, f'_T$ from (3) *gives*

$$\sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq \mathcal{R}_T(\tau_1, \ldots, \tau_T). \qquad (6)$$

Here, the proof depends on the fact that $\|\mathbf{w} - \mathbf{w}_t\| \leq 2R$ for any $\mathbf{w}, \mathbf{w}_t \in S$. The strength of the regret bound, depends on the choice of constants $\tau_1, \ldots, \tau_T$. The key to the proximal regularization algorithm, then, is picking these constants so as to ensure that the regret is small.

3.2. Choosing proximal parameters

Suppose that the values λ_t and G_t for $t = 1, \ldots, T$ are determined independently of the choices made in the algorithm. We describe two approximate schemes for choosing τ_t's. The first scheme is a practical online balancing heuristic due to (Bartlett et al., 2008). The second scheme, makes the additional assumptions that the λ_t and G_t do not vary with t but has the benefit of allowing us to choose the τ_t's so that the regret bound is as tight as possible.

Strategy 1: Balancing heuristic. In the first approach, observe that the expression in (5) consists of two terms, one of which increases and one of which decreases as τ_t increases. During the tth step of the algorithm, consider the choice of $\tau_t \geq 0$ that ensures that the two terms are equal, i.e., $2\tau_t R^2 = \frac{G_t^2}{2(\lambda_{1:t}+\tau_{1:t})}$. This is a quadratic equation, with positive solution,

$$\tau_t = \tfrac{1}{2}\left(-\lambda_{1:t} - \tau_{1:t-1} + \sqrt{(\lambda_{1:t} + \tau_{1:t-1})^2 + \tfrac{G_t^2}{R^2}}\right).$$

In Algorithm 2, we provide pseudocode for the proximal projected subgradient descent algorithm using the balancing heuristic. Applying Lemma 3.1 from (Bartlett et al., 2008), we obtain the following bound:[3]

Theorem 1. *The regret obtained by Algorithm 2 is at most twice that of the optimal offline choice of $\tau_1, \ldots, \tau_T$, i.e.,*

$$\sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq 2\min_{\tau_1,\ldots,\tau_T}\frac{1}{2}\sum_{t=1}^{T}\left[4\tau_t R^2 + \frac{G_t^2}{\lambda_{1:t} + \tau_{1:t}}\right].$$

Strategy 2: Bound optimization. In the second approach, we bound the regret directly, via the following proposition:

Proposition 1. *Let*

$$(\tau_1^*, \ldots, \tau_T^*) = \operatorname*{arg\,min}_{\tau_1,\ldots,\tau_T \geq 0} \mathcal{R}_T(\tau_1, \ldots, \tau_T). \qquad (7)$$

Then $\tau_i^* = 0$ for all $i \neq 1$.

The benefit of the above proposition is that it allows us to reduce an optimization over many variables to a much simpler convex optimization problem over just a single variable, τ_1^* (which we simply call τ). If $\lambda_t = \lambda > 0$,[4] and $G_t = G$, then we can upper bound the regret with a simple closed form expression, parameterized by τ:

Theorem 2. *Under the above assumptions, let $\mathcal{R}_T$ denote the worst-case regret suffered by Algorithm 2. Then, for any $\tau > 0$, we have the upper bound, $\mathcal{R}_T \leq B(\tau)$, where*

$$B(\tau) := 4\tau R^2 + \frac{G^2}{\lambda}\left[\frac{1}{1+\tau/\lambda} + \log\left(\frac{T+\tau/\lambda}{1+\tau/\lambda}\right)\right].$$

Since the upper bound is a convex differentiable function of τ over the domain $\tau \geq 0$, one could optimize the bound directly using standard line search techniques. Alternatively, by substituting different values for τ into the expression above, we can obtain various upper bounds on the regret that Algorithm 2 will achieve. In particular,

Corollary 2. *When $\tau = 0$, then $B(\tau) := \frac{G^2}{\lambda}(1 + \log T)$.*

Corollary 3. *When $\tau = \frac{G\sqrt{T}}{2R}$, then $\lim_{\lambda\to 0} B(\tau) = 4RG\sqrt{T}$.*

The key intuition behind the efficiency of Algorithm 2 is that in some cases, one of these bounds may be better than the other. For example, when RG is sufficiently small relative to $\frac{G^2}{\lambda}$, the seemingly inferior square root bound can actually be better than the logarithmic regret bound for values of T that are not too large. Regardless of the situation, Theorem 1 implies that Algorithm 2 achieves a total regret no worse than twice the best bound for any τ.

3.3. Application: Linear SVMs

In this section, we consider the task of training a linear SVM. The approach we take here was inspired by the Pegasos algorithm (Shalev-Shwartz et al., 2007), currently regarded as one of the fastest methods for SVM training on

[3]For comparison, (Bartlett et al., 2008) derived a bound of

$$\sum_{t=1}^{T}(f_t(\mathbf{w}_t) - f_t(\mathbf{u})) \leq \min_{\tau_1,\ldots,\tau_T}\sum_{t=1}^{T}\left[3\tau_t R^2 + 2\frac{G_t^2}{\lambda_{1:t} + \tau_{1:t}}\right],$$

when using the modified functions in (4). These two expression are not directly comparable, though as we show in Section 5, the proximal algorithm performs better in our experiments.

[4]Note that if $\lambda = 0$, then the bound reduces to $2\tau R^2 + \sum_{t=1}^{T} G_t^2/2\tau$, whose minimum occurs at $\tau = \sqrt{\sum_{t=1}^{T} G_t^2/4R^2}$.

large-scale datasets. At its core, the Pegasos algorithm is essentially a wrapper for Algorithm 1.

Given training inputs $\{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{m}$, the Pegasos algorithm defines a sequence of functions $f_1, \ldots, f_T$. In the tth round, Pegasos randomly samples a subset A_t of fixed size k from $\{1, 2, \ldots, m\}$, defines $f_t(\mathbf{w})$ to be

$$\frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{|A_t|}\sum_{i \in A_t} \max(0, 1 - y^{(i)}\mathbf{w}^\mathsf{T}\mathbf{x}^{(i)}). \quad (8)$$

and runs Algorithm 1 with $\lambda_t = \lambda$, $G_t = \sqrt{\lambda} + \max_i \|\mathbf{x}^{(i)}\|$, $R = \frac{1}{\sqrt{\lambda}}$, and $S = \{\mathbf{w} \in \mathbb{R}^n : \|\mathbf{w}\| \le \frac{1}{\sqrt{\lambda}}\}$.[5]

To characterize the relationship between the strongly convex game defined by Pegasos and the linear SVM training problem, we state the following theorem and its corollary, both of whose proofs closely mirror that of Theorems 2 and 3 from (Shalev-Shwartz et al., 2007):

Theorem 3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a (strongly) convex function, let $S \subseteq \mathbb{R}^n$ be compact, and suppose $\mathbf{w}^* := \arg\min_{\mathbf{w}\in S} f(\mathbf{w})$. Let $\mathcal{A}$ be an algorithm for (strongly) convex repeated games with regret bound $\mathcal{R}_T$. Now, suppose we run $\mathcal{A}$ on a sequence of (strongly) convex functions $f_1, \ldots, f_T$ which satisfy, for all $t \in \{1, \ldots, T\}$, (1) $\mathbb{E}_{f_1,\ldots,f_t,\mathbf{w}_1,\ldots,\mathbf{w}_t}[f_t(\mathbf{w})] \le f(\mathbf{w})$ for all $\mathbf{w} \in S$; and (2) $\mathbb{E}_{f_1,\ldots,f_t,\mathbf{w}_1,\ldots,\mathbf{w}_{t-1}|\mathbf{w}_t}[f_t(\mathbf{w}_t)] = f(\mathbf{w}_t)$. If r is drawn uniformly at random from $\{1, \ldots, T\}$, then*

$$\mathbb{E}_r \mathbb{E}_{f_1,\ldots,f_t,\mathbf{w}_1,\ldots,\mathbf{w}_T}[f(\mathbf{w}_r) - f(\mathbf{w}^*)] \le \frac{\mathcal{R}_T}{T}. \quad (9)$$

Informally, this result provides an estimate of the average suboptimality Pegasos obtains in terms of the existing regret bound for its underlying algorithm for convex games.[6] Using Markov's inequality, it turns out that convergence in expected suboptimality implies convergence to optimality with high probability in the following sense:

Proposition 2. *Let $\delta \in (0, 1)$. Under the conditions above, with probability at least $1 - \delta$, $f(\mathbf{w}_r) - f(\mathbf{w}^*) \le \frac{\mathcal{R}_T}{\delta T}$.*

Now, we turn to the task of converting Algorithm 2 into an SVM solver, in the same manner as Pegasos. This time, we again assume that the functions $f_1, \ldots, f_T$ are sampled as in the same manner for the Pegasos algorithm, and for now, we assume the same settings of the constants λ_t and G_t. We now analyze the efficiency of our optimization algorithm by characterizing the number of iterations needed to guarantee ϵ-optimality with high probability.

[5]As shown in (Shalev-Shwartz et al., 2007), one can guarantee using a strong duality argument that the optimal solution will always have norm at most $\frac{1}{\sqrt{\lambda}}$, so using S as the feasible set does not impose any additional restrictions.

[6]Note that a version of the theorem replacing $\mathbf{w}_r$ with $\overline{\mathbf{w}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{w}_t$ follows easily from Jensen's inequality. Though this leads to a potentially more stable version of Algorithm 3, the resulting algorithm in practice often converges less quickly and may be less computationally efficient to implement (for problems where the feature vectors $\mathbf{x}_i$ are sparse).

Using Corollaries 2 and 3 from Section 3.2, and applying Proposition 2 we have

- With probability at least $1 - \delta$, $f(\mathbf{w}_r) - f(\mathbf{w}^*) \le \frac{G^2(1+\log T)}{\delta\lambda T}$. To ensure that the right hand side is no greater than ϵ requires $T \ge \tilde{O}(\frac{G^2}{\delta\lambda\epsilon})$ iterations.
- With probability at least $1-\delta$, $f(\mathbf{w}_r)-f(\mathbf{w}^*) \le \frac{4RG}{\delta\sqrt{T}}$. To ensure that the right hand side is no greater than ϵ requires $T \ge \frac{16R^2G^2}{\delta^2\epsilon^2}$ iterations.

In the first bound, we recover the $\tilde{O}(\frac{G^2}{\delta\lambda\epsilon})$ convergence rate of the Pegasos algorithm. In the second bound, we recover the $O(\frac{R^2G^2}{\delta^2\epsilon^2})$ rate of (Zinkevich, 2003), that, at least at first, appears not to depend on λ, suggesting that perhaps the proximal algorithm ought to give improved convergence when λ is small. On a closer examination, however, the dependence on λ is "hidden" inside the $R = \frac{1}{\sqrt{\lambda}}$ bound from the Pegasos algorithm. Making this dependence explicit, we achieve a rate of only $O(\frac{G^2}{\delta^2\lambda\epsilon^2})$.[7]

Here, the weak link in our analysis is the dependence of R on λ. In practice, however, the bound $R = \frac{1}{\sqrt{\lambda}}$ is often quite loose. Knowing ahead of time the norm of $\mathbf{w}^* = \arg\min_{\mathbf{w}} f(\mathbf{w})$ would help by allowing us to define a smaller feasible set S and thus obtain tighter bounds.

3.4. An optimistic strategy

With the above motivation in mind, we propose the adaptive strategy shown in Algorithm 3. In this method, we assume that we are initially given some desired level of suboptimality ϵ. Optimization proceeds in several phases. At the beginning of each phase, we "hypothesize" a setting of R. During each phase, we run the proximal projected subgradient strategy until either (1) $\|\mathbf{w}_t\|$ gets "close" to R, forcing us to increase R by a factor of $\sqrt{2}$ and start a new phase; or (2) enough iterations pass without this occurring, allowing us to declare convergence. The algorithm is "optimistic" in the sense that it initially assumes R to be small and only increases it as necessary. One can prove that:

Lemma 2. *Suppose that some particular phase ends without any increase in R. Define $\mathbf{w}^* = \arg\min_{\mathbf{w}\in S} f(\mathbf{w})$. Let r be chosen uniformly at random from $\{1, \ldots, T\}$. Then with probability at least $1 - \hat{\delta}$, $\mathbf{w}_r$ is ϵ-optimal, i.e., $f(\mathbf{w}_r) - f(\mathbf{w}^*) \le \epsilon$.*

Theorem 4. *For $\epsilon < \frac{1}{2}$, Algorithm 3 terminates after processing at most $\tilde{O}(\frac{G^2}{\delta\lambda\epsilon})$ examples; with probability at least $1 - \delta$, the resulting parameters $\mathbf{w}_r$ will be ϵ-optimal.*

Our analysis thus shows that the modified algorithm, in the worst case, is asymptotically equivalent to Pegasos up to

[7]These results are not particularly surprising, given the recent minimax analysis of (Abernethy et al., 2008), who showed that under certain assumptions, the regret bound of the regular projected subgradient algorithm is worst case optimal.

Algorithm 3 Optimistic proximal SVM solver

```
input  Training set $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^m$
       Regularization parameter $\lambda$
       Desired suboptimality $\epsilon$
       Allowed failure probability $\delta$
       Mini-batch size $k$

Define $S := \{\mathbf{w} \in \mathbb{R}^n : \|\mathbf{w}\| \leq 1/\sqrt{\lambda}\}$.
Set $G \leftarrow \max_i \left\|\mathbf{x}^{(i)}\right\| + \sqrt{\lambda}$.
Set $\hat{\delta} \leftarrow \frac{\delta}{3-\log_2 \lambda}$.
Initialize $R \leftarrow \min(1, \frac{1}{\sqrt{\lambda}})$.
Initialize $\mathbf{w}_1 \leftarrow \mathbf{0}$.
repeat
  Set CONVERGED $\leftarrow$ true.
  Find smallest $T$ such that $\min(\frac{G^2(1+\log T)}{\hat{\delta}\lambda T}, \frac{4RG}{\hat{\delta}\sqrt{T}}) \leq \epsilon$.
  for $t \leftarrow 1, \ldots, T$ do
    Sample $A_t \subseteq \{1, \ldots, m\}$ such that $|A_t| = k$.
    Define $f_t(\mathbf{w})$ according to (8).
    Choose $\mathbf{g}_t \in \partial f_t(\mathbf{w}_t)$.
    Set $\tau_t \leftarrow \frac{-\lambda_{1:t} - \tau_{1:t-1} + \sqrt{(\lambda_{1:t}+\tau_{1:t-1})^2 + \frac{G^2}{R^2}}}{2}$.
    Set $\eta_t \leftarrow 1/(\lambda_{1:t} + \tau_{1:t})$.
    Set $\mathbf{w}_{t+1} \leftarrow \Pi_S[\mathbf{w}_t - \eta_t \mathbf{g}_t]$.
    if $\|\mathbf{w}_{t+1}\| \geq R - \sqrt{\frac{2\epsilon}{\lambda}}$ then
      Set $R \leftarrow \sqrt{2}R$.
      Set CONVERGED $\leftarrow$ false.
      break
    end if
  end for
until CONVERGED
Choose $r$ uniformly at random from $\{1, \ldots, T\}$.
return $\mathbf{w}_r$.
```

logarithmic factors. In practice, however, Algorithm 3 can be significantly faster when $\|\mathbf{w}^*\| \ll \frac{1}{\sqrt{\lambda}}$. In these cases, the algorithm will tend to operate in the regime of small R, and will achieve $O(\frac{R^2G^2}{\delta^2\epsilon^2})$ regret, independent of λ.[8]

4. Batch proximal learning

In the batch learning setting, we are no longer presented with a sequence of objective functions but rather we are given a single λ-strongly convex objective function $f : \mathbb{R}^n \to \mathbb{R}$ that we would like to optimize. Batch algorithms are often appropriate when the training set is not particularly large, but the cost of inference with respect to any individual training example is high. This type of scenario occurs frequently in structured prediction problems, where inference may involve either a computationally expensive dynamic programming step, or even solving a combinatorial optimization problem as a subroutine.

[8]As an anecdotal example, on the "combined" dataset in our experiments, the parameter norm bound corresponding to the λ which gave the best test set performance was $\frac{1}{\sqrt{\lambda}} \approx 3 \times 10^5$, whereas $\|\mathbf{w}^*\| = 12.94$. On this run, the proximal algorithm estimated an upper bound of $R = 16$.

Algorithm 4 Proximal bundle method

```
Initialize $\mathbf{w}_1 \leftarrow \mathbf{0}$.
for $t \leftarrow 1, \ldots, T$ do
  Choose $\mathbf{a}_t \in \partial\ell(\mathbf{w}_t)$.
  Set $b_t \leftarrow \ell(\mathbf{w}_t) - \mathbf{a}_t^\mathsf{T}\mathbf{w}_t$.
  Set $\tau_t \leftarrow \frac{-\lambda t - \tau_{1:t-1} + \sqrt{(\lambda t + \tau_{1:t-1})^2 + \frac{(\lambda R + A_t)^2}{R^2}}}{2}$.
  Compute $\boldsymbol{\alpha}_t = \arg\max_{\boldsymbol{\alpha} \in \mathbb{R}^t : \boldsymbol{\alpha} \succeq \mathbf{0}, \boldsymbol{\alpha}^T\mathbf{1} \leq t} \mathcal{D}'_t(\boldsymbol{\alpha})$.
  Set $\mathbf{w}_{t+1} = \frac{\sum_{i=1}^t (\tau_i \mathbf{w}_i - \alpha_{t,i}\mathbf{a}_i)}{\lambda t + \tau_{1:t}}$.
end for
return $\mathbf{w}_{T+1}$.
```

The prototypical batch algorithm from which we start is the cutting plane optimization method of (Joachims, 2006) as reformulated and generalized in (Teo et al., 2007) and (Smola et al., 2008). In this method, f is assumed to be everywhere nonnegative, and one creates a sequence of lower-bound approximations to f of the form,

$$\mathcal{P}_t(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \max\left(0, \max_{i \in \{1,\ldots,t\}} \left(\mathbf{a}_i^\mathsf{T}\mathbf{w} + b_i\right)\right).$$

Initially, $\mathbf{w}_1 = \mathbf{0}$. During each iteration $t \in \{1, \ldots, T\}$, $\mathbf{a}_t$ and b_t are chosen so that $\mathbf{a}_t^\mathsf{T}\mathbf{w} + b_t$ is the first-order Taylor expansion of $\ell(\mathbf{w})$ at $\mathbf{w}_t$, and $\mathbf{w}_{t+1}$ is chosen to be the minimizer of $\mathcal{P}_t$. To date, the best convergence results known for bundle methods state that at most $O(\frac{1}{\lambda T})$ iterations are needed to achieve ϵ-optimality, as proved in (Teo et al., 2007) and (Smola et al., 2008). However, when $\lambda \approx 0$, the number of iterations needed can still be very large, just as in the online case.

To counter these problems, we propose a *proximal bundle method*, as shown in Algorithm 4. In particular, consider the sequence of primal and dual optimization problems,

$$\min_{\mathbf{w} \in \mathbb{R}^n} \mathcal{P}'_t(\mathbf{w}) \quad \text{for} \quad t = 1, 2, \ldots, T \qquad (10)$$

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^t : \boldsymbol{\alpha} \succeq \mathbf{0}, \boldsymbol{\alpha}^\mathsf{T}\mathbf{1} \leq t} \mathcal{D}'_t(\boldsymbol{\alpha}) \quad \text{for} \quad t = 1, 2, \ldots, T \qquad (11)$$

where

$$\mathcal{P}'_t(\mathbf{w}) := t \cdot \mathcal{P}_t(\mathbf{w}) + \sum_{i=1}^t \frac{\tau_i}{2}\|\mathbf{w} - \mathbf{w}_i\|^2$$

$$\mathcal{D}'_t(\boldsymbol{\alpha}) := \sum_{i=1}^t \left(\frac{\tau_i}{2}\|\mathbf{w}_i\|^2 + \alpha_i b_i\right) - \frac{\left\|\sum_{i=1}^t (\tau_i \mathbf{w}_i - \alpha_i \mathbf{a}_i)\right\|^2}{2(\lambda t + \tau_{1:t})}.$$

for some constants $\tau_1, \ldots, \tau_T$. As in the standard bundle algorithm, $\mathbf{w}_{t+1} := \arg\min_{\mathbf{w} \in \mathbb{R}^n} \mathcal{P}_t(\mathbf{w})$. If $\boldsymbol{\alpha}_t = \arg\max_{\boldsymbol{\alpha} \in \mathbb{R}^t : \boldsymbol{\alpha} \succeq \mathbf{0}, \boldsymbol{\alpha}^\mathsf{T}\mathbf{1} \leq t} \mathcal{D}'_t(\boldsymbol{\alpha})$, then the two optima are related by $\mathbf{w}_{t+1} = \frac{\sum_{i=1}^t (\tau_i \mathbf{w}_i - \alpha_{t,i}\mathbf{a}_i)}{\lambda t + \tau_{1:t}}$ using strong duality.

In most cutting plane analyses, convergence rates are established by lower-bounding the dual improvement in each

Table 1. Convergence of *Pegasos*, *Adaptive*, and *Proximal* on nine binary classification tasks. The second through fifth columns give the size of the training and testing sets, number of features, and the optimal regularization parameter. The last two sets of three columns report the best SVM training loss, $\hat{f} = \min_{t \in 1,\ldots,T} f(\mathbf{w}_t)$, seen for each tested algorithm, and the number of iterations needed to reduce the initial objective function by $0.99(f(\mathbf{w}_1) - \hat{f})$. n/a is reported for cases where the optimizer failed to find a better objective than the starting parameter set. The best numbers in each group are shown in bold.

Dataset	m_{train}	m_{test}	n	λ_{best}	Best training loss			Eff. iterations to convergence		
					Pegasos	*Adaptive*	*Proximal*	*Pegasos*	*Adaptive*	*Proximal*
a9a	32,561	16281	123	10^{-4}	0.3537	**0.3531**	0.3533	28	19	**18**
combined	78,823	19705	100	10^{-9}	0.5299	0.2760	**0.2336**	100	99	**8**
connect-4	54,045	13512	126	10^{-7}	6.8229	0.9698	**0.5136**	n/a	99	**63**
covtype	464,808	116204	54	10^{-8}	1.4852	0.7217	**0.5830**	n/a	96	**12**
ijcnn1	35,000	91701	22	10^{-7}	0.3582	0.2088	**0.1857**	89	98	**3**
mnist	60,000	10000	780	10^{-5}	0.1200	0.1033	**0.1012**	75	28	**3**
rcv1	20,242	677399	47,236	10^{-7}	0.0084	**0.0035**	0.0487	53	**10**	83
real-sim	57,846	14463	20,958	10^{-5}	**0.0602**	**0.0602**	**0.0602**	6	**5**	7
w8a	49,749	14951	300	10^{-8}	1.5146	0.1391	**0.1292**	n/a	45	**13**

iteration, and then arguing that only a limited number of iterations can occur before some termination criteria (e.g., primal-dual gap) is satisfied. Here, we again use the dual improvement argument, though we obtain somewhat different results, given that the dual objective function changes after each iteration due to the changing proximal regularization terms. Our analysis is closely related to the online learning framework of (Shalev-Shwartz & Kakade, 2009).

Lemma 3. *Let* $\mathbf{w}_1, \ldots, \mathbf{w}_{t-1} \in \mathbb{R}^n$ *and* $\boldsymbol{\alpha} \in \mathbb{R}^{t-1}$ *be vectors such that* $\boldsymbol{\alpha} \succeq \mathbf{0}$ *and* $\boldsymbol{\alpha}^T \mathbf{1} \leq t-1$. *If we define* $\mathbf{w}_t := \frac{\sum_{i=1}^{t-1} (\tau_i \mathbf{w}_i - \alpha_i \mathbf{a}_i)}{\lambda(t-1) + \tau_{1:t-1}}$, *then*

$$\mathcal{D}'_t([\boldsymbol{\alpha}; 1]) - \mathcal{D}'_{t-1}(\boldsymbol{\alpha}) = f(\mathbf{w}_t) - \frac{\|\lambda \mathbf{w}_t + \mathbf{a}_t\|^2}{2(\lambda t + \tau_{1:t})}. \quad (12)$$

Using this lower bound, we can then bound the best suboptimality obtained by our algorithm after t steps:

Proposition 3. *Let* $\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w})$. *Suppose that* $\|\mathbf{w}_t\| \leq R$ *and* $\|\mathbf{a}_t\| \leq A_t$ *for* $t = 1, \ldots, T$. *Then,*

$$\min_{t \in \{1,\ldots,T\}} f(\mathbf{w}_t) - f(\mathbf{w}^*) \leq \frac{1}{T} \sum_{t=1}^{T} \left[2\tau_t R^2 + \frac{(\lambda R + A_t)^2}{2(\lambda t + \tau_{1:t})} \right].$$

Remarkably, the suboptimality guarantees in the proposition above have essentially the same form as the regret bounds stated in Corollary 1. As a result, we can make use of the balancing heuristic for choosing the proximal constants $\tau_1, \ldots, \tau_T$. Furthermore, the optimistic strategy for bounding the optimal parameter norm, as described in Section 3.4, also carries over with little modification. For the sake of space, we show only the proximal bundle method using the balancing heuristic in Algorithm 4; we do not give pseudocode for the optimistic extension explicitly. Using Proposition 3 and the argument in Theorem 1, we have the following theorem,

Theorem 5. *Suppose that* $\|\mathbf{w}_t\| \leq R$ *and* $\|\mathbf{a}_t\| \leq A$ *for* $t = 1, \ldots, T$. *Then, Algorithm 4 achieves,*

$$\min_{t \in \{1,\ldots,T\}} f(\mathbf{w}_t) - f(\mathbf{w}^*) \leq \frac{(\lambda R + A)^2 (1 + \log T)}{\lambda T}.$$

Provided that $\lambda R + A = \tilde{O}(1)$, then our analysis yields a worst case convergence rate of $\tilde{O}(\frac{1}{\lambda T})$, matching the convergence rate of our online algorithm, as well as the best known convergence rates for bundle methods.

We note that the idea of stabilizing standard bundle method algorithms to improve convergence has been suggested previously in the bundle method literature. Proximal bundle methods originated with (Kiwiel, 1983), and are closely related to trust region (Schramm & Zowe, 1992) and level set (Lemaréchal et al., 1995) techniques for bundle method improvement. In practice, each of these prior methods require considerable parameter tuning on the part of the user. In contrast, our bundle algorithm is straightforward, with the curvature terms τ_t automatically chosen in order to minimize a regret bound.

5. Experiments

We carried out two sets of experiments with proximal algorithms. For the first set of tasks, we tested online algorithms for large-scale binary classification. For the second set of tasks, we performed batch training of structured output SVMs for RNA folding and web ranking. In both the online and batch cases, we ran the optimistic version of our proximal algorithm, setting $\epsilon = 0$ and $\delta = 1$, stopping after a fixed number of iterations, and returning $\mathbf{w}_{T+1}$ instead of $\mathbf{w}_r$, as in (Shalev-Shwartz et al., 2007).

5.1. Online learning with binary classification

In this experiment, we tested the behavior of our algorithm on nine binary classification datasets.[9] For each of these datasets, we first determined the optimal setting of λ_{best} for ensuring good generalization performance using cross-validation. We then compared the *Proximal* online al-

[9]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/. For each dataset where a binary classification version was not available, we reduced multiclass to a single class vs. rest problem. When separate testing sets were not available, we reserved 80% of the data set for training and 20% for testing.

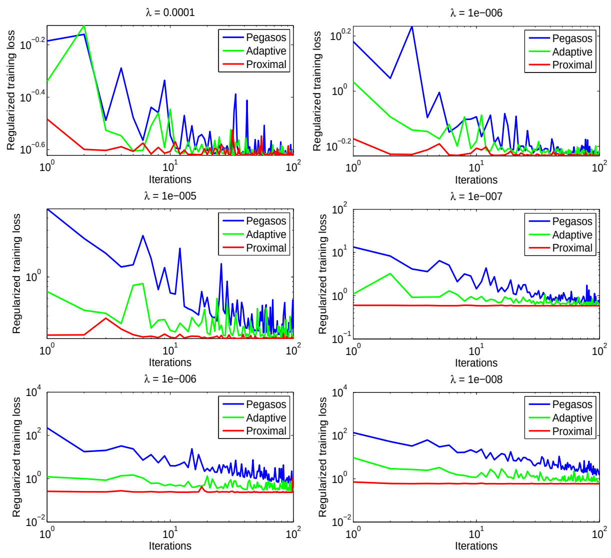

Figure 1. Convergence of *Pegasos*, *Adaptive*, and *Proximal* for combined and covtype. Left column: combined; Right column: covtype. Each row corresponds to a regularization parameter λ. Bottom row: $\lambda = \lambda_{\text{best}}$, middle row: $\lambda = 10\lambda_{\text{best}}$, top row: $\lambda = 100\lambda_{\text{best}}$. Effective iterations are shown on the x-axis.

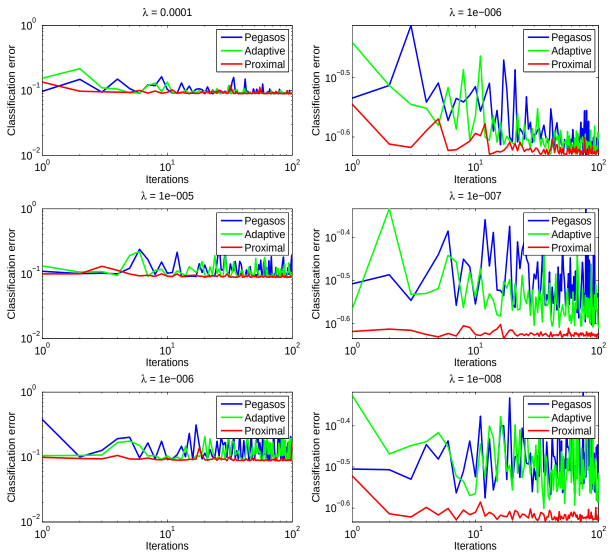

Figure 2. Test errors of *Pegasos*, *Adaptive*, and *Proximal* for combined and covtype during the course of optimization. Each row corresponds to a regularization parameter λ. Bottom row: $\lambda = \lambda_{\text{best}}$, middle row: $\lambda = 10\lambda_{\text{best}}$, top row: $\lambda = 100\lambda_{\text{best}}$. Effective iterations are shown on the x-axis.

gorithm against *Pegasos* (Shalev-Shwartz et al., 2007) and *Adaptive* online gradient descent (Bartlett et al., 2008) by running each algorithm for 100 effective iterations[10] under a variety of regularization parameter settings.

In Table 1, we provide some statistics on the training and testing datasets used. We record the best objective value $\hat{f} = \min_{t \in 1,\ldots,T} f(\mathbf{w}_t)$ obtained for each algorithm over the 100 effective iterations. We also record the number of iterations needed to achieve a objective function reduction of $0.99(f(\mathbf{w}_1) - \hat{f})$ for each algorithm. The *Proximal* algorithm achieves the best objective on 7 out of 9 datasets, while consistently requiring few effective iterations.

Figures 1 and 2 show learning curve plots and test error plots for two of the datasets (combined and covtype). As shown, the proximal algorithm enjoys a comfortable advantage over the other methods, especially for small λ_{best}.

5.2. Batch learning with RNA folding and web ranking

In this experiment, we compared our batch proximal learning algorithm against standard bundle algorithms (Smola et al., 2008) for learning RNA folding and web search ranking models. Both of these problems can be formulated as nonsmooth structured SVMs ((Chapelle et al., 2007) for ranking and (Do et al., 2006) for RNA folding). To date, the fastest approaches for dealing with this type of nonsmooth optimization problem are cutting plane/bundle methods (e.g., SVMPerf (Joachims, 2006) and BMRM (Teo et al., 2007)).

[10]One pass through the entire dataset is an effective iteration.

In the RNA folding experiment, the dataset contained RNA sequences taken from 151 separate RNA families (Do et al., 2006), and we used a model with approximately 350 distinct features based largely on existing thermodynamic scoring schemes for RNA folding. In the ranking experiment, the dataset contained 1000 queries for training, 1000 queries for validation, with an average of 50 documents per query. In both cases, we compared the performance of the proximal bundle method against the standard bundle method for various values of λ.

Figure 3 shows training loss curves depicting the best training loss obtained so far for a standard bundle method compared to our proximal variant. In both methods, many iterations pass before the algorithms are able to identify parameters which improve upon the initial parameter set; for the standard bundle method, this problem is especially pronounced for small regularization parameters. Again, the results show that the proximal variant significantly outperforms the standard algorithm, especially when λ is small.

6. Discussion

Functions with low curvature are the Achilles's heel of optimization algorithms in machine learning. In this paper, we propose new online and batch learning algorithms, which sequentially modify the objective functions used during optimization. By choosing these modified tasks carefully, our methods ensure that (1) the sequence of solutions given by these modified tasks will lead to a good approximate minimizer of the original optimization problem, and (2) the regret bounds obtained in the online setting and

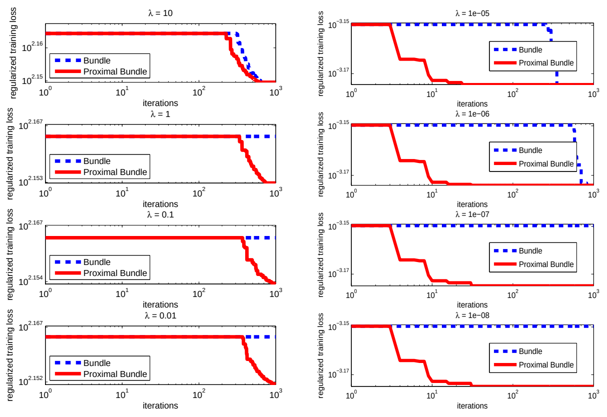

Figure 3. Comparison of a standard bundle method to our proximal bundle method for SVM structured learning with various choices of λ. Left column: RNA folding; Right column: Web ranking. Each row corresponds to a regularization parameter λ. The regularization parameter decreases from top to bottom.

the convergence rates obtained in the batch setting will be improved due to the increased curvature.

The idea of adding curvature in order to improve regret bounds for online algorithms was introduced in (Bartlett et al., 2008), and the online algorithmic schemes proposed there have much in common with the basic online methods proposed here. We apply these ideas to the problem of training linear SVMs and structured prediction models, where we introduce a new adaptive strategy for optimistically bounding the norm of the optimal parameters. We also transfer these ideas to the batch setting, where we present improved bundle methods for structured learning.

Experimentally, we show that the problem of low curvature is not simply a matter of theoretical concern. Rather, for many real world large-scale learning problems, the optimal regularization penalty (as determined by holdout cross-validation) is often very small. For problems where high regularization is appropriate (e.g., when the dimensionality of the data is large relative to the number of training examples), our algorithm performs as well as the best existing methods, such as Pegasos. When low regularization is needed, however, our algorithm offers dramatic improvements over state-of-the-art techniques, converging in a few passes through the dataset when other algorithms may fail to converge at all.

Acknowledgments

We thank Andrew Ng for his support in this project. We thank anonymous reviewers for their comments on the manuscript. CBD was supported by an NSF Graduate Research Fellowship. CSF was supported by an A*STAR National Science Scholarship.

References

Abernethy, J., Bartlett, P. L., Rakhlin, A., & Tewari, A. (2008). Optimal strategies and minimax lower bounds for online convex games. *Proceedings of the 21st Annual Conference on Computational Learning Theory*.

Bartlett, P., Hazan, E., & Rakhlin, A. (2008). Adaptive online gradient descent. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in Neural Information Processing Systems 20*, 65–72. MIT Press.

Chapelle, O., Le, Q. V., & Smola, A. J. (2007). Large margin optimization of ranking measures. *NIPS Workshop: Machine Learning for Web Search*.

Do, C. B., Woods, D. A., & Batzoglou, S. (2006). CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, *22*, e90–e98.

Hazan, E., Agarwal, A., & Kale, S. (2007). Logarithmic regret algorithms for online convex optimization. *Mach Learn*, *69*, 169–192.

Joachims, T. (2006). Training linear SVMs in linear time. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 217–226).

Kiwiel, K. C. (1983). Proximity control in bundle methods for convex nondifferentiable minimization. *Math Program*, *27*, 320–341.

Lemaréchal, C., Nemirovskii, A., & Nesterov, Y. (1995). New variants of bundle methods. *Math Program*, *69*, 111–147.

Schramm, H., & Zowe, J. (1992). A version of the bundle idea for minimizing a nonsmooth function: conceptual idea, convergence analysis, numerical results. *SIAM J Optim*, *2*, 121–152.

Shalev-Shwartz, S., & Kakade, S. M. (2009). Mind the duality gap: Logarithmic regret algorithms for online optimization. In D. Koller, D. Schuurmans, Y. Bengio and L. Bottou (Eds.), *Advances in Neural Information Processing Systems 21*, 1457–1464. MIT Press.

Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. *Proceedings of the 24th Annual International Conference on Machine Learning* (pp. 807–814).

Smola, A., Vishwanathan, S. V. N., & Le, Q. (2008). Bundle methods for machine learning. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in Neural Information Processing Systems 20*, 1377–1384. MIT Press.

Teo, C. H., Smola, A., Vishwanathan, S. V., & Le, Q. V. (2007). A scalable modular convex solver for regularized risk minimization. *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 727–736).

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *Proceedings of the 20th Annual International Conference on Machine Learning*.