Stochastic Methods for ℓ_1 Regularized Loss Minimization

Shai Shalev-Shwartz SHAI@TTI-C.ORG
Ambuj Tewari TEWARI@TTI-C.ORG
Toyota Technological Institute at Chicago, 6045 S. Kenwood Ave., Chicago, IL 60637, USA

Abstract

We describe and analyze two stochastic methods for ℓ_1 regularized loss minimization problems, such as the Lasso. The first method updates the weight of a single feature at each iteration while the second method updates the entire weight vector but only uses a single training example at each iteration. In both methods, the choice of feature/example is uniformly at random. Our theoretical runtime analysis suggests that the stochastic methods should outperform state-of-the-art deterministic approaches, including their deterministic counterparts, when the size of the problem is large. We demonstrate the advantage of stochastic methods by experimenting with synthetic and natural data sets.

1. Introduction

We present optimization procedures for solving problems of the form:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{m}\sum_{i=1}^{m} L(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i) + \lambda\|\mathbf{w}\|_1 \ , \qquad (1)$$

where $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m) \in ([-1,+1]^d \times \mathcal{Y})^m$ is a sequence of training examples, $L : \mathbb{R}^d \times \mathcal{Y} \to [0,\infty)$ is a non-negative loss function, and $\lambda > 0$ is a regularization parameter. This generic problem includes as special cases the Lasso (Tibshirani, 1996), in which $L(a,y) = \frac{1}{2}(a-y)^2$, and logistic regression, in which $L(a,y) = \log(1+\exp(-ya))$.

Our methods can also be adapted to deal with additional boxed constraints of the form $w_i \in [a_i, b_i]$, which enables us to utilize them for solving the dual problem of Support Vector Machine (Cristianini & Shawe-Taylor, 2000). For concreteness, and due to the lack of space, we focus on the formulation given in Eq. (1).

Appearing in *Proceedings of the 26^{th} International Conference on Machine Learning*, Montreal, Canada, 2009.

Throughout the paper, we assume that L is convex w.r.t. its first argument. This implies that Eq. (1) is a convex optimization problem, and therefore can be solved using standard optimization techniques, such as interior point methods. However, standard methods scale poorly with the size of the problem (i.e. m and d). In recent years, machine learning methods are proliferating in data-laden domains such as text and web processing in which data sets of millions of training examples or features are not uncommon. Since traditional methods for solving Eq. (1) generally scale very poorly with the size of the problem, their usage is inappropriate for data-laden domains. In this paper, we discuss how to overcome this difficulty using stochastic methods. We describe and analyze two practical methods for solving Eq. (1) even when the size of the problem is very large.

The first method we propose is a stochastic version of the familiar coordinate descent approach. The coordinate descent approach for solving ℓ_1 regularized problems is not new (as we survey below in Section 1.1). At each iteration of coordinate descent, a single element of $\mathbf{w}$ is updated. The only twist we propose here regarding the way one should choose the next feature to update. We suggest to choose features uniformly at random from the set $[d] = \{1, \ldots, d\}$. This simple modification enables us to show that the runtime required to achieve ϵ (expected) accuracy is upper bounded by

$$\frac{m\, d\, \beta\, \|\mathbf{w}^\star\|_2^2}{\epsilon} \ , \qquad (2)$$

where β is a constant which only depends on the loss function (e.g. $\beta = 1$ for the quadratic loss function) and $\mathbf{w}^\star$ is the optimal solution. This bound tells us that the runtime grows only linearly with the size of the problem. Furthermore, the stochastic method we propose is parameters-free and is very simple to implement.

Another well known stochastic method, which has been successfully applied for loss minimization problems, is stochastic gradient descent (e.g. Bottou & LeCunn, 2005; Shalev-Shwartz et al., 2007). In stochastic gradient descent, at each iteration we pick one example from the training set, uniformly at random, and update the weight vec-

tor based on the chosen example. The attractiveness of stochastic gradient descent methods is that their runtime do not depend at all on the number of examples, and can even sometime decrease with the number of examples (see Bottou & Bousquet, 2008; Shalev-Shwartz & Srebro, 2008). Regretfully, the stochastic gradient descent method fails to produce sparse solutions, which makes the algorithm both slower and less attractive as sparsity is one of the major reasons to use ℓ_1 regularization. To overcome this obstacle, two variants were recently proposed. First, Duchi et al., 2008 suggested to replace the ℓ_1 regularization term with a constraint of the form $\|\mathbf{w}\|_1 \leq B$, and then to use stochastic gradient projection procedure. Another solution, which uses the regularization form given in Eq. (1), has been proposed by Langford et al., 2009 and is called truncated gradient descent. In this approach, the elements of $\mathbf{w}$ which cross 0 after the stochastic gradient step are truncated to 0, hence sparsity is achieved. The disadvantage of both Duchi et al., 2008 and Langford et al., 2009 methods is that in some situations, their runtime might grow quadratically with the dimension d, even if the optimal predictor $\mathbf{w}^\star$ is very sparse (see Section 1.1 below for details). This quadratic dependence on d can be avoided if one uses mirror descent updates (Beck & Teboulle, 2003) such as the exponentiated gradient approach (Littlestone, 1988; Kivinen & Warmuth, 1997; Beck & Teboulle, 2003). However, this approach again fails to produce sparse solutions. In this paper, we combine the idea of truncating the gradient (Langford et al., 2009) with another variant of stochastic mirror descent, which is based on p-norm updates (Grove et al., 2001; Gentile, 2003). The resulting algorithm both produces sparse solutions and has $\tilde{O}(d)$ dependence on the dimension. We call the algorithm SMIDAS for "Stochastic MIrror Descent Algorithm made Sparse".

We provide runtime guarantees for SMIDAS as well. In particular, for the logistic-loss and the squared-loss we obtain the following upper bound on the runtime to achieving ϵ expected accuracy:

$$O\left(\frac{d\,\log(d)\,\|\mathbf{w}^\star\|_1^2}{\epsilon^2}\right)\,. \tag{3}$$

Comparing the above with the runtime bound of the stochastic coordinate descent method given in Eq. (2) we note three major differences. First, while the bound in Eq. (2) depends on the number of examples, m, the runtime of SMIDAS does not depend on m at all. On the flip side, the dependence of stochastic coordinate descent on the dimension is better both because the lack of the term $\log(d)$ and because $\|\mathbf{w}^\star\|_2^2$ is always smaller than $\|\mathbf{w}^\star\|_1^2$ (the ratio is at most d). Last, the dependence on $\frac{1}{\epsilon}$ is linear in Eq. (2) and quadratic in Eq. (3). If ϵ is the same order as the objective value at $\mathbf{w}^\star$ it is possible to improve the dependence on $1/\epsilon$. We omit the better bound due to lack of space. Finally, we would like to point out that while the stochastic coordinate descent method is parameters-free, the success of SMIDAS and of the method of Langford et al., 2009, depends on a careful tuning of a learning rate parameter.

1.1. Related Work

We now survey several existing methods and in particular show how our stochastic twist enables us to give superior runtime guarantees.

Coordinate descent methods for ℓ_1 regularization Following the Gauss-Siedel approach of Zhang & Oles, 2001, Genkin et al., 2007 described a coordinate descent method (called BBR) for minimizing ℓ_1 regularized objectives. This approach is similar to our method, with three main differences. First, and most important, at each iteration we choose a coordinate uniformly at random. This allows us to provide theoretical runtime guarantees. We note that no theoretical guarantees are provided by Zhang & Oles, 2001; Genkin et al., 2007. Second, we solely use gradient information which makes our algorithm parameters-free and extremely simple to implement. In contrast, the Gauss-Siedel approach is more complicated and involves second order information, or a line search procedure, or a trusted region Newton step. Last, the generality of our derivation allows us to tackle a more general problem. For example, it is easy to deal with additional boxed constraints. Friedman et al., 2008 generalized the approach of Genkin et al., 2007 to include the case of elastic-net regularization. In a series of experiments, they observed that cyclical coordinate descent outperforms many alternative popular methods such as LARS (Efron et al., 2004), an interior point method called `l1lognet` (Koh et al., 2007), and the Lasso Penalized Logistic (LPL) program (Wu & Lange, 2008). However, no theoretical guarantees are provided in Friedman et al., 2008 as well. Our analysis can partially explain the experimental result of Friedman et al., 2008 since updating the coordinates in a cyclical order can in practice be very similar to stochastic updates.

Luo & Tseng, 1992 established a linear convergence result for coordinate descent algorithms. This convergence result tells us that after an unspecified number of iterations, the algorithm converges very fast to the optimal solution. However, this analysis is useless in data laden domains as it can be shown that the initial unspecified number of iterations depends at least quadratically on the number of training examples. In an attempt to improve the dependence on the size of the problem, Tseng & Yun, 2009 recently studied other variants of block coordinate descent for optimizing 'smooth plus separable' objectives. In particular, ℓ_1 regularized loss minimization (Eq. (1)) is of this form, provided that the loss function is smooth. The algorithm proposed by Tseng & Yun, 2009 is not stochastic. Translated to our notation, the runtime bound given in Tseng &

Yun, 2009 is of order[1] $\frac{m\,d^2\,\beta\,\|\mathbf{w}^\star\|_2^2}{\epsilon}$. This bound is inferior to our runtime bound for stochastic coordinate descent given in Eq. (2) by a factor of the dimension d.

Coordinate descent methods for ℓ_1 domain constraints A different, but related, optimization problem is to minimize the loss, $\frac{1}{m}\sum_i L(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i)$, subject to a domain constraint of the form $\|\mathbf{w}\|_1 \leq B$. Many authors presented a forward greedy selection algorithm (a.k.a. Boosting) for this problem. We refer the reader to (Frank & Wolfe, 1956; Zhang, 2003; Clarkson, 2008). These authors derived the upper bound $O\left(\beta\,\|\mathbf{w}^\star\|_1^2/\epsilon\right)$ on the number of iterations required by this algorithm to find an ϵ-accurate solution. Since at each iteration of the algorithm, one needs to calculate the gradient of the loss at $\mathbf{w}$, the runtime of each iteration is $m\,d$. Therefore, the total runtime becomes $O\left(m\,d\,\beta\,\|\mathbf{w}^\star\|_1^2/\epsilon\right)$. Note that this bound is better than the bound given by Tseng & Yun, 2009, since for any vector in $\mathbb{R}^d$ we have $\|\mathbf{w}\|_1 \leq \sqrt{d}\,\|\mathbf{w}\|_2$. However, the boosting bound given above is still inferior to our bound given in Eq. (2) since $\|\mathbf{w}^\star\|_1 \geq \|\mathbf{w}^\star\|_2$. Furthermore, in the extreme case we have $\|\mathbf{w}^\star\|_1^2 = d\|\mathbf{w}^\star\|_2^2$, thus our bound can be better than the boosting bound by a factor of d. It is possible to show (proof is omitted due to lack of space) that the *iteration* bound (not runtime) of any algorithm cannot be smaller than $\Omega(\|\mathbf{w}^\star\|_1^2/\epsilon)$. This seems to imply that *any* deterministic method, which goes over the entire data at each iteration, will induce a runtime which is inferior to the runtime we derive for stochastic coordinate descent.

Stochastic Gradient Descent and Mirror Descent Stochastic gradient descent (SGD) is considered to be one of the best methods for large scale loss minimization, when we measure how fast a method achieves a certain generalization error. This has been observed in experiments (Bottou, Web Page) and also has been analyzed theoretically by Bottou & Bousquet, 2008; Shalev-Shwartz & Srebro, 2008.

As mentioned before, one can apply SGD for solving Eq. (1), however, SGD fails to produce sparse solutions. Langford et al., 2009 proposed an elegant simple modification of the SGD update rule, which yields a variant of SGD with sparse intermediate solutions. They also provide bounds on the runtime of the resulting algorithm. In the general case (i.e. without assuming low objective relative to ϵ), their analysis implies the following runtime bound

$$O\left(\frac{d\,\|\mathbf{w}^\star\|_2^2\,X_2^2}{\epsilon^2}\right)\,, \tag{4}$$

where $X_2^2 = \frac{1}{m}\sum_i \|\mathbf{x}_i\|_2^2$ is the average squared norm of an instance. Comparing this bound with our bound in Eq. (3) we observe that none of the bounds dominates the other, and their relative performance depends on properties of the training set and the optimal solution $\mathbf{w}^\star$. Specifically, if $\mathbf{w}^\star$ has only $k \ll d$ non-zero elements and each $\mathbf{x}_i$ is dense (say $\mathbf{x}_i \in \{-1,+1\}^d$), then the ratio between the above bound of SGD and the bound in Eq. (3) becomes $\frac{d}{k\,\log(d)} \gg 1$. On the other hand, if $\mathbf{x}_i$ has only k non-zeros while $\mathbf{w}^\star$ is dense, then the ratio between the bounds can be $\frac{k}{d\,log(d)} \ll 1$. Although the relative performance is data dependent, in most applications if one prefers ℓ_1 regularization over ℓ_2 regularization, he should also believe that $\mathbf{w}^\star$ is sparse, and thus our runtime bound in Eq. (3) is likely to be superior[2].

The reader familiar with the online learning and mirror descent literature will not be surprised by the above discussion. Bounds that involved $\|\mathbf{w}^\star\|_1$ and $\|\mathbf{x}_i\|_\infty$, as in Eq. (3), are well known and the relative performance discussed above was pointed out in the context of additive vs. multiplicative updates (see e.g. Kivinen & Warmuth, 1997). However, the most popular algorithm for obtaining bounds of the form given in Eq. (3) is the EG approach (Kivinen & Warmuth, 1997), which involves the exponential potential, and this algorithm cannot yield intermediate sparse solutions. One of the contributions of this paper is to show that with a different potential, which is called the p-norm potential, one can obtain the bound given in Eq. (3) while still enjoying sparse intermediate solutions.

2. Stochastic Coordinate Descent

In this section we describe and analyze a simple stochastic coordinate descent algorithm for solving ℓ_1 regularized problems of the form given in Eq. (1). We first present the following equivalent optimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}^{2d}} \frac{1}{m}\sum_{i=1}^{m} L(\langle \mathbf{w}, \hat{\mathbf{x}}_i\rangle, y_i) + \lambda \sum_{i=1}^{2d} w_i \;\text{ s.t. } \mathbf{w} \geq \mathbf{0}\,, \tag{5}$$

where $\hat{\mathbf{x}}_i = [\mathbf{x}_i\,;\,-\mathbf{x}_i]$. It is easy to verify that if $\mathbf{v}^\star \in \mathbb{R}^{2d}$ minimizes Eq. (5) then $\mathbf{w}^\star \in \mathbb{R}^d$ defined by $w_i^\star = v_{d+i}^\star - v_i^\star$ minimizes Eq. (1). Furthermore, if the objective of Eq. (5) at $\mathbf{v}$ is at most ϵ away from the optimum of Eq. (5), then the objective of Eq. (1) at $\mathbf{w}$, s.t. $w_i = v_{d+i} - v_i$, is at most ϵ away from the optimum of Eq. (1). Therefore, it suffices to present an algorithm for approximately solving Eq. (5) and the output of the algorithm immediately translates to an approximate solution of Eq. (1).

To further simplify the presentation and for the purpose of generality, we derive and analyze the algorithm for opti-

[1] To see this, note that the iterations bound in Equation (21) of Tseng & Yun, 2009 is: $\frac{\beta\,\|\mathbf{w}^\star\|_2^2}{\epsilon\,\nu}$, and using Equation (25) in Section 6, we can set the value of ν to be $\nu = 1/d$ (since in our case there are no linear constraints). The complexity bound now follows from the fact that the cost of each iteration is $O(d\,m)$.

[2] One important exception is the large scale text processing application described in Langford et al., 2009 where the dimension is so large and ℓ_1 is used simply because we cannot store a dense weight vector in memory.

mization problems of the form:

$$\min_{\mathbf{w}\in\mathbb{R}^{2d}} \; R(\mathbf{w}) \;\; \text{s.t. } \mathbf{w} \geq \mathbf{0} \;, \tag{6}$$

where $R : \mathbb{R}^{2d} \to \mathbb{R}$. If convenient, one may think about R as being the function

$$R(\mathbf{w}) = \frac{1}{m}\sum_{i=1}^{m} L(\langle \mathbf{w}, \hat{\mathbf{x}}_i\rangle, y_i) + \lambda \sum_{i=1}^{2d} w_i \;. \tag{7}$$

We are now ready to present the stochastic coordinate descent algorithm. The algorithm initializes $\mathbf{w}$ to be $\mathbf{0}$. At each iteration, we pick a coordinate j uniformly at random from $[2d]$. Then, the derivative of $R(\mathbf{w})$ w.r.t. the jth element of $\mathbf{w}$, $g_j = (\nabla R(\mathbf{w}))_j$, is calculated. For example, if $R(\mathbf{w})$ is as defined in Eq. (7) then $g_j = \frac{1}{m}\sum_{i=1}^{m} L'(\langle \mathbf{w}, \hat{\mathbf{x}}_i\rangle, y_i)\hat{x}_{i,j} + \lambda$, where L' is the derivative of the loss function w.r.t. its first argument. Simple calculus yields

$$L'(a,y) = \begin{cases} (a-y) & \text{for squared-loss} \\ \frac{-y}{1+\exp(a\,y)} & \text{for logistic-loss} \end{cases} \tag{8}$$

Next, a step size is determined based on the value of g_j and a parameter of the loss function denoted β. This parameter is an upper bound on the second derivative of the loss. Again, for our running examples we have

$$\beta = \begin{cases} 1 & \text{for squared-loss} \\ 1/4 & \text{for logistic-loss} \end{cases} \tag{9}$$

The step size is trimmed so as to ensure that after performing the update, the constraint $w_j \geq 0$ will not be violated. Finally, we update w_j according to the calculated step size.

Algorithm 1 Stochastic Coordinate Descent (SCD)

let $\mathbf{w} = \mathbf{0}$
for $t = 1, 2, \ldots$
  sample j uniformly at random from $\{1, \ldots, 2d\}$
  let $g_j = (\nabla R(\mathbf{w}))_j$
  let $\eta = \max\{-w_j \,,\, -g_j/\beta\}$
  let $w_j = w_j + \eta$
end

2.1. Efficient Implementation

We now present an efficient implementation of Algorithm 1, assuming that R is of the form given in Eq. (7). The simple idea is to maintain a vector $\mathbf{z} \in \mathbb{R}^m$ such that $z_i = \langle \mathbf{w}, \mathbf{x}_i\rangle$. Once we have this vector, calculating g_j on average requires $O(s\,m)$ iterations, where

$$s = \frac{|\{(i,j) : \hat{x}_{i,j} \neq 0\}|}{m\,d} \tag{10}$$

is the average number of non-zeros in our training set. Concretely, we obtain the following procedure for logistic-loss and squared-loss.

Algorithm 2 SCD for Logistic-loss and squared-loss

let $\mathbf{w} = \mathbf{0} \in \mathbb{R}^{2d}$, $\mathbf{z} = \mathbf{0} \in \mathbb{R}^m$
for $t = 1, 2, \ldots$
  sample j uniformly at random from $\{1, \ldots, 2d\}$
  let L' and β be as defined in Eq. (8) and Eq. (9)
  let $g_j = \frac{1}{m}\sum_{i:\hat{x}_{i,j}\neq 0} L'(z_i, y_i)\hat{x}_{i,j} + \lambda$
  let $\eta = \max\{-w_j \,,\, -g_j/\beta\}$
  let $w_j = w_j + \eta$
  $\forall i$ s.t. $\hat{x}_{i,j} \neq 0$ let $z_i = z_i + \eta\hat{x}_{i,j}$
end

2.2. Runtime Guarantee

Theorem 1 *Let $R(\mathbf{w}) : \mathbb{R}^{2d} \to \mathbb{R}$ be a convex objective function and assume that there exists $\beta > 0$ such that for all vector $\mathbf{w}$, scalar η and index j we have*

$$R(\mathbf{w} + \eta\mathbf{e}^j) \leq R(\mathbf{w}) + \eta(\nabla R(\mathbf{w}))_j + \tfrac{\beta\,\eta^2}{2} \;.$$

Let $\mathbf{w}^\star$ be a minimizer of Eq. (6) and let $\mathbf{w}_o$ be the output of Algorithm 1 after performing T iterations. Then,

$$\mathbb{E}[R(\mathbf{w}_o)] - R(\mathbf{w}^\star) \;\leq\; \frac{d\,\left(\beta\,\|\mathbf{w}^\star\|_2^2 + 2R(\mathbf{0})\right)}{2T} \;.$$

Proof Sketch We define the following 'double potential': $\Psi(\mathbf{w}) = \frac{\beta}{2}\|\mathbf{w} - \mathbf{w}^\star\|_2^2 + R(\mathbf{w})$. The main component of the proof is to show that for any vector $\mathbf{w}$ and index j, if η is defined as in Algorithm 1 we have $\Psi(\mathbf{w}) - \Psi(\mathbf{w} + \eta\mathbf{e}^j) \geq (w_j - w_j^\star)g_j$. We show this using the assumption on the smoothness of R and algebraic manipulations. Denote by $\mathbf{w}_t$ the value of $\mathbf{w}$ at iteration t of the algorithm. Taking expectation of the aforementioned inequality w.r.t. the choice of j we obtain that $\mathbb{E}[\Psi(\mathbf{w}_t) - \Psi(\mathbf{w}_{t+1})] \geq \frac{1}{d}\mathbb{E}[\langle \mathbf{w}_t - \mathbf{w}^\star, \nabla R(\mathbf{w}_t)\rangle]$. But since R is convex, the right-hand side of the above upper bounds $\mathbb{E}[\epsilon_t]/d$, where $\epsilon_t = R(\mathbf{w}_t) - R(\mathbf{w}^\star)$ is the sub-optimality at iteration t. Summing over t and using the fact that ϵ_t is a monotonically non-increasing sequence we obtain $T\mathbb{E}[\epsilon_{T+1}] \leq \mathbb{E}[\sum_t \epsilon_t] \leq d(\Psi(\mathbf{w}_1) - \Psi(\mathbf{w}_{T+1}))$, which concludes our proof. ■

Next, we specify the runtime bound for the case of ℓ_1 regularized logistic-regression and squared-loss. First, it is easy to show that for R as defined in Eq. (7), if the second derivative of L is bounded by β then the condition on R given in Theorem 1 holds. Additionally, for the logistic-loss we have $R(\mathbf{0}) \leq 1$. Therefore, for logistic-loss, after performing $\frac{d\,(\frac{1}{4}\,\|\mathbf{w}^\star\|_2^2 + 2)}{\epsilon}$ iterations of Algorithm 2 we have that $\mathbb{E}[R(\mathbf{w}_o)] - R(\mathbf{w}^\star) \leq \epsilon$. Since the average cost of each iteration is $s\,m$, where s is as defined in Eq. (10), we end up with the total runtime $\frac{s\,m\,d\,(\frac{1}{4}\,\|\mathbf{w}^\star\|_2^2 + 2)}{\epsilon}$. For the squared-loss we have $R(\mathbf{0}) = \frac{1}{m}\sum_i y_i^2$. Assuming

that the targets are normalized so that $R(\mathbf{0}) \leq 1$, and using similar derivation we obtain the total runtime bound $\frac{s\,m\,d\,(\|\mathbf{w}^\star\|_2^2+2)}{\epsilon}$.

3. Stochastic Mirror Descent made Sparse

In this section we describe our mirror descent approach for ℓ_1 regularized loss minimization, which maintains intermediate sparse solutions. To simplify the notation throughout this section, we rewrite the problem in Eq. (1) using the notation

$$\min_{\mathbf{w}\in\mathbb{R}^d} \overbrace{\underbrace{\frac{1}{m}\sum_{i=1}^{m} L(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i)}_{\equiv C(\mathbf{w})} + \lambda\|\mathbf{w}\|_1}^{\equiv P(\mathbf{w})} . \qquad (11)$$

Mirror descent algorithms maintain two weight vectors: primal $\mathbf{w}$ and dual $\boldsymbol{\theta}$. The connection between the two vectors is via a link function $\boldsymbol{\theta} = f(\mathbf{w})$, where $f : \mathbb{R}^d \to \mathbb{R}^d$. The link function is invertible, and therefore $\mathbf{w} = f^{-1}(\boldsymbol{\theta})$. In our mirror descent variant, we use the p-norm link function. That is, the jth element of f is $f_j(\mathbf{w}) = (\text{sign}(\mathbf{w}_j)\,|\mathbf{w}_j|^{q-1})/\|\mathbf{w}\|_q^{q-2}$, where $\|\mathbf{w}\|_q = (\sum_j |w_j|^q)^{1/q}$ and $q = p/(p-1)$. The inverse function is (see e.g. Gentile, 2003)

$$f_j^{-1}(\boldsymbol{\theta}) = \frac{\text{sign}(\theta_j)\,|\theta_j|^{p-1}}{\|\boldsymbol{\theta}\|_p^{p-2}} . \qquad (12)$$

We first describe how mirror descent algorithms can be applied to the objective $C(\mathbf{w})$ without the ℓ_1 regularization term. At each iteration of the algorithm, we first sample a training example i uniformly at random from $\{1,\ldots,m\}$. We then estimate the gradient of $C(\mathbf{w})$ by calculating the vector: $\mathbf{v} = L'(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i)\,\mathbf{x}_i$. Note that the expectation of $\mathbf{v}$ over the random choice of i is $\mathbb{E}[\mathbf{v}] = \nabla C(\mathbf{w})$. That is, $\mathbf{v}$ is an unbiased estimator of the gradient of $C(\mathbf{w})$. Next, we update the dual vector according to $\boldsymbol{\theta} = \boldsymbol{\theta} - \eta\,\mathbf{v}$. If the link function is the identity mapping, this step is identical to the update of stochastic gradient descent. However, in our case f is not the identity function and it is important to distinguish between $\boldsymbol{\theta}$ and $\mathbf{w}$. The above update of $\boldsymbol{\theta}$ translates to an update of $\mathbf{w}$ by applying the link function $\mathbf{w} = f^{-1}(\boldsymbol{\theta})$. So far, we ignored the additional ℓ_1 regularization term. The simplest way to take this term into account is by also subtracting from $\boldsymbol{\theta}$ the gradient of the term $\lambda\|\mathbf{w}\|_1$. (More precisely, since the ℓ_1 norm is not differentiable, we will use any subgradient of $\|\mathbf{w}\|_1$ instead, e.g. the vector whose jth element is $\text{sign}(w_j)$, where we interpret $\text{sign}(0) = 0$.) Therefore, we could have redefined the update of $\boldsymbol{\theta}$ to be $\theta_j = \theta_j - \eta(v_j + \lambda\,\text{sign}(w_j))$. Regretfully, as noted in Langford et al., 2009, this update leads to a dense vector $\boldsymbol{\theta}$, which in turn leads to a dense vector $\mathbf{w}$. The solution proposed in Langford et al., 2009 breaks the update into three phases. First, we let $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} - \eta\,\mathbf{v}$. Second, we let $\hat{\boldsymbol{\theta}} = \tilde{\boldsymbol{\theta}} - \eta\,\lambda\,\text{sign}(\tilde{\boldsymbol{\theta}})$. Last, if in the second step we crossed the zero value, i.e. $\text{sign}(\hat{\theta}_j) \neq \text{sign}(\tilde{\theta}_j)$, then we truncate the jth element to be zero. Intuitively, the goal of the first step is to decrease the value of $C(\mathbf{w})$ and this is done by a (mirror) gradient step, while the goal of the second and third steps is to decrease the value of $\lambda\|\mathbf{w}\|_1$. So, by truncating $\boldsymbol{\theta}$ at zero we make the value of $\lambda\|\mathbf{w}\|_1$ even smaller.

Algorithm 3 Stochastic Mirror Descent Algorithm made Sparse (SMIDAS)

parameter: $\eta > 0$
let $p = 2\,\ln(d)$ and let f^{-1} be as in Eq. (12)
let $\boldsymbol{\theta} = \mathbf{0}$
for $t = 1, 2, \ldots$
  let $\mathbf{w} = f^{-1}(\boldsymbol{\theta})$
  sample i uniformly at random from $\{1,\ldots,m\}$
  let $\mathbf{v} = L'(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i)\,\mathbf{x}_i$
  (L' is the derivative of L. See e.g. Eq. (8))
  let $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} - \eta\,\mathbf{v}$
  let $\forall j, \theta_j = \text{sign}(\tilde{\theta}_j)\,\max\{0, |\tilde{\theta}_j| - \eta\lambda\}$
end

3.1. Efficient Implementation

A naive implementation of Algorithm 3 leads to an $O(d)$ runtime for each iteration. We now present two improvements for two specific scenarios. First, assume that the data is sparse, that is $\mathbf{x}_i$ contains only s non-zero elements on average (see Eq. (10)). In this case, we can implement each iteration of Algorithm 3 in time $O(s)$. The idea is to maintain two additional scalars, α_1 and α_2, such that $\alpha_1 = \sum_j |\theta_j|^p$ and $\alpha_2 = \alpha_1^{1-2/p}$. Since $w_j = f_j^{-1}(\boldsymbol{\theta})$, we can then rewrite $w_j = |\theta_j|^{p-1}/\alpha_2$. Therefore, instead of maintaining $\mathbf{w}$ we can maintain a vector $\mathbf{z}$, where $z_j = |\theta_j|^{p-1}$, and the scalar α_2. Clearly, we can calculate $\langle \mathbf{w}, \mathbf{x}_i\rangle$ based on $\mathbf{z}$ and α_2 by making a single pass over the non-zero elements of $\mathbf{x}_i$. Additionally, we can update $\boldsymbol{\theta}, \mathbf{z}$ and α_1 by making an additional single pass over the non-zeros elements of $\mathbf{x}_i$. Finally, we calculate α_2 from α_1 using $O(1)$ operations. Therefore, each iteration of Algorithm 3 requires two passes over the non-zero elements of $\mathbf{x}_i$, and the total runtime is $O(s)$.

Next, we describe a different scenario in which the main bottleneck is to calculate the values of the features. In this case, when calculating $\langle \mathbf{w}, \mathbf{x}_i\rangle$ we only need to calculate the features of $\mathbf{x}_i$ for which $w_j \neq 0$. Since the algorithm maintains sparse intermediate solutions, this can be much faster than calculating all the features of $\mathbf{x}_i$. Next, when updating $\boldsymbol{\theta}$, we note that if $\|\mathbf{x}_i\|_\infty \leq 1$ and $L'(\langle \mathbf{w}, \mathbf{x}_i\rangle, y_i) \leq \lambda$, then any element of θ_j which is currently zero will remain zero also after the update. So, again, in this case we do not need to calculate all the features of $\mathbf{x}_i$.

3.2. Runtime Guarantee

We now provide runtime guarantees for Algorithm 3. We introduce two types of assumptions on the loss function:

$$|L'(a,y)| \leq \rho \quad (13)$$
$$|L'(a,y)|^2 \leq \rho\, L(a,y) \quad (14)$$

In the above, L' is the derivative w.r.t. the first argument and can also be a sub-gradient of L if L is not differentiable. It is easy to verify that Eq. (14) holds for the squared-loss with $\rho = 4$ and that Eq. (13) holds for the hinge-loss, $L(a,y) = \max\{0, 1 - ya\}$, with $\rho = 1$. Interestingly, for the logistic-loss, both Eq. (13) holds with $\rho = 1$ and Eq. (14) holds with $\rho = 1/2$.

Theorem 2 *Let $\mathbf{w}^\star$ be a minimizer of Eq. (11) and let L satisfy Eq. (13) or Eq. (14). Then, if Algorithm 3 is run with $\eta = \rho\|\mathbf{w}^\star\|_1\sqrt{2(p-1)e/T}$ for T iterations and $\mathbf{w}_o = \mathbf{w}_r$ for r chosen uniformly at random from $[T]$, we have*

$$\mathbb{E}[P(\mathbf{w}_o)] - P(\mathbf{w}^\star) \leq O\left(\rho\,\|\mathbf{w}^\star\|_1\sqrt{\tfrac{\log(d)}{T}}\right) .$$

Proof Sketch Due to lack of space, we only sketch the proof for the case that Eq. (13) holds. Let $\boldsymbol{\theta}_t, \tilde{\boldsymbol{\theta}}_t$ be the values of $\boldsymbol{\theta}, \tilde{\boldsymbol{\theta}}$ at iteration t of the algorithm. Let $\mathbf{w}_t = f^{-1}(\boldsymbol{\theta}_t)$ and $\tilde{\mathbf{w}}_t = f^{-1}(\tilde{\boldsymbol{\theta}}_t)$. Define the Bregman divergence potential $\Psi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}^\star\|_q^2 - \frac{1}{2}\|\mathbf{w}\|_q^2 - \langle f(\mathbf{w}), \mathbf{w}^\star - \mathbf{w}\rangle$. We first rewrite $\Psi(\mathbf{w}_t) - \Psi(\mathbf{w}_{t+1}) = (\Psi(\mathbf{w}_t) - \Psi(\tilde{\mathbf{w}}_t)) + (\Psi(\tilde{\mathbf{w}}_t) - \Psi(\mathbf{w}_{t+1}))$. Standard analysis of mirror descent yields $\Psi(\mathbf{w}_t) - \Psi(\tilde{\mathbf{w}}_t) \geq \eta(L(\langle\mathbf{w}_t, \mathbf{x}_i\rangle, y_i) - L(\langle\mathbf{w}^\star, \mathbf{x}_i\rangle, y_i)) - \frac{\eta^2\,(p-1)\|\mathbf{v}_t\|_p^2}{2}$. See e.g., Beck & Teboulle, 2003. From Eq. (13) we obtain that $\|\mathbf{v}_t\|_p^2 \leq \rho^2\, d^{2/p} = \rho^2\, e$. Thus, $\Psi(\mathbf{w}_t) - \Psi(\tilde{\mathbf{w}}_t) \geq \eta(L(\langle\mathbf{w}_t, \mathbf{x}_i\rangle, y_i) - L(\langle\mathbf{w}^\star, \mathbf{x}_i\rangle, y_i)) - \frac{\eta^2\,(p-1)\,\rho^2\,e}{2}$. The more involved part of the proof is to show that $\Psi(\tilde{\mathbf{w}}_t) - \Psi(\mathbf{w}_{t+1}) \geq \eta\lambda(\|\mathbf{w}_{t+1}\|_1 - \|\mathbf{w}^\star\|_1)$. To show this, we use the definition of Ψ and the non-negativity of Bregman divergence to get that $\Psi(\tilde{\mathbf{w}}_t) - \Psi(\mathbf{w}_{t+1}) \geq \langle\mathbf{w}^\star - \mathbf{w}_{t+1}, \boldsymbol{\theta}_{t+1} - \tilde{\boldsymbol{\theta}}_t\rangle$. Using the definition of $\boldsymbol{\theta}_{t+1}$ and $\mathbf{w}_{t+1}$ we have $\langle\mathbf{w}_{t+1}, \boldsymbol{\theta}_{t+1} - \tilde{\boldsymbol{\theta}}_t\rangle = \eta\lambda\|\mathbf{w}_{t+1}\|_1$.[3] In addition, Holder inequality implies that $\langle\mathbf{w}^\star, \boldsymbol{\theta}_{t+1} - \tilde{\boldsymbol{\theta}}_t\rangle \leq \eta\,\lambda\,\|\mathbf{w}^\star\|_1$. Combining all the above together and taking expectation w.r.t. i we get $\mathbb{E}[P(\mathbf{w}_t)] - P(\mathbf{w}^\star) \leq \lambda\mathbb{E}\left[\|\mathbf{w}_t\|_1 - \|\mathbf{w}_{t+1}\|_1\right] + \frac{1}{\eta}\,\mathbb{E}\left[\Psi(\mathbf{w}_t) - \Psi(\mathbf{w}_{t+1})\right] + \frac{\eta(p-1)\rho^2\,e}{2}$. Summing over t, rearranging, and optimizing over η concludes our proof. ■

The bound in the above theorem can be improved if Eq. (14) holds and the desired accuracy is the same order as $P(\mathbf{w}^\star)$. We omit the details due to lack of space.

[3] Note that this equality does not hold for the Bregman potential corresponding to exponentiated gradient.

4. Experiments

We consider 4 datasets for our experiments: REUTERS, ARCENE, MAGIC04S, and MAGIC04D. REUTERS is a dataset obtained from the Reuters RCV1 collection. Examples in this collection can have more than 1 label. We created a binary classification dataset out of this by treating any example that was labelled CCAT as having label $+1$. Rest were assigned the label -1. This gave us a 378,452 dimensional dataset with 806,791 examples. There were 9.8×10^7 non-zero entries in the example matrix corresponding to a sparsity level of 0.03%. ARCENE is a dataset from the UCI Machine Learning repository where the task is to distinguish cancer patterns from normal ones based on 10,000 mass-spectrometric features. Out of these, 3,000 features are synthetic features as this dataset was designed for the NIPS 2003 variable selection workshop. There are 100 examples in this dataset and the example matrix contains 5.4×10^5 non-zero entries corresponding to a sparsity level of 54%. The datasets MAGIC04S and MAGIC04D were obtained by adding 1,000 random features to the MAGIC Gamma Telescope dataset from the UCI Machine Learning repository. The original dataset has 19,020 examples with 10 features. This is also a binary classification dataset and the task is to distinguish high-energy gamma particles from background using a gamma telescope. Following the experimental setup of Langford et al., 2009, we added 1,000 random features, each of which takes value 0 with probability 0.95 or 1 with probability 0.05, to create a sparse dataset, MAGIC04S. We also created a dense dataset, MAGIC04D, in which the random features took value -1 or $+1$, each with probability 0.5. MAGIC04S and MAGIC04D has sparsity levels of 5.81% and 100% respectively.

We ran 4 algorithms on these datasets: SCD, DETCD, SMIDAS, and TRUNCGRAD. SCD is the stochastic coordinate descent algorithm given in Section 2 above. DETCD is the corresponding deterministic version of the same algorithm. The coordinate to be updated at each iteration is chosen in a deterministic manner to maximize a lower bound on the guaranteed decrease in the objective function. This type of deterministic criterion for choosing features is common in Boosting approaches. Since choosing a coordinate (or feature in our case) in a deterministic manner involves significant computation in case of large datasets, we expect that the deterministic algorithm will converge much slower than the stochastic algorithm. We also tried using a deterministic coordinate descent algorithm that works with the domain constraint formulation as discussed in Section 1.1. It was also slower than SCD although we do not report its results here as it does not work directly with the regularized loss. SMIDAS is the mirror descent algorithm given in Section 3 above. TRUNCGRAD is the truncated gradient algorithm of Langford et al., 2009 (In fact, Langford et al., 2009 suggests another way to trun-

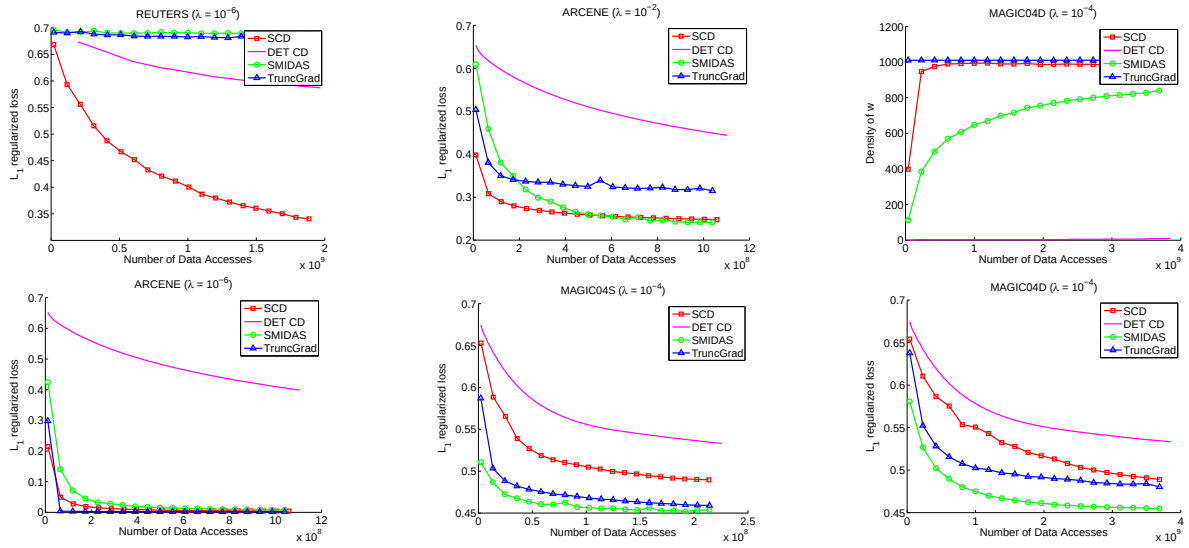

Figure 1. Performance of SCD, DETCD, SMIDAS, and TRUNCGRAD on 4 datasets (see Section 4 for details)

cate the gradient. Here, we refer to the variant corresponding to SMIDAS.) Of these 4, the first two are parameter-free algorithms while the latter two require a parameter η. In our experiments, we ran SMIDAS and TRUNCGRAD for a range of different values of η and chose the one that yielded the minimum value of the objective function (i.e. the regularized loss).

The top-left plot in Figure 1 is for the REUTERS dataset. It shows the regularized loss objective achieved by the 4 algorithms as a function of the number of times they access the data matrix $(x_{i,j})$ of the REUTERS dataset. We choose to use this as opposed to, say CPU time, as this is an implementation independent quantity. Moreover, the actual time taken by these algorithms will be roughly proportional to this quantity. The regularization parameter λ was set to 10^{-6} (but similar results hold for other values of λ as well). The η parameter of SMIDAS and TRUNCGRAD was searched over the range 10^{-6} to 10^{-1} in exponentially increasing step sizes. It is clear that SCD outperforms the other three algorithms. DETCD is much slower compared to SCD because, as we mentioned above, it spends a lot of time in finding the best coordinate to update. The two algorithms having a tunable parameter η do much worse here. The situation is even worse if we add up the time to perform several runs of these algorithms for tuning η. This illustrates a problem practitioners have to deal with when faced with large datasets. A parameter-free algorithm that is also quicker to decrease the objective function is clearly preferred in such situations.

The bottom-left and top-middle plots in Figure 1 are for the ARCENE dataset. We have plotted the regularized loss objective against the number of data matrix accesses for a small and a large value of λ (10^{-6} and 10^{-2} respectively). SMIDAS does much better than TRUNCGRAD for the large value and is worse, but still competitive, for the small value. Note that SCD does well and DETCD is quite slow in both scenarios.

For the MAGIC datasets, SMIDAS does much better than TRUNCGRAD for the MAGIC04D dataset (where the example vectors are dense). TRUNCGRAD is competitive for the MAGIC04S dataset (where the example vectors are sparse). This is illustrated in the bottom-middle and bottom-right plots in Figure 1. Note that this behavior is consistent with the bounds Eq. (3) and Eq. (4) given above. These bounds suggest that if the true solution has low ℓ_1 norm, SMIDAS will require fewer iterations than TRUNCGRAD when the examples are dense.

For MAGIC04D, we also plotted the density (or the ℓ_0 norm) of the weight vector $\mathbf{w}$ as a function of number of accesses to the data matrix for all 4 algorithms. This is shown in the top-right plot in Figure 1. It is interesting to note that SCD not only minimizes the objective quickly but it also maintains sparsity along the way. If we compare SCD with TRUNCGRAD in the two plots on the right, we find that TRUNCGRAD is better at minimizing the objective while SCD finds slightly sparser $\mathbf{w}$'s. All plots for the MAGIC datasets are for $\lambda = 10^{-3}$.

5. Discussion

Focusing on the problem of ℓ_1 regularized loss minimization, we showed how stochastic approaches can yield simple and practical methods for large data sets. Furthermore, the stochastic methods described in this paper outperform their corresponding deterministic approaches. To the best of our knowledge, the only known provably correct deter-

ministic coordinate method for ℓ_1 regularization is that of Tseng & Yun, 2009, which both has worse runtime and is much more complicated to implement.[4]

There are several possible extension to this work. For example, the p-norm algorithm has been originally suggested as an interpolation between additive and multiplicative updates. Naturally, one can use the SMIDAS algorithm with different values of p, yielding an interpolation between Langford et al., 2009 and SMIDAS. Another possible extension of this work is the usage of our analysis for better understanding stochastic coordinate descent methods for optimizing the dual of Support Vector Machines.

References

Beck, A., & Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, *31*, 167–175.

Bottou, L. (Web Page). Stochastic gradient descent examples. `http://leon.bottou.org/projects/sgd`.

Bottou, L., & Bousquet, O. (2008). The tradeoffs of large scale learning. *Advances in Neural Information Processing Systems 20* (pp. 161–168).

Bottou, L., & LeCunn, Y. (2005). On-line learning for very large datasets. *Appl. Stoch. Model. Bus. and Ind.*, *21*, 137–151.

Clarkson, K. (2008). Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 922–931).

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines*. Cambridge University Press.

Duchi, J., Shalev-Shwartz, S., Singer, Y., & Chandra, T. (2008). Efficient projections onto the ℓ_1-ball for learning in high dimensions. *Proceedings of the 25th International Conference on Machine Learning* (pp. 272–279).

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Ann. Statist.*, *32*, 407–499.

Frank, M., & Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Res. Logist. Quart.*, *3*, 95–110.

Friedman, J., Hastie, T., & Tibshirani, R. (2008). *Regularized paths for generalized linear models via coordinate descent* (Technical Report). Department of Statistics, Stanford University.

[4]The deterministic coordinate descent method for loss minimization with ℓ_1 constraint is maybe the best deterministic competitor, but as we showed, it is both theoretically and empirically inferior to our stochastic gradient descent algorithm.

Genkin, A., Lewis, D., & Madigan, D. (2007). Large-scale Bayesian logistic regression for text categorization. *Technometrics*, *49*, 291–304.

Gentile, C. (2003). The robustness of the p-norm algorithms. *Machine Learning*, *53*, 265–299.

Grove, A. J., Littlestone, N., & Schuurmans, D. (2001). General convergence results for linear discriminant updates. *Machine Learning*, *43*, 173–210.

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

Koh, K., Kim, S., & Boyd, S. (2007). An interior-point method for large-scale ℓ_1-regularized logistic regression. *Journal of Machine Learning Research*, *8*, 1519–1555.

Langford, J., Li, L., & Zhang, T. (2009). Sparse online learning via truncated gradient. *Advances in Neural Information Processing Systems 21* (pp. 905–912).

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, *2*, 285–318.

Luo, Z., & Tseng, P. (1992). On the convergence of coordinate descent method for convex differentiable minimization. *J. Optim. Theory Appl.*, *72*, 7–35.

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

Shalev-Shwartz, S., & Srebro, N. (2008). SVM optimization: Inverse dependence on training set size. *Proceedings of the 25th International Conference on Machine Learning* (pp. 928–935).

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B.*, *58*, 267–288.

Tseng, P., & Yun, S. (2009). A block-coordinate gradient descent method for linearly constrained nonsmooth separable optimization. *J. Optim. Theory Appl.*, *140*, 513–535.

Wu, T. T., & Lange, K. (2008). Coordinate descent algorithms for lasso penalized regression. *Annals of Applied Statistics*, *2*, 224–244.

Zhang, T. (2003). Sequential greedy approximation for certain convex optimization problems. *IEEE Transaction on Information Theory*, *49*, 682–691.

Zhang, T., & Oles, F. J. (2001). Text categorization based on regularized linear classification methods. *Information Retrieval*, *4*, 5–31.