Fast Stochastic Alternating Direction Method of Multipliers

Leon Wenliang Zhong WZHONG@CSE.UST.HK
James T. Kwok JAMESK@CSE.UST.HK
Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong

Abstract

We propose a new stochastic alternating direction method of multipliers (ADMM) algorithm, which incrementally approximates the full gradient in the linearized ADMM formulation. Besides having a low per-iteration complexity as existing stochastic ADMM algorithms, it improves the convergence rate on convex problems from $\mathcal{O}(1/\sqrt{T})$ to $\mathcal{O}(1/T)$, where T is the number of iterations. This matches the convergence rate of the batch ADMM algorithm, but without the need to visit all the samples in each iteration. Experiments on the graph-guided fused lasso demonstrate that the new algorithm is significantly faster than state-of-the-art stochastic and batch ADMM algorithms.

1. Introduction

The alternating direction method of multipliers (ADMM) (Boyd, 2010; Gabay & Mercier, 1976; Glowinski & Marrocco, 1975) considers problems of the form

$$\min_{x,y} \quad \Phi(x,y) \equiv \phi(x) + \psi(y) \ : \ Ax + By = c, \quad (1)$$

where ϕ, ψ are convex functions, and A, B (resp. c) are constant matrices (resp. vector) of appropriate sizes. Because of the flexibility in splitting the objective into $\phi(x)$ and $\psi(y)$, it has been a popular optimization tool in many machine learning, computer vision and data mining applications. For example, on large-scale distributed convex optimization, each ϕ or ψ can correspond to an optimization subproblem on the local data, and the constraint $Ax+By = c$ is used to ensure all the local variables reach a global consensus; for regularized risk minimization which will be the focus of this paper, ϕ can be used for the empirical loss, ψ for the regularizer, and the constraint for encoding the sparsity pattern of the model parameter. In comparison with other state-of-the-art optimization methods such as proximal gradient methods (Duchi & Singer, 2009; Xiao, 2010), the use of ADMM has been shown to have faster convergence in several difficult structured sparse regularization problems (Suzuki, 2013).

Existing works on ADMM often assume that $\Phi(x, y)$ is deterministic. In the context of regularized risk minimization, this corresponds to batch learning and each iteration needs to visit all the samples. With the proliferation of data-intensive applications, it can quickly become computationally expensive. For example, in using ADMM on the overlapping group lasso, the matrix computations become costly when both the number of features and data set size are large (Qin & Goldfarb, 2012). To alleviate this problem, the use of stochastic and online techniques have recently drawn a lot of interest. Wang & Banerjee (2012) first proposed the online ADMM, which learns from only one sample (or a small mini-batch) at a time. However, in general, each round involves nonlinear optimization and is thus not computationally appealing. Very recently, three stochastic variants of ADMM are independently proposed (Ouyang et al., 2013; Suzuki, 2013). Two are based on the stochastic gradient descent (SGD) (Bottou, 2004), while one is based on regularized dual averaging (RDA) (Xiao, 2010). In both cases, the difficult nonlinear optimization problem inherent in the online ADMM is circumvented by linearization, which then allows the resultant iterations in these stochastic variants to be efficiently performed.

However, despite their low per-iteration complexities, these stochastic ADMM algorithms converge at a suboptimal rate compared to their batch counterpart. Specifically, the algorithms in (Ouyang et al., 2013; Suzuki, 2013) all achieve a rate of $\mathcal{O}(1/\sqrt{T})$, where T is the number of iterations, for general convex problems and $\mathcal{O}(\log T/T)$ for strongly convex problems; whereas batch ADMM achieves $\mathcal{O}(1/T)$ and linear convergence, respectively (Deng & Yin, 2012; He & Yuan, 2012). This gap in the convergence rates between stochastic and batch ADMM algorithms is indeed not surprising, as it is also observed between SGD and batch gradient descent in the analogous unconstrained optimization setting (Mairal, 2013). Recently, there have been several attempts on bridging this gap (Le Roux et al., 2012;

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

Mairal, 2013; Shalev-Shwartz & Zhang, 2013a). For example, Le Roux et al. (2012) proposed an approach whose per-iteration cost is as low as SGD, but can achieve linear convergence for strongly convex functions.

Along this line, we propose in the sequel a novel stochastic algorithm that bridges the $\mathcal{O}(1/\sqrt{T})$ vs $\mathcal{O}(1/T)$ gap in the convergence rates for ADMM. The new algorithm enjoys the same computational simplicity as existing stochastic ADMM algorithms, but with a much faster convergence rate matching that of its batch counterpart. Experimental results demonstrate that it dramatically outperforms existing stochastic and batch ADMM algorithms.

Notation. Let $\|\cdot\|$ be the Euclidean norm. For a function f, we let f' be a subgradient in the subdifferential set $\partial f(x) = \{g \mid f(y) \geq f(x) + g^T(y-x),\ \forall y\}$. When f is differentiable, we use ∇f for its gradient. A function f is L-Lipschitz if $\|f(x) - f(y)\| \leq L\|x-y\|\ \forall x, y$. It is L-smooth if $\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|$, or equivalently,

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \frac{L}{2}\|x-y\|^2. \quad (2)$$

Moreover, a function f is μ-strongly convex if $f(y) \geq f(x) + g^T(y-x) + \frac{\mu}{2}\|y-x\|^2$ for $g \in \partial f(x)$.

2. Related Work

2.1. Batch and Stochastic ADMM

As in the method of multipliers, ADMM starts with the augmented Lagrangian of problem (1):

$$\begin{aligned} L(x,y,\beta) \quad = \quad & \phi(x) + \psi(y) + \beta^T(Ax+By-c) \\ & + \frac{\rho}{2}\|Ax+By-c\|^2, \end{aligned} \quad (3)$$

where β is the vector of Lagrangian multipliers, and $\rho > 0$ is a penalty parameter. At the tth iteration, the values of x and y (denoted x_t, y_t) are updated by minimizing $L(x, y, \beta)$ w.r.t. x and y. Unlike the method of multipliers, these are minimized in an alternating manner, which allows the problem to be more easily decomposed when ϕ and ψ are separable. Using the scaled dual variable $\alpha_t = \beta_t/\rho$, the ADMM update can be expressed as (Boyd, 2010):

$$x_{t+1} \leftarrow \arg\min_x\ \phi(x) + \frac{\rho}{2}\|Ax + By_t - c + \alpha_t\|^2, \quad (4)$$

$$y_{t+1} \leftarrow \arg\min_y\ \psi(y) + \frac{\rho}{2}\|Ax_{t+1} + By - c + \alpha_t\|^2, \quad (5)$$

$$\alpha_{t+1} \leftarrow \alpha_t + Ax_{t+1} + By_{t+1} - c. \quad (6)$$

In the context of regularized risk minimization, x denotes the model parameter to be learned. Moreover,

$$\phi(x) = \frac{1}{n}\sum_{i=1}^n \ell_i(x) + \Omega(x), \quad (7)$$

where n is the number of samples, $\ell_i(x)$ is sample i's contribution to the (possibly nonsmooth) empirical loss, and $\Omega(x)$ is the regularizer. Problem (4) then becomes

$$\begin{aligned} x_{t+1} \quad \leftarrow \quad & \arg\min_x\ \frac{1}{n}\sum_{i=1}^n \ell_i(x) + \Omega(x) \\ & + \frac{\rho}{2}\|Ax + By_t - c + \alpha_t\|^2. \end{aligned} \quad (8)$$

When the data set is large, solving (8) can be computationally expensive. To alleviate this problem, Wang & Banerjee (2012) proposed the *online ADMM* that uses only one sample in each iteration. Consider $\Omega = 0$. Let the index of the sample selected at iteration t be $k(t) \in \{1, 2, \ldots, n\}$. Instead of using (8), x is updated as $x_{t+1} \leftarrow \arg\min_x\ \ell_{k(t)}(x) + \frac{\rho}{2}\|Ax + By_t - c + \alpha_t\|^2 + \frac{D(x,x_t)}{\eta_{t+1}}$, where $D(x, x_t)$ is a Bregman divergence between x and x_t, and $\eta_t \propto \frac{1}{\sqrt{t}}$ is the stepsize. In general, this involves nonlinear optimization, making it potentially expensive.

Very recently, several stochastic versions of ADMM have been independently proposed. Ouyang et al. (2013) proposed the *stochastic ADMM*[1], which updates x as

$$\begin{aligned} x_{t+1} \leftarrow & \arg\min_x\ \ell'_{k(t)}(x_t)^T(x-x_t) + \frac{\|x-x_t\|^2}{2\eta_{t+1}} \\ & + \frac{\rho}{2}\|Ax + By_t - c + \alpha_t\|^2 \end{aligned} \quad (9)$$

$$\begin{aligned} = & \left(\frac{1}{\eta_{t+1}}I + \rho A^TA\right)^{-1} \cdot \\ & \left[\frac{x_t}{\eta_{t+1}} - \ell'_{k(t)}(x_t) - \rho A^T(By_t - c + \alpha_t)\right] \end{aligned} \quad (10)$$

where $\ell'_{k(t)}(x_t)$ is the (sub)gradient of $\ell_{k(t)}$ at x_t, $\eta_t \propto \frac{1}{\sqrt{t}}$ is the stepsize, and I is the identity matrix. The updates for y and α are the same as in (5) and (6).

For the special case where $B = -I$ and $c = 0$, Suzuki (2013) proposed a similar approach called *online proximal gradient descent ADMM* (OPG-ADMM), which uses the inexact Uzawa method (Zhang et al., 2011) to further linearize the last term in (9). Specifically, let L_A be an upper bound on the eigenvalues of ρA^TA it upper-bounds $\frac{\rho}{2}\|Ax - y_t + \alpha_t\|^2$ with $\frac{\rho}{2}\|Ax_t - y_t + \alpha_t\|^2 + \rho A^T(Ax - y_t + \alpha_t)^T(x - x_t) + \frac{L_A}{2}\|x - x_t\|^2$, leading to

$$\begin{aligned} x_{t+1} \leftarrow & \arg\min_x\ (\ell'_{k(t)}(x_t) + \rho A^T(Ax_t - y_t + \alpha_t))^T x \\ & + \frac{\|x-x_t\|^2}{2\eta_{t+1}} \end{aligned} \quad (11)$$

$$= x_t - \eta_{t+1}\left[\ell'_{k(t)}(x_t) + \rho A^T(Ax_t - y_t + \alpha_t)\right]. \quad (12)$$

[1] To avoid confusion, this particular stochastic variant will be called STOC-ADMM in the sequel.

Compared to (10), it avoids the inversion of $\frac{1}{\eta_{t+1}}I+\rho A^TA$ which can be computationally expensive when A^TA is large. Suzuki (2013) also proposed another stochastic variant called *RDA-ADMM* based on the method of regularized dual averaging (RDA) (Xiao, 2010) (again for the special case with $B=-I$ and $c=0$), in which x is updated as

$$\begin{aligned} x_{t+1} \leftarrow & \arg\min_x (\bar{g}_t + \rho A^T(A\bar{x}_t^{\text{RDA}} - \bar{y}_t^{\text{RDA}} + \bar{\alpha}_t^{\text{RDA}})^T x \\ & + \frac{\|x\|^2}{\eta_{t+1}} \\ = & -\eta_{t+1}\left[\bar{g}_t + \rho A^T(A\bar{x}_t^{\text{RDA}} - \bar{y}_t^{\text{RDA}} + \bar{\alpha}_t^{\text{RDA}})\right]. \end{aligned} \quad (13)$$

Here, $\eta_t \propto \sqrt{t}$, $\bar{g}_t = \frac{1}{t}\sum_{j=1}^t \ell'_{k(j)}(x_j)$, $\bar{x}_t^{\text{RDA}} = \frac{1}{t}\sum_{j=1}^t x_j$, $\bar{y}_t^{\text{RDA}} = \frac{1}{t}\sum_{j=1}^t y_j$, and $\bar{\alpha}_t^{\text{RDA}} = \frac{1}{t}\sum_{j=1}^t \alpha_j$ are averages obtained from the past t iterations.

For general convex problems, these online/stochastic ADMM approaches all converge at a rate of $\mathcal{O}(\frac{\sigma}{\sqrt{T}})$ w.r.t. either the objective value (Suzuki, 2013) or a weighted combination of the objective value and feasibility violation (Ouyang et al., 2013), where σ is the standard deviation of the stochastic gradient. When ϕ is further required to be strongly convex and that the gradient's norm is bounded by M, the convergence rate can be improved to $\mathcal{O}(\frac{M^2\log T}{\mu T})$ (except for RDA-ADMM whose convergence rate in this situation is unclear). However, in both cases (general and strongly convex), these are inferior to the $\mathcal{O}(\frac{1}{T})$ and linear convergence rates of their batch ADMM counterparts (Deng & Yin, 2012; He & Yuan, 2012; Hong & Luo, 2012).

2.2. Stochastic Optimization with Finite Samples

While the full gradient descent has linear convergence, it is well-known that SGD only achieves sublinear convergence (Mairal, 2013) (i.e., $\mathcal{O}(\frac{L}{T})$). Recently, by observing that the training set is indeed finite, it is shown that the convergence rates of stochastic algorithms can be improved to match those of the batch learning algorithms. A pioneering approach along this line is the *stochastic average gradient* (SAG) (Le Roux et al., 2012), which considers the optimization of a strongly convex sum of smooth functions ($\min_x \frac{1}{n}\sum_{i=1}^n \ell_i(x)$). By updating an estimate of the full gradient incrementally in each iteration, the per-iteration time complexity of SAG is only as low as SGD, yet surprisingly its convergence rate is linear. Specifically, at iteration t, a sample with index $k(t) \in \{1,2,\ldots,n\}$ is randomly chosen and its contribution to the objective's gradient recomputed. Variable x is then updated as

$$x_{t+1} \leftarrow x_t - \frac{\eta}{n}\sum_{i=1}^n \nabla\ell_i(x_{\tau_i(t)}), \quad (14)$$

where

$$\tau_i(t) = \begin{cases} t & i = k(t) \\ \tau_i(t-1) & \text{otherwise} \end{cases}, \quad (15)$$

and η is a constant stepsize. The recently proposed *stochastic coordinate descent* (Shalev-Shwartz & Zhang, 2013b) also obtains a similar convergence rate.

Another closely related approach is the *minimization by incremental surrogate optimization* (MISO) (Mairal, 2013), which replaces each ℓ_i by some "surrogate" function in an incremental manner similar to SAG. To optimize $\frac{1}{n}\sum_{i=1}^n \ell_i(x) + \Omega(x)$, where $\ell_i(x)$ is smooth and convex, and $\Omega(x)$ is nonsmooth, MISO first approximates the smooth part as $P_t^\ell(x) \equiv \frac{1}{n}\sum_{i=1}^n \ell_i(x_{\tau_i(t)}) + \nabla\ell_i(x_{\tau_i(t)})^T(x - x_{\tau_i(t)}) + \frac{L}{2}\|x - x_{\tau_i(t)}\|^2$, and then updates x as $x_{t+1} \leftarrow \arg\min_x P_t^\ell(x) + \Omega(x)$. This can be viewed as an extension of SAG, and enjoys the same convergence rate as proximal methods with full gradient, i.e., $\mathcal{O}\left(\frac{n}{T}\right)$ for general convex problems and linear convergence for strongly convex problems. Proximal stochastic coordinate descent methods are also studied recently in (Shalev-Shwartz & Zhang, 2013a).

3. Stochastic Average ADMM (SA-ADMM)

On comparing the update rules on x for the STOC-ADMM and OPG-ADMM ((9) and (11)) with that of the batch ADMM (8), one can see that the empirical loss on the whole training set (namely, $\frac{1}{n}\sum_{i=1}^n \ell_i(x)$) is replaced by the linear approximation based on one single sample plus a proximal term $\frac{\|x-x_t\|^2}{2\eta_{t+1}}$. This follows the standard approach taken by SGD. As discussed in Section 2.2, SGD has slower convergence than full gradient descent. This also agrees with the result in Section 2.1 that the existing stochastic versions of ADMM all have slower convergence rates than their batch counterpart.

Motivated by the recent stochastic optimization results in Section 2.2, we will propose a novel stochastic ADMM algorithm that achieves the same convergence rate as batch ADMM on general convex problems. Unlike SAG or MISO, the proposed algorithm is more general and can be applied to optimization problems with equality constraints, which are naturally handled by ADMM.

3.1. Algorithm

In the following, we assume that ℓ_i in (7) is L-smooth (e.g., square loss and logistic loss). As in existing stochastic ADMM approaches (Wang & Banerjee, 2012; Ouyang et al., 2013; Suzuki, 2013), y and α are still updated by (5), (6). The key difference is on the update of x (Algorithm 1).

First, consider the special case where $\Omega = 0$. At iteration t, we randomly choose a sample $k(t)$ uniformly from $\{1,2,\ldots,n\}$, and then update x as

$$x_{t+1} \leftarrow \arg\min_x P_t^\ell(x) + r(x, y_t, \alpha_t), \quad (16)$$

Algorithm 1 Stochastic average alternating direction method of multipliers (SA-ADMM).

Initialize: x_0, y_0, α_0 and $\tau_i(-1) = 0 \forall i$.
for $t = 0, 1, \ldots, T-1$ **do**
  randomly choose $k \in \{1, 2, \ldots, n\}$, and set $\tau_i(t)$ as in (15);
  update x_{t+1} using (17) or (20) when $\Omega = 0$; and use (23) when $\Omega \neq 0$;
  $y_{t+1} \leftarrow \arg\min_y \psi(y) + \frac{\rho}{2}\|Ax_{t+1} + By - c + \alpha_t\|^2$;
  $\alpha_{t+1} \leftarrow \alpha_t + (Ax_{t+1} + By_{t+1} - c)$;
end for
Output: $\bar{x}_T \leftarrow \frac{1}{T}\sum_{t=1}^T x_t, \bar{y}_T \leftarrow \frac{1}{T}\sum_{t=1}^T y_t$.

where

$$\begin{aligned} P_t^\ell(x) &\equiv \frac{1}{n}\sum_{i=1}^n \ell_i(x_{\tau_i(t)}) + \nabla \ell_i(x_{\tau_i(t)})^T(x - x_{\tau_i(t)}) \\ &\quad + \frac{L}{2}\|x - x_{\tau_i(t)}\|^2, \\ r(x, y, \alpha) &\equiv \frac{\rho}{2}\|Ax + By - c - \alpha\|^2, \end{aligned}$$

and $\tau_i(t)$ is as defined in (15). Hence, as in SAG, out of the n gradient terms in (16), only one of them (which corresponds to sample $k(t)$) is based on the current iterate x_t, while all others are previously-stored gradient values. Moreover, note its similarity with the STOC-ADMM update in (9), which only retains terms related to $\ell_{k(t)}$, while (16) uses the information from all of $\{\ell_1, \ell_2, \ldots, \ell_n\}$. Another difference with (9) is that the proximal term in (16) involves a constant L, while (10) requires a time-varying stepsize η_t.

By setting the derivative of (16) to zero, we have

$$\begin{aligned} x_{t+1} \leftarrow\ & (\rho A^T A + LI)^{-1} \\ & \cdot \left[L\bar{x}_t - \rho A^T(By_t - c + \alpha_t) - \overline{\nabla \ell}_t\right], \end{aligned} \quad (17)$$

where $\bar{x}_t = \frac{1}{n}\sum_{i=1}^n x_{\tau_i(t)}$, and $\overline{\nabla \ell}_t = \frac{1}{n}\sum_{i=1}^n \nabla \ell_i(x_{\tau_i(t)})$. When the dimension of $A^T A$ is manageable, $(\rho A^T A + LI)^{-1}$ can be pre-computed and stored. On the other hand, when $\rho A^T A + LI$ is large, even storing its inverse can be expensive. A technique that has been popularly used in the recent ADMM literature is the inexact Uzawa method (Zhang et al., 2011), which uses (2) to approximate $r(x, y_t, \alpha_t)$ by its upper bound:

$$\begin{aligned} r(x, y_t, \alpha_t) &\leq P_t^r(x) \\ &\equiv r(x_t, y_t, \alpha_t) + \nabla_x^t r^T(x - x_t) \\ &\quad + \frac{L_A}{2}\|x - x_t\|^2, \end{aligned} \quad (18)$$

where $\nabla_x^t r \equiv \rho A^T(Ax_t + By_t - c + \alpha_t)$ and L_A is an upper bound on the eigenvalues of $\rho A^T A$ (He & Yuan, 2012). Hence, (16) becomes

$$\begin{aligned} x_{t+1} &\leftarrow \arg\min_x P_t^\ell(x) + P_t^r(x) & (19) \\ &= \frac{L\bar{x}_t + L_A x_t - \left[\overline{\nabla \ell}_t + \nabla_x^t r)\right]}{L_A + L}. & (20) \end{aligned}$$

Analogous to the discussion on the relationship between (16) and (9) above, our (20) is also similar to the OPG-ADMM update in (12), except that all the information from $\{\ell_1, \ell_2, \ldots, \ell_n\}$ are now used. Moreover, note that although RDA-ADMM also uses an average of gradients ($\bar{g}_t$ in (13)), its convergence is still slower than the proposed algorithm (as will be seen in Section 3.4).

When $\Omega \neq 0$, it can be added back to (16), leading to

$$x_{t+1} \leftarrow \arg\min_x P_t^\ell(x) + \Omega(x) + r(x, y_t, \alpha_t). \quad (21)$$

In general, it is easier to solve with the inexact Uzawa simplification. The update then becomes

$$\begin{aligned} x_{t+1} &\leftarrow \arg\min_x P_t^\ell(x) + P_t^r(x) + \Omega(x) & (22) \\ &= \arg\min_x \frac{1}{2}\left\| x - \frac{L\bar{x}_t + L_A x_t - \left[\overline{\nabla \ell}_t + \nabla_x^t r\right]}{L_A + L}\right\|^2 \\ &\quad + \frac{\Omega(x)}{L_A + L}. & (23) \end{aligned}$$

This is the standard proximal step popularly used in optimization problems with structured sparsity (Bach et al., 2011). As is well-known, it can be efficiently computed as Ω is assumed "simple" (e.g., $\Omega(x) = \|x\|_1$, $\|x\|_2$, $\|x\|_\infty$ and various mixed norms (Duchi & Singer, 2009)).

3.2. Discussion

In the special case where $\psi = 0$, (1) reduces to $\min_x \phi(x)$ and the feasibility violation $r(x, y_t, \alpha_t)$ can be dropped. The update rule in (16) then reduces to MISO using the Lipschitz gradient surrogate; and (21) corresponds to the proximal gradient surrogate (Mairal, 2013). SAG, on the other hand, does not have the proximal term $\|x - x_{\tau_i(t)}\|^2$ in its update rule, and also cannot handle a nonsmooth Ω. When $\psi \neq 0$, (18) can be regarded as a quadratic surrogate (Mairal, 2013). Then, (19) (resp. (21)) is a combination of the Lipschitz (resp. proximal) gradient surrogate and quadratic surrogate, which can be easily seen to be another surrogate function in the sense of (Mairal, 2013). Note that when $\psi \neq 0$, MISO has to compute the proximity operator w.r.t. the equality constraints, which is difficult in general.

The stochastic algorithms in Section 2.1 only require ϕ to be convex, and do not explicitly consider its form in (7). Hence, there are two possibilities in the handling of a nonzero Ω. The first approach directly takes the nonsmooth ϕ in (7), and uses its subgradient in the update equations ((10), (12) and (13)). However, unlike the proximal

step in (23), this does not exploit the structure of ϕ and subgradient descent often has slow empirical convergence (Duchi & Singer, 2009). The second approach folds Ω into ψ by rewriting the optimization problem as

$$\min_{x,\begin{bmatrix}y\\z\end{bmatrix}} \quad \frac{1}{n}\sum_{i=1}^{n}\ell_i(x)+[\psi(y)+\Omega(z)]$$
$$\text{s.t.} \quad \begin{bmatrix}A\\I\end{bmatrix}x+\begin{bmatrix}B & 0\\0 & -I\end{bmatrix}\begin{bmatrix}y\\z\end{bmatrix}=\begin{bmatrix}c\\0\end{bmatrix}.$$

In the update step for $\begin{bmatrix}y\\z\end{bmatrix}$, it is easy to see from (5) that y and z are decoupled and thus can be optimized separately. In comparison with (23), a disadvantage of this reformulation is that an additional variable z (which is of the same size as x) has to be introduced. Hence, it is more computationally expensive empirically. Moreover, the radius of the parameter space is also increased, leading to bigger constants in the big-O notation of the convergence rate (Ouyang et al., 2013; Suzuki, 2013).

3.3. Per-Iteration Time Complexity

The proposed algorithm uses the same update rules for y_t and α_t as the existing stochastic ADMM algorithms (in Section 2.1). Hence, we only consider the complexities of the x_t updates here. For easy comparison, we take $B=-I$ and $c=0$, as assumed in (Suzuki, 2013). OPG-ADMM is the most efficient. Its update rule (12) takes $\mathcal{O}(nz(A))$ time, where $nz(A)$ is the number of nonzero elements in A. Both RDA-ADMM and the proposed update rule (20) require taking the average of the gradients/variables. The resultant complexities are $\mathcal{O}(nz(A)+d_x)$ and $\mathcal{O}(nz(A)+d_y)$, respectively, where d_x (resp. d_y) is the dimensionality of x (resp. y). Typically, $nz(A)$ is larger than d_x and d_y, and so OPG-ADMM, RDA-ADMM and rule (20) have the same complexity. STOC-ADMM (with update rule (10)) is the most expensive as matrix inversion is involved, which takes $\mathcal{O}(d_x^3)$ time. As to the proposed rule (17), one can pre-compute the matrix inverse as the step size is a constant.

3.4. Convergence Analysis

In this section, we show that the proposed ADMM algorithm has fast convergence. Recall that in the standard ADMM, equation (4) is used for updating x. In the proposed algorithm, the loss and feasibility violation are linearized, making the convergence analysis more difficult than those in (He & Yuan, 2012; Wang & Banerjee, 2012). Moreover, though related to MISO, our analysis is a non-trivial extension because of equality constraints and additional Lagrangian multipliers in the ADMM formulation.

Let $\|x\|_H \equiv x^THx$ for a psd matrix H, $H_x \equiv L_AI-\rho A^TA$, and $H_y \equiv \rho B^TB$. Denote the optimal solution of (1) by (x^*,y^*). As in (Ouyang et al., 2013), we first consider the convergence of $(\bar{x}_T,\bar{y}_T)$ in terms of a combination of the objective value and feasibility violation (weighted by some $\gamma>0$). The following theorem establishes $\mathcal{O}\left(\frac{1}{T}\right)$ convergence.

Theorem 1 *Using update rule (23),*

$$\mathbb{E}\left[\Phi(\bar{x}_T,\bar{y}_T)-\Phi(x^*,y^*)+\gamma\|A\bar{x}_T+B\bar{y}_T-c\|\right] \le \frac{1}{2T}\left\{\|x^*-x_0\|_{H_x}^2+nL\|x^*-x_0\|^2+\|y^*-y_0\|_{H_y}^2 +2\rho\left(\frac{\gamma^2}{\rho^2}+\|\alpha_0\|^2\right)\right\}.$$

When $\Omega=0$ and the inexact Uzawa simplification is not used, update rule (17) leads to

$$\mathbb{E}\left[\Phi(\bar{x}_T,\bar{y}_T)-\Phi(x^*,y^*)+\gamma\|A\bar{x}_T+B\bar{y}_T-c\|\right] \le \frac{1}{2T}\left\{nL\|x^*-x_0\|^2+\|y^*-y_0\|_{H_y}^2 +2\rho\left(\frac{\gamma^2}{\rho^2}+\|\alpha_0\|^2\right)\right\}.$$

Remark 1 *When $\Omega=0$, (22) reduces to (19). Hence, Theorem 1 trivially holds when update rule (20) is used.*

However, similar to the other ADMM algorithms, the $(\bar{x}_T,\bar{y}_T)$ obtained from Algorithm 1 may not exactly satisfy the linear constraint $Ax+By=c$. As discussed in (Suzuki, 2013), when B is invertible, the feasibility violation can be reduced to zero by obtaining y from $\bar{x}_T$ as $y(\bar{x}_T)=B^{-1}(c-A\bar{x}_T)$. The following Corollary shows that the $(\bar{x}_T,y(\bar{x}_T))$ pair still have $\mathcal{O}\left(\frac{1}{T}\right)$ convergence towards the objective value.

Corollary 1 *Assume that ψ is $\tilde{L}$-Lipschitz continuous, and B is invertible. Using the update rule (20) or (23), we have*

$$\mathbb{E}\left[\Phi(\bar{x}_T,y(\bar{x}_T))-\Phi(x^*,y^*)\right] \le \frac{1}{2T}\left\{\|x^*-x_0\|_{H_x}^2+nL\|x^*-x_0\|^2+\|y^*-y_0\|_{H_y}^2 +\rho\left(\frac{\tilde{L}^2L_B}{\rho^2}+\|\alpha_0\|^2\right)\right\},$$

where L_B is the largest eigenvalue of $(B^{-1})^TB^{-1}$. When update rule (17) is used,

$$\mathbb{E}\left[\Phi(\bar{x}_T,y(\bar{x}_T))-\Phi(x^*,y^*)\right] \le \frac{1}{2T}\left\{nL\|x^*-x_0\|^2+\|y^*-y_0\|_{H_y}^2 +\rho\left(\frac{\tilde{L}^2L_B}{\rho^2}+\|\alpha_0\|^2\right)\right\}.$$

Alternatively, when A is invertible, one can obtain x from $\bar{y}_T$ as $x(\bar{y}_T) = A^{-1}(c - B\bar{y}_T)$. The following corollary shows that $(x(\bar{y}_T), \bar{y}_T)$ also has $\mathcal{O}\left(\frac{1}{T}\right)$ convergence.

Corollary 2 *Assume that Ω is $\hat{L}$-Lipschitz continuous, and A is invertible. Using the update rule (20) or (23), we have*

$$\begin{aligned}&\mathbb{E}\left[\Phi(x(\bar{y}_T), \bar{y}_T) - \Phi(x^*, y^*)\right] \\ &\leq \frac{1}{2T}\left\{\|x^* - x_0\|_{H_x}^2 + nL\|x^* - x_0\|^2 + \|y^* - y_0\|_{H_y}^2 \right. \\ &\quad \left. +\rho\left(\frac{(\hat{L}+L)^2\hat{L}_A}{\rho^2} + \|\alpha_0\|^2\right)\right\},\end{aligned}$$

where $\hat{L}_A$ is the largest eigenvalue of $(A^{-1})^T A^{-1}$. When update rule (17) is used,

$$\begin{aligned}&\mathbb{E}\left[\Phi(x(\bar{y}_T), \bar{y}_T) - \Phi(x^*, y^*)\right] \\ &\leq \frac{1}{2T}\left\{nL\|x^* - x_0\|^2 + \|y^* - y_0\|_{H_y}^2 \right. \\ &\quad \left. +\rho\left(\frac{(\hat{L}+L)^2\hat{L}_A}{\rho^2} + \|\alpha_0\|^2\right)\right\}.\end{aligned}$$

Remark 2 *The convergence rates obtained here are of the form $\mathcal{O}(\frac{n}{T})$. Recall that batch ADMM has $\mathcal{O}(\frac{1}{T})$ convergence (Section 2.1). It may thus appear that the proposed stochastic algorithms are much slower, especially for large n. However, in the worst case, this is inevitable. For example, consider the lasso problem on the data set $\{(z_i, l_i)\}_{i=1}^n$, where z_i is the input and l_i is the output. This can be formulated as $\phi(x) = \sum_{i=1}^n (l_i - z_i^T x)^2, \psi(y) = \|y\|_1, A = -B = I$, and $c = 0$. Assume that each input sample is nonzero in only one and unique dimension (e.g., $[z_i]_i \neq 0$ and $[z_i]_j = 0\ \forall i \neq j$). Obviously, there are indeed n independent optimization problems. The stochastic algorithm can only update one dimension in each iteration, and so will inevitably be n times slower than the batch algorithm. However, this situation is very rare in practice. Empirical results in Section 4 show that stochastic algorithms always outperform their batch counterparts. It is also interesting to compare with the existing stochastic ADMM algorithms. In this worst-case scenario, $\sigma^2 = \mathcal{O}(n)$, and these algorithms converge as $\mathcal{O}(\frac{\sigma}{\sqrt{T}}) = \mathcal{O}(\frac{\sqrt{n}}{\sqrt{T}})$. When the whole data set is processed more than once (as is typically the case), $T > n$ as each iteration processes only one sample, and $\mathcal{O}(\frac{\sqrt{n}}{\sqrt{T}})$ is worse than our $\mathcal{O}(\frac{n}{T})$ rate.*

Instead of updating the gradient of only one sample in each iteration, extension to the use of mini-batch is straightforward. This is particularly useful when (i) the gradients of the multiple samples in the mini-batch can be computed in parallel; or (ii) the other steps are much more expensive than computing the gradient of one single sample. The analysis in this section still holds, with n simply replaced by $\frac{n}{n_b}$, where n_b is the mini-batch size. This can thus also be used to reduce the essential value of n.

Remark 3 *Compared with the existing stochastic ADMM algorithms, the faster convergence of Algorithm 1 comes at the cost of an extra $\mathcal{O}(nd_x)$ memory for the historical information used in Step 4. This is similar to the algorithms in (Le Roux et al., 2012; Mairal, 2013).*

Remark 4 *The present analysis considers convergence w.r.t. the training objective (or a combination with the feasibility violation), but does not directly bound the generalization performance. Intuitively, for any algorithm that only have access to a finite training set, it is impossible to achieve perfect generalization even after infinite iterations. Nevertheless, improved generalization performance is observed in our experiments (Section 4). Moreover, while Agarwal et al. (2012) show that $\mathcal{O}(\frac{1}{\sqrt{T}})$ is the optimal convergence rate for any stochastic optimization scheme on general convex problems, this does not contradict our analysis as we further assume a finite training sample.*

4. Experiments

In this section, we perform experiments on the generalized lasso model (Tibshirani & Taylor, 2011): $\min_x \frac{1}{n}\sum_{i=1}^n \ell_i(x) + \lambda\|Ax\|_1$, where λ is the regularization parameter, and A is a penalty matrix specifying the desired structured sparsity pattern of x. With different settings of A, this can be reduced to models such as the fused lasso, trend filtering, and wavelet smoothing. Here, we will focus on the graph-guided fused lasso (Kim et al., 2009). As in (Ouyang et al., 2013), the graph is obtained by sparse inverse covariance selection (Banerjee et al., 2008), and $A = [G; I]$ for the sparsity pattern G. Moreover, as classification problems are considered here, we use the logistic loss instead of the square loss.

While proximal methods have been used in the optimization of graph-guided fused lasso (Barbero & Sra, 2011; Liu et al., 2010), in general, the underlying proximal step is difficult to solve because of the presence of A. ADMM, by splitting the objective as $\phi(x) = \frac{1}{n}\sum_{i=1}^n \ell_i(x), \psi(y) = \lambda\|y\|_1$ and with constraint $Ax = y$, has been shown to be more efficient (Ouyang et al., 2013; Suzuki, 2013). In this experiment, we compare

1. two variants of the proposed method: SA-ADMM, which uses update rule (17); and SA-IU-ADMM, which uses (20)) based on the inexact Uzawa method;
2. three existing stochastic ADMM algorithms: STOC-ADMM (Ouyang et al., 2013); OPG-ADMM (Suzuki, 2013); and RDA-ADMM (Suzuki, 2013);

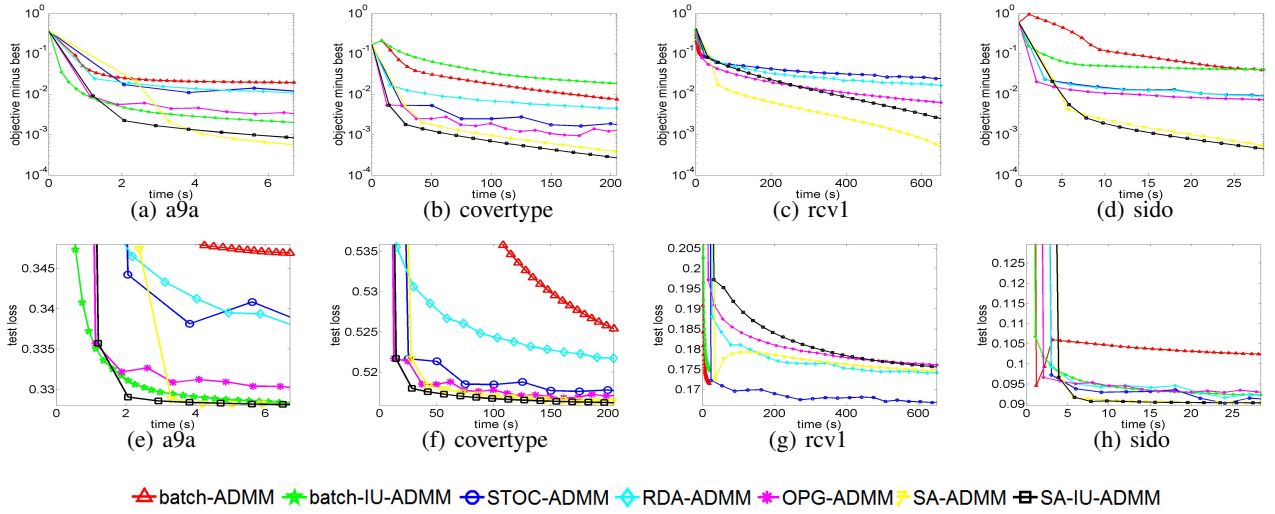

Figure 1. Performance versus running time on general convex problems. Top: objective value; Bottom: testing loss.

3. two deterministic ADMM variants: batch-ADMM, which is the batch version of SA-ADMM using full gradient (i.e., $\tau_i(t) = t \ \forall i$); and batch-IU-ADMM, which is the batch version of SA-IU-ADMM.

All these share the same update rules for y and α (i.e., steps 5 and 6 in Algorithm 1), and differ only in the updating of x, which are summarized in Table 1. We do not compare with the online ADMM (Wang & Banerjee, 2012) or a direct application of the batch ADMM, as an expensive matrix inverse is required for the update of x in (10). Moreover, it has been shown that the online ADMM is slower than RDA-ADMM (Suzuki, 2013).

Table 1. A comparison of the update rules on x.

	gradient computation	linearize $\|Ax+By-c\|^2$?	constant stepsize?
SA-ADMM	average	no	yes
SA-IU-ADMM	average	yes	yes
STOC-ADMM	one sample	no	no
OPG-ADMM	one sample	yes	no
RDA-ADMM	average (history)	yes	no
batch-ADMM	all samples	no	yes
batch-IU-ADMM	all samples	yes	yes

Experiments are performed on four popular binary classification data sets[2] (Table 2) (Le Roux et al., 2012; Suzuki, 2013). For each data set, half of the samples are used for training, while the rest for testing. To reduce statistical variability, results are averaged over 10 repetitions. To ensure that the ADMM iterates satisfy the constraint during the iterations, we report the performance using $(x_t, y(x_t))$ as discussed in Section 3.4. Moreover, we fix the regularization parameter λ to 10^{-5} for *a9a, covertype*, and to 10^{-4} for *rcv1* and *sido*. For ρ in (3) and the stepsize (or its proportionality constant), we use a small training subset with 500 samples, and choose the parameter setting with the smallest training objective value after running the stochastic algorithm over 5 data passes; or after running the batch algorithm for 100 iterations. As the proposed methods require historical information, we initialize them with OPG-ADMM for the first n iterations. All methods are implemented in MATLAB, and experiments are performed on a PC with an Intel i7-2600K CPU and 32GB memory.

Table 2. Summary of data sets.

data set	number of samples	dimensionality
a9a	32,561	123
covertype	581,012	54
rcv1	20,242	47,236
sido	12,678	4,932

Figure 1 shows the objective value and testing loss obtained by the various algorithms versus running time. Overall, SA-ADMM and SA-IU-ADMM are the fastest and rapidly lead to a model with good generalization performance. These are followed by the other stochastic algorithms, whie the batch ADMM algorithms are the slowest.

As in (Mairal, 2013), we also study the variation of the objective value with the number of "effective passes" over the data[3]. For a batch algorithm, one effective pass is the same as one iteration; whereas for a stochastic algorithm, one effective pass is equal to n iterations (as each iteration processes only one sample). Results are shown in Figure 2. As can be seen, the relative performance of the various al-

[2] *a9a*, *covertype* and *rcv1* are from the LIBSVM archive, and *sido* from the Causality Workbench website.

[3] Because of the lack of space, the variation of the testing loss with the number of effective passes is not shown here. Similar to the objective value, the trend is similar to that in Figure 1.

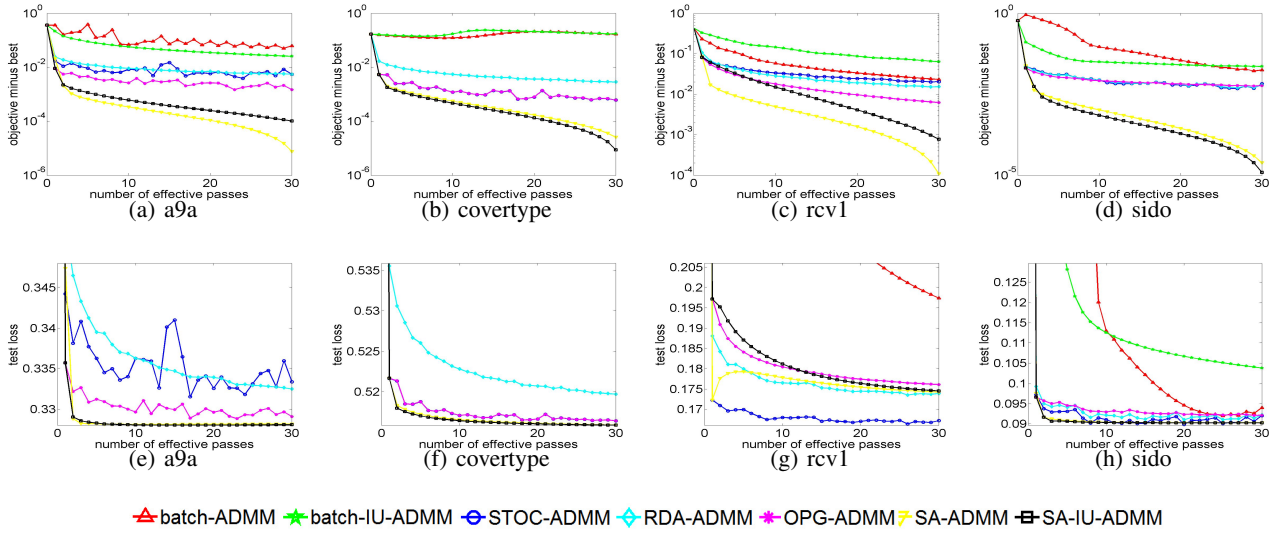

Figure 2. Performance versus the number of effective passes on general convex problems. Top: objective value; Bottom: testing loss.

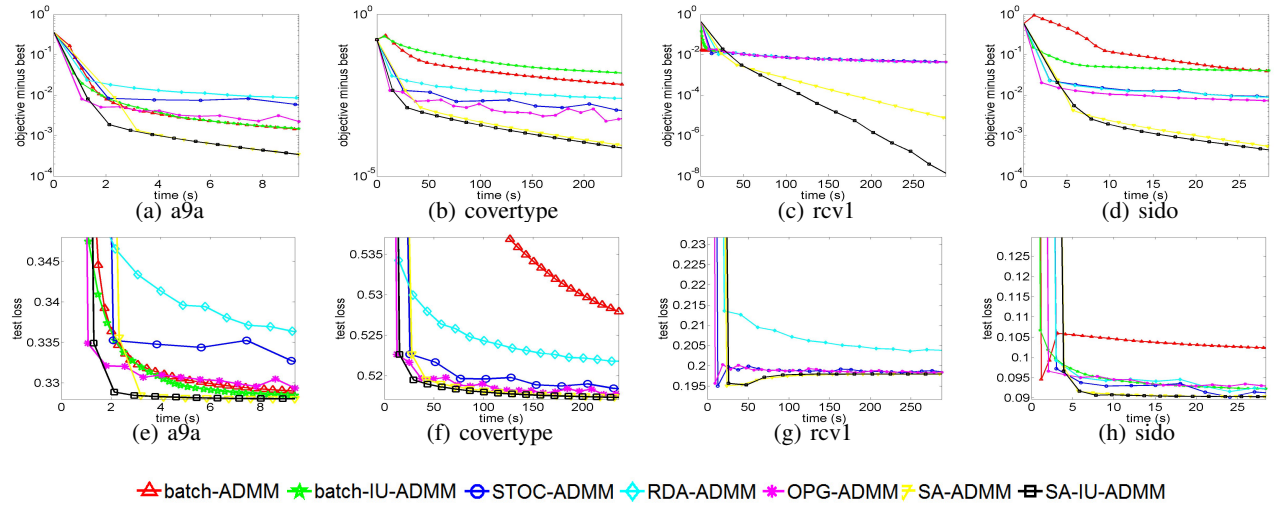

Figure 3. Performance versus the running time on strongly convex problems. Top: objective value; Bottom: testing loss.

gorithms is similar to that in Figure 1. In other words, not only is the total time required by the proposed algorithms shorter, the number of iterations required is also smaller.

As discussed in Section 2.1, STOC-ADMM and OPG-ADMM has $\mathcal{O}(\frac{\log T}{T})$ convergence when the loss is strongly convex. It is still an open question whether this also holds for the proposed algorithm. In the following, we compare their performance empirically by adding an extra ℓ_2-regularizer on x to the model. Results are shown in Figure 3. As can be seen, the improvement of SA-IU-ADMM over others is even more dramatic.

5. Conclusion

In this paper, we developed a novel stochastic algorithm that incrementally approximates the full gradient in the linearized ADMM formulation. It enjoys the same computational simplicity as existing stochastic ADMM algorithms, but has a fast convergence rate that matches the batch ADMM. Empirical results on both general convex and strongly convex problems demonstrate its efficiency over batch and stochastic ADMM algorithms. In the future, we will investigate the theoretical convergence rate of the proposed algorithm on strongly convex problems.

Acknowledgments

This research was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region (Grant 614311).

References

Agarwal, A., Bartlett, P.L., Ravikumar, P., and Wainwright, M.J. Information-theoretic lower bounds on the oracle complexity of stochastic convex optimization. *IEEE Transactions on Information Theory*, 58(5):3235–3249, 2012.

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. Convex optimization with sparsity-inducing norms. In Sra, S., Nowozin, S., and Wright, S.J. (eds.), *Optimization for Machine Learning*, pp. 19–53. MIT Press, 2011.

Banerjee, O., El Ghaoui, L., and d'Aspremont, A. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, 2008.

Barbero, A. and Sra, S. Fast Newton-type methods for total variation regularization. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 313–320, New York, NY, USA, 2011.

Bottou, L. Stochastic learning. In *Advanced Lectures on Machine Learning*, pp. 146–168. Springer Verlag, 2004.

Boyd, S. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2010.

Deng, W. and Yin, W. On the global and linear convergence of the generalized alternating direction method of multipliers. Technical Report TR12-14, Rice University, 2012.

Duchi, J. and Singer, Y. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2873–2908, 2009.

Gabay, D. and Mercier, B. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

Glowinski, R. and Marrocco, A. Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualite, d'une classe de problems de dirichlet non lineares. *Revue Francaise d'Automatique, Informatique, et Recherche Opérationelle*, 9:41–76, 1975.

He, B. S. and Yuan, X. M. On the $O(1/t)$ convergence rate of alternating direction method. *SIAM Journal on Numerical Analysis*, 22(4):1431–1448, 2012.

Hong, M. and Luo, Z.-Q. On the linear convergence of the alternating direction method of multipliers. Technical Report arXiv:1208.3922, University of Minnesota, 2012.

Kim, S., Sohn, K.-A., and Xing, E.P. A multivariate regression approach to association analysis of a quantitative trait network. *Bioinformatics*, 25(12):i204–i212, 2009.

Le Roux, N., Schmidt, M., and Bach, F. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems 25*, pp. 2672–2680, 2012.

Liu, J., Yuan, L., and Ye, J. An efficient algorithm for a class of fused lasso problems. In *Proceedings of the 16th International Conference on Knowledge Discovery and Data Mining*, pp. 323–332, Washington, DC, USA, 2010.

Mairal, J. Optimization with first-order surrogate functions. In *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, GA, USA, 2013.

Ouyang, H., He, N., Tran, L., and Gray, A. Stochastic alternating direction method of multipliers. In *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, GA, USA, 2013.

Qin, Z. and Goldfarb, D. Structured sparsity via alternating direction methods. *Journal of Machine Learning Research*, 13:1435–1468, 2012.

Shalev-Shwartz, S. and Zhang, T. Proximal stochastic dual coordinate ascent. *arXiv:1211.2717*, 2013a.

Shalev-Shwartz, S. and Zhang, T. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14:437–469, 2013b.

Suzuki, T. Dual averaging and proximal gradient descent for online alternating direction multiplier method. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 392–400, Atlanta, GA, USA, 2013.

Tibshirani, R. J. and Taylor, J. The solution path of the generalized lasso. *Annals of Statistics*, 39(3):1335–1371, 2011.

Wang, H. and Banerjee, A. Online alternating direction method. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 1119–1126, Edinburgh, Scotland, UK, 2012.

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

Zhang, X., Burger, M., and Osher, S. A unified primal-dual algorithm framework based on Bregman iteration. *Journal of Scientific Computing*, 46(1):20–46, January 2011.