Accelerated Alternating Direction Method of Multipliers

Mojtaba Kadkhodaie
Department of Electrical and Computer Engineering
University of Minnesota
kadkh004@umn.edu

Konstantina Christakopoulou
Department of Computer Science and Engineering
University of Minnesota
christa@cs.umn.edu

Maziar Sanjabi
Department of Electrical and Computer Engineering
University of Minnesota
maz@umn.edu

Arindam Banerjee
Department of Computer Science and Engineering
University of Minnesota
banerjee@cs.umn.edu

ABSTRACT

Recent years have seen a revival of interest in the Alternating Direction Method of Multipliers (ADMM), due to its simplicity, versatility, and scalability. As a first order method for general convex problems, the rate of convergence of ADMM is $O(1/k)$ [4, 25]. Given the scale of modern data mining problems, an algorithm with similar properties as ADMM but faster convergence rate can make a big difference in real world applications. In this paper, we introduce the Accelerated Alternating Direction Method of Multipliers (A2DM2) which solves problems with the same structure as ADMM. When the objective function is strongly convex, we show that A2DM2 has a $O(1/k^2)$ convergence rate. Unlike related existing literature on trying to accelerate ADMM, our analysis does not need any additional restricting assumptions. Through experiments, we show that A2DM2 converges faster than ADMM on a variety of problems. Further, we illustrate the versatility of the general A2DM2 on the problem of learning to rank, where it is shown to be competitive with the state-of-the-art specialized algorithms for the problem on both scalability and accuracy.

Keywords

Alternating Direction Method of Multipliers, Ranking on the Top of the List

1. INTRODUCTION

In the past decade the advances in many areas, such as data mining and machine learning, has led to formulating many arising problems into an optimization formulation. Therefore, the proposed methodologies in these areas, require solving an optimization problem in their core and their applicability is dependent on solving such problems as fast and efficiently as possible. The proposed algorithms for solving such optimization problems should remain efficient as the size of the problem grows. This scalability criterion would cross out many traditional optimization methods such as interior point methods [21] and Newton method which are based on complicated iterations (due to requiring the second order information and matrix inversion). An alternative approach is to use first order methods that have lower cost per iteration, but show slower convergence. Unfortunately, general first order methods [15, 2, 10] require projections to the problem solution set. Such projections, even for solution sets that are defined by simple linear constraints, can be intractable in high dimensions. Therefore the applicability of such methods is limited.

Our focus in this work is on an alternative procedure that helps us deal with linear equality constraints. The algorithm that we will discuss is called Alternating Direction Method of Multiplier (ADMM) which has a long history in literature. It was first proposed in [12] and has recently gained lots of attention [4, 25] due to its simplicity and wide range of problems that it covers. Specifically, ADMM is designed for solving convex optimization problems of the form

$$\min_{x,y} f_1(x) + f_2(y) \quad \text{subject to} \quad Ax + By = c \tag{1}$$

where $x \in \mathbb{R}^{n_1}$, $y \in \mathbb{R}^{n_2}$ are the optimization variables, $A \in \mathbb{R}^{m \times n_1}$, $B \in \mathbb{R}^{m \times n_2}$ are linear operators, $c \in \mathbb{R}^m$ is a vector of data, and finally f_1 and f_2 are closed convex functions. As the formulation suggests, the objective is separable across the variables, while the constraints are coupling the variables. Such coupling linear equality constraints are not easy to deal with in general.

ADMM [4, 25] is an iterative method that uses a Gauss Seidel type update to solve (1). Given a penalty parameter $\tau > 0$, ADMM minimizes the augmented Lagrangian function

$$L(x, y, \lambda) = f_1(x)+f_2(y)-\langle\lambda, Ax+By-c\rangle+\frac{\tau}{2}\|c-Ax-By\|^2$$

with respect to x and y alternatively and then updates the dual variable $\lambda \in \mathbb{R}^m$. Steps of ADMM are summarized in Algorithm 1.

One of the main advantages of ADMM framework to other methods is its flexibility towards parallel computation [4]. As a result, in many applications it might be favorable to

KDD '15, August 10-13, 2015, Sydney, NSW, Australia.
© 2015 ACM. ISBN 978-1-4503-3664-2/15/08…$15.00.
http://dx.doi.org/10.1145/2783258.2783400.

cast an unconstrained optimization problem into a constrained form (by introducing new variables) and solve the resulting constrained formulation by ADMM in a parallel fashion (for examples of this type of reformulation, see [4]).

Note that the effectiveness of ADMM depends on the simplicity of its updates for x and y in Algorithm 1. There are other variations of ADMM that consider inexact updates for x and y in order to make the algorithm more tractable in practice (for such inexact variants of ADMM see [25, 4]).

The iteration complexity of ADMM has been extensively studied in the literature (see [25] and the references therein). It is shown that the algorithm has $\mathcal{O}(1/k)$ convergence rate [4, 25] under some mild conditions on the problem. Recently, some variants of this algorithm were studied which exhibit faster convergence rates while requiring only a little change in the computational effort of each iteration [14, 13]. The acceleration methods considered in these works are of the form first proposed by Nesterov [20] for gradient descent algorithms. Nesterov's accelerated gradient descent scheme in [20] was initially designed for solving unconstrained smooth convex problems and was shown to provide a $\mathcal{O}(1/k^2)$ rate of convergence. His acceleration scheme has inspired many researchers to develop accelerated variants of other existing iterative methods (for instance see [3] which proposes an accelerated variant of the proximal splitting method).

In [13], the authors propose a Nesterov-type acceleration of ADMM for problems of the form (1) in the special case where both A and B are identity matrices, and one of f_1 or f_2 is differentiable. Their accelerated scheme has $\mathcal{O}(1/k^2)$ convergence rate and is based on the "symmetric" ADMM method, which differs from Algorithm 1 in that it involves two dual updates per iteration rather than one. The authors handle weakly convex problems by introducing a "step-skipping" process that applies the acceleration selectively on certain iterations. However, the step-skipping process turns the algorithm to one with a more complicated sequence of steps than conventional ADMM. The major drawback of their analysis is that it requires the matrices A and B to be identity. Such assumption restricts the application of their accelerated version of ADMM.

In a related work [14], it was shown that by applying Nesterov's acceleration scheme, ADMM can have a $\mathcal{O}(1/k^2)$ convergence rate provided that some assumptions hold true about the problem. The proposed accelerated method is simply ADMM with a predictor-corrector type acceleration step. The convergence rate of this algorithm is analyzed in [14] under the assumptions that both objective terms are strongly convex and one of them is quadratic. These assumptions enable the authors to use a similar proof technique as in [20] to show fast convergence of their algorithm. Instead of analyzing the convergence rate of the algorithm in terms of decrease in the primal objective sequence, [14] considers the dual problem of (1) which involves maximizing the dual function

$$\max_{\lambda} D(\lambda) \overset{\text{def}}{=} -f_1^*(A^T\lambda) - f_2^*(B^T\lambda) + \langle \lambda, c\rangle \qquad (2)$$

where f_1^* and f_2^* are Fenchel conjugate functions [11] of f_1 and f_2, respectively, defined as

$$F^*(u) = \max_{v} \langle u, v\rangle - F(v), \qquad (3)$$

for any closed convex function F. In the case where f_1 and f_2 are strongly convex, the conjugate functions turn out to be smooth with Lipschitz continuous gradients and hence the dual problem (2) simply becomes an unconstrained smooth convex optimization. As a result, if an accelerated gradient ascent method, as the one in [20], is applied to the dual problem (2), a $\mathcal{O}(1/k^2)$ convergence rate will be obtained. However, since the ADMM algorithm exploits inexact gradient ascent type of update for the dual variable λ, a further technical condition needs to be satisfied in every iteration of the accelerated algorithm. Therefore, in order to prove the iteration complexity of accelerated ADMM, the authors in [14] require both functions f_1 and f_2 to be strongly convex as well as f_2 to be quadratic.

In this paper, we introduce a novel algorithm called Accelerated Alternating Direction Method of Multipliers (A2DM2), and prove that the algorithm has a $O(1/k^2)$ convergence rate as long as f_1, f_2 are strongly convex. In particular, unlike [13], the functions need not be differentiable, and unlike [14], neither of them needs to be quadratic. Further, the analysis works out without any restricting assumptions on the matrices A and B. The analysis technique is similar to the ones in [14, 20]. To illustrate the versatility of the proposed A2DM2, we consider the problem of learning to rank with emphasis on accuracy at the top of the list, and show how A2DM2 can be applied to the problem. Through extensive empirical evaluation on a wide variety of datasets, we illustrate that A2DM2 is competitive, often by an order of magnitude, to specialized algorithms designed for the ranking problem. We also show the generality and wide applicability of A2DM2 by highlighting other problems, including dirty statistical models and elastic net problems, where it is readily applicable.

The rest of the paper is organized as follows. In Section 2, we introduce the Accelerated ADMM (A2DM2) algorithm, and prove the $O(1/k^2)$ convergence rate of the algorithm. In Section 3, we present the problem of learning to rank and illustrate how A2DM2 can be applied to solve the problem. In Section 4, we present experimental results comparing A2DM2 with ADMM on synthetic data, dirty statistical models, and elastic nets. In Section 5, we present extensive comparisons of A2DM2 on the learning to rank problem with the state-of-the-art and basic ADMM in terms of both optimization time and accuracy. We conclude in Section 6.

2. ACCELERATED ADMM ALGORITHM

In this section we introduce our accelerated ADMM algorithm, which we call A2DM2, for solving (1). First, we need to assume strong convexity of f_1, and f_2 with corresponding constants σ_1 and σ_2, i.e. for $i = 1, 2$ and every $x, x^{'} \in \mathbb{R}^{n_i}$

$$f_i(x) - f_i(x') \geq \langle g, x - x'\rangle + \frac{\sigma_i^2}{2}\|x - x'\|^2, \ \forall\ g \in \partial f_i(x'),$$

where $\partial f_i(\cdot)$ denotes the sub-differential set of f_i. As a result of strong convexity of f_i, $i = 1, 2$, the conjugate function, defined in (3), would have a Lipschitz continuous gradient with constant $1/\sigma_i$, $i = 1, 2$.

Now we are ready to define the A2DM2 algorithm. Our accelerated ADMM algorithm uses Nesterov's method to extrapolate the update of λ in each iteration of ADMM. In order to guarantee the convergence, it is required to update the variable y based on this extrapolated version of λ. The overall algorithm is summarized in Algorithm 2. Compared to the conventional ADMM algorithm, our method requires extrapolating λ and updating y twice; therefore, each iteration for accelerated ADMM would be more costly than the usual

Algorithm 1: ADMM

```
Require: $y_0 \in \mathbb{R}^{n_2}$, $\lambda_0 \in \mathbb{R}^m$, $\tau > 0$
1. for $k = 0, 1, 2, 3, \cdots$ do
2.     $x_{k+1} = \text{argmin}_x L(x, y_k, \lambda_k)$
3.     $y_{k+1} = \text{argmin}_y L(x_{k+1}, y, \lambda_k)$
4.     $\lambda_{k+1} = \lambda_k - \tau(Ax_{k+1} + By_{k+1} - c)$
5. end for
```

Algorithm 2: Accelerated ADMM (A2DM2)

```
Require: $y_0 = \hat{y}_0 \in \mathbb{R}^{n_2}$, $\lambda_0 = \hat{\lambda}_0 \in \mathbb{R}^m$, $\tau > 0$, $a_0 = 1$
1. for $k = 0, 1, 2, 3, \cdots$ do
2.     $x_k = \text{argmin}_x L(x, \hat{y}_k, \hat{\lambda}_k)$
3.     $y_k = \text{argmin}_y L(x_k, y, \hat{\lambda}_k)$
4.     $\lambda_k = \hat{\lambda}_k - \tau(Ax_k + By_k - c)$
5.     $a_{k+1} = \frac{1+\sqrt{1+4a_k^2}}{2}$
6.     $\hat{\lambda}_{k+1} = \lambda_k + \frac{a_k - 1}{a_{k+1}}(\lambda_k - \lambda_{k-1})$
7.     $\hat{y}_{k+1} = \text{argmin}_y f_2(y) + \langle \hat{\lambda}_{k+1}, -By \rangle$
8. end for
```

ADMM. But as we will see in the numerical experiments, in many scenarios this extra cost is negligible compared to the speed-up that is gained using accelerated ADMM.

One appealing feature of ADMM, which makes it suitable for large-scale problems, is the capability of being executed on parallel machines. Similar to ADMM, A2DM2 is also parallelizable. This is because the primal and dual updates of A2DM2 include the ones for ADMM (steps 2-4 of Algorithm 2). The extra update in A2DM2 for the dual variable $\hat{\lambda}$ (step 6) is an element-wise operation, which is easy to parallelize. Moreover, the additional variable $\hat{y}$ is iteratively set to the minimizer of the associated objective term f_2 augmented by a linear function. This can also be done in parallel provided that f_2 is decomposable across variables.

Regarding the convergence rate of the algorithm, the following theorem states $O(1/k^2)$ convergence of A2DM2.

Theorem 1. *Suppose that f_1 and f_2 are strongly convex with σ_1, σ_2. Moreover, assume that $\tau^3 \leq \frac{\sigma_1 \sigma_2^2}{\rho(A^T A)\rho^2(B^T B)}$, where $\rho(\cdot)$ denotes the maximum singular value of the matrix. Then the iterates λ_k generated by Algorithm 2 would satisfy*

$$D(\lambda^*) - D(\lambda_k) \leq \frac{2\|\hat{\lambda}_1 - \lambda^*\|^2}{\tau(k+2)^2}, \tag{4}$$

where λ^* is an optimal solution of the dual problem (2).

Proof. For an optimal Lagrange multiplier λ^* of (2), define $s_k = a_k \lambda_k - (a_k - 1)\lambda_{k-1} - \lambda^*$. Then using the following lemma helps us establish Theorem 1.

Lemma 1. *For the sequence s_k defined as $s_k = a_k\lambda_k - (a_k - 1)\lambda_{k-1} - \lambda^*$ the assumptions of Theorem 1 imply*

$$\|s_{k+1}\|^2 - \|s_k\|^2 \leq 2a_k^2\tau(D(\lambda^*) - D(\lambda_k)) - 2a_{k+1}^2\tau(D(\lambda^*) - D(\lambda_{k+1})). \tag{5}$$

Now using Lemma 1, it is easy to see that [1]

$$2a_{k+1}^2\tau(D(\lambda^*) - D(\lambda_{k+1})) \leq 2a_k^2\tau(D(\lambda^*) - D(\lambda_k)) + \|s_k\|^2.$$

Furthermore, rewriting (5) and using induction, it is easy to see that

$$\|s_k\|^2 + 2a_k^2\tau(D(\lambda^*) - D(\lambda_k)) \leq \|s_1\|^2 + 2a_1^2\tau(D(\lambda^*) - D(\lambda_1)), \quad \forall\ k. \tag{6}$$

Now in order to prove the result, we need the following lemma, for which the proof is relegated to the appendix.

Lemma 2. *When the conditions of Theorem 1 are satisfied, then for any $\gamma \in \mathbb{R}^m$, $\forall k$,*

$$D(\lambda_{k+1}) - D(\gamma) \geq \frac{1}{\tau}\langle \gamma - \hat{\lambda}_{k+1}, \hat{\lambda}_{k+1} - \lambda_{k+1} \rangle + \frac{1}{2\tau}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2. \tag{7}$$

Applying Lemma 2 with $k = 0$ and $\gamma = \lambda^*$, we get

$$D(\lambda_1) - D(\lambda^*) \geq \frac{1}{\tau}\langle \gamma - \hat{\lambda}_1, \hat{\lambda}_1 - \lambda_1 \rangle + \frac{1}{2\tau}\|\lambda_1 - \hat{\lambda}_1\|^2 = \frac{1}{2\tau}\Big(\|\lambda_1 - \lambda^*\|^2 - \|\hat{\lambda}_1 - \lambda^*\|^2\Big). \tag{8}$$

Combining (6) and (8) plus using definition of $s_1 = \lambda_1 - \lambda^*$ yields

$$2a_k^2\tau(D(\lambda^*) - D(\lambda^k)) \leq \|\hat{\lambda}_1 - \lambda^*\|^2.$$

Note that we have ignored the term $\|s_k\|^2 \geq 0$ on the left hand side of (6). In order to get the final result, note that $a_k > a_{k-1} + \frac{1}{2} > 1 + \frac{k}{2}$. Thus, $D(\lambda^*) - D(\lambda_k) \leq \frac{2\|\hat{\lambda}_1 - \lambda^*\|^2}{\tau(k+2)^2}$. □

In the convergence analysis of ADMM, primal and dual residuals play an important role [25, 14]. For the accelerated ADMM algorithm, the primal and dual residuals can also be defined as $r_k \overset{\text{def}}{=} b - Ax_k - By_k = \lambda_k - \hat{\lambda}_k$ and $d_k \overset{\text{def}}{=} A^T B(y_k - \hat{y}_k)$, respectively. It can be shown that for A2DM2 these residuals will decrease in $O(1/k^2)$ (The proof is similar to Lemma 6 of [14]. However, because of the different update rule for $\hat{y}_k$, it can be shown through Lemma (1) that f_2 is no longer needed to be quadratic.) As we will see in the next section we can use these residuals to propose another variant of the accelerated ADMM.

2.1 Accelerated ADMM with Restarting

Here we introduce a variant of A2DM2 to address the issue of possible spiral movements around the optimal solution, which is quite common among accelerated algorithms

[1] Lemma 1 can be proved in a similar way as Lemma 5 of [14] except that since f_2 is not restricted here to be a quadratic function, we need our Lemma 2 to complete the proof.

[9]. One common way to reduce such movements is to use a restarting rule. Similar to [14], in order to find out when the good time to restart is, we use the sum of two terms, which are proportional to the primal and dual residuals respectively,

$$m_k \overset{\text{def}}{=} \frac{1}{\tau}\|\lambda_k - \hat{\lambda}_k\|^2 + \tau\|B(y_k - \hat{y}_k)\|^2. \tag{9}$$

At every iteration of the accelerated algorithm with restarting, which we often refer to as A2DM2+Restart, we compare m_k with m_{k-1} and if $m_k > \eta\, m_{k-1}$, where $0 < \eta < 1$ is a constant close to one, we restart the method by setting $a_{k+1} = 1$, $\hat{y}_{k+1} = y_k$ and $\hat{\lambda}_{k+1} = \lambda_k$.

Interestingly, our empirical studies show that while in some cases restarting really helps to improve the performance of A2DM2 (see section 4), in others its performance is inferior (see section 5).

3. TOP RANKING OPTIMIZATION

Now we focus on using the A2DM2 framework to solve the problem of ranking on the top of a list. The goal is to provide an alternative optimization solution for pushing down the highest ranked negative example in the ranked list. In this section, we appropriately reformulate the TopPush optimization problem [18], which is the most efficient pairwise ranking solution up to date, and derive its updates within the A2DM2 framework. This results in an algorithm competitive with the TopPush algorithm of [18], in terms of both ranking performance and computational efficiency.

3.1 Related Work

Ranking problems are prevalent in a wide range of domains where a long list of objects, such as web links or products, needs to be ranked. Typically, in such scenarios, what matters the most is the quality of ranking near the top of the ranked list. Towards this direction, a number of learning to rank algorithms which put more emphasis towards the top of the ranked list have been developed (see [5, 6, 7, 19, 22] and the references therein.)

One main group of ranking algorithms aims to optimize a convex upper-bound of specific metrics which look at the top of the ranked list. Examples of such metrics are Average Precision and Normalized Discounted Cumulative Gain [17, 6]. Another line of work was initiated by the so-called p-Norm Push ranking method [23] which optimizes a novel measure that concentrates harder on the high ranked negative examples and *pushes* them down. Extending [23], Infinite Push [1] seeks to push down the *top* irrelevant item, by minimizing the maximum number of positive examples ranked below any negative. Both [23] and [1] are pairwise ranking approaches, thus inheriting the downside of computational cost proportional to the number of positive-negative pairs, which is prohibitive for large datasets. Recently, [18] addressed this issue by reformulating the Infinite Push objective as the number of positive examples ranked below the highest ranked negative. This results in a pairwise ranking algorithm, named TopPush, with time complexity linear in the number of training instances.

3.2 Bipartite Ranking

Consider the bipartite setup for ranking in which the samples are either relevant (positive) or irrelevant (negative). Assume that $\mathcal{X} \subseteq \mathbb{R}^d$ is the instance space and that we are given the training sample $S = (S_+, S_-) \in \mathcal{X}^m \times \mathcal{X}^n$, where $S_+ = (x_1^+, \cdots, x_m^+)$ and $S_- = (x_1^-, \cdots, x_n^-)$ are positive and negative samples, respectively. As in [18], our goal is to learn a ranking function $f : \mathcal{X} \to \mathbb{R}$ that maximizes the number of positive instances that are ranked higher than the top-ranked negative sample. In other words, the ranking task is translated into minimizing the following loss over the choice of f

$$\mathcal{L}(f;S) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}\left(f(x_i^+) \le \max_{1\le j\le n} f(x_j^-)\right) \tag{10}$$

where $\mathbb{I}(\cdot)$ is the indicator function which equals one if the input argument is true and zero otherwise. Here, we restrict ourselves to the class of linear scoring functions, i.e. $f(x) = w^T x$ for some weight vector $w \in \mathbb{R}^d$. Since the indicator function $\mathbb{I}(\cdot)$ is not convex, [18] suggests replacing it with a convex loss function $\ell(\cdot)$ and then solves the following problem

$$\min_{w\in\mathbb{R}^d} \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \ell(\max_{1\le j\le n} w^T x_j^- - w^T x_i^+) \tag{11}$$

where the regularization term $\frac{\lambda}{2}\|w\|^2$ is added to avoid overfitting. The loss function $\ell(\cdot)$ is further assumed to be non-decreasing and differentiable. When the loss is the truncated quadratic loss, i.e. $\ell(z) = ([1+z]_+)^2$, the authors of [18] solve the dual of (11) by using an accelerated gradient projection algorithm. Instead, A2DM2 solves (11) in its primal form. Even though in the sequel we restrict ourselves to the truncated quadratic loss, our framework is general enough to incorporate any other appropriate loss function.

3.3 A2DM2 for Ranking

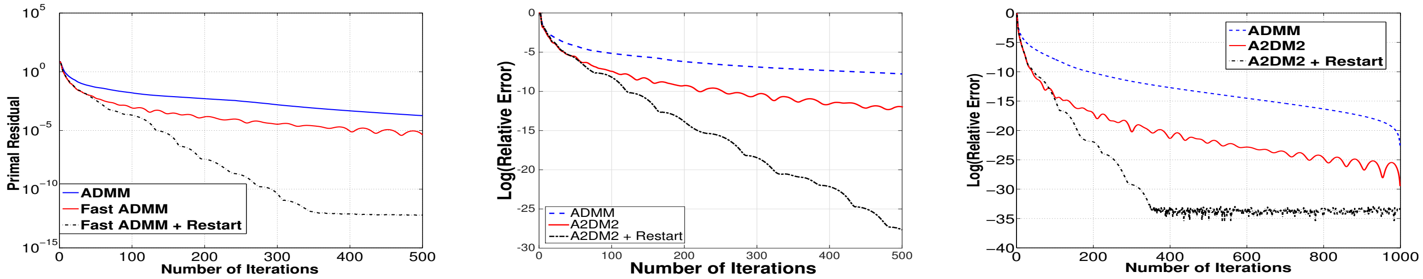
We first illustrate how ADMM can be applied to solving (11). Define $a \overset{\text{def}}{=} \max_j w^T x_j^-$, then (11) can be cast as

$$\begin{aligned} \min_{w\in\mathbb{R}^d, a\in\mathbb{R}} \quad & \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum \ell(a - w^T x_i^+) \\ \text{subject to} \quad & a = \max_{1\le j\le n} w^T x_j^- \end{aligned}$$

Since the above constraint is not linear in w, we need to define further extra variables. Let $s_j \overset{\text{def}}{=} w^T x_j^- - a$, $j = 1, \cdots, n$. Note that s_j has to be non-positive since a is, by definition, the maximum of all linear combinations $w^T x_j^-$ for $j = 1, \cdots, n$. Moreover, let $b_i \overset{\text{def}}{=} a - w^T x_i^+$, $i = 1, \cdots, m$ and then the above problem translates to

$$\begin{aligned} \min_{w\in\mathbb{R}^d, a\in\mathbb{R}, b\in\mathbb{R}^m} \quad & \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m} \ell(b_i) \\ \text{subject to} \quad & s_j = w^T x_j^- - a, \ 1 \le j \le n \\ & b_i = a - w^T x_i^+, \ 1 \le i \le m \\ & s_j \le 0, \ 1 \le j \le n. \end{aligned}$$

To write the constraints in a compact form, we stack the negative and positive samples as rows of $\boldsymbol{X^-}$ and $\boldsymbol{X^+}$, respectively, and let $\boldsymbol{X^-} \overset{\text{def}}{=} \left[(x_1^-)^T; (x_2^-)^T; \ldots; (x_n^-)^T\right] \in \mathbb{R}^{n\times d}$ and $\boldsymbol{X^+} \overset{\text{def}}{=} \left[(x_1^+)^T; (x_2^+)^T; \ldots, (x_m^+)^T\right] \in \mathbb{R}^{m\times d}$. Also defining the vector $s = [s_1, s_2, \cdots, s_n]^T \in \mathbb{R}^n$, the problem be-



(a) Primal Residual

(b) Primal Objective Optimality Distance

(c) Dual Objective Optimality Distance

Figure 1: Convergence behavior of ADMM, A2DM2 and A2DM2 + Restart for the Elastic net with ℓ_1 regularization problem. A2DM2 performs better than the ADMM. ADM2+Restart has the best performance.

comes

$$\begin{aligned} \min_{\substack{w\in\mathbb{R}^d, a\in R\\ b\in\mathbb{R}^m, s\in\mathbb{R}^n_-}} \quad & \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m}\ell(b_i) \\ \text{subject to} \quad & a\mathbf{1}_n + s = \boldsymbol{X}^- w \\ & b = a\mathbf{1}_m - \boldsymbol{X}^+ w, \quad s \leq 0 \end{aligned} \tag{12}$$

where $\mathbf{1}_m$ and $\mathbf{1}_n$ are all-one vectors of lengths m and n. Introducing dual variables $\gamma_1 \in \mathbb{R}^n$ and $\gamma_2 \in \mathbb{R}^m$ corresponding to the first and second sets of linear constraints, we can formulate the augmented Lagrangian as

$$\begin{aligned} \mathrm{L}(w, a, b, s, \gamma_1, \gamma_2) = & \frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m}\ell(b_i) \\ & + \gamma_1^T(a\mathbf{1}_n + s - \boldsymbol{X}^- w) + \gamma_2^T(b - a\mathbf{1}_m + \boldsymbol{X}^+ w) \\ & + \frac{\rho}{2}\|a\mathbf{1}_n + s - \boldsymbol{X}^- w\|^2 + \frac{\rho}{2}\|b - a\mathbf{1}_m + \boldsymbol{X}^+ w\|^2 \end{aligned}$$

where $\rho > 0$ is a penalty parameter. Then the ADMM algorithm will consist of the following steps. At every iteration:

Step 1. Update (w, a) according to

$$\begin{aligned} (w, a) \leftarrow \arg\min_{(w,a)} \quad & \frac{\lambda}{2}\|w\|^2 + \gamma_1^T(a\mathbf{1}_n - \boldsymbol{X}^- w) \\ & + \gamma_2^T(-a\mathbf{1}_m + \boldsymbol{X}^+ w) + \frac{\rho}{2}\|a\mathbf{1}_n + s - \boldsymbol{X}^- w\|^2 \\ & + \frac{\rho}{2}\|b - a\mathbf{1}_m + \boldsymbol{X}^+ w\|^2 \end{aligned}$$

which is a convex quadratic function with respect to (w, a). Let us define

$$A \overset{\text{def}}{=} \begin{bmatrix} \boldsymbol{X}^+ & -\mathbf{1}_m \\ -\boldsymbol{X}^- & \mathbf{1}_n \end{bmatrix} \in \mathbb{R}^{(n+m)\times(d+1)}.$$

Then the update of (w, a) will be

$$\begin{bmatrix} w \\ a \end{bmatrix} \leftarrow -\left(\rho A^T A + \begin{bmatrix} \lambda I_d & 0 \\ 0 & 0 \end{bmatrix}\right)^{-1} A^T \begin{bmatrix} \rho b + \gamma_2 \\ \rho s + \gamma_1 \end{bmatrix}$$

Step 2. Update (b, s) according to

$$\begin{aligned} (b, s) \leftarrow \arg\min_{(b,s), s\leq 0} & \frac{1}{m}\sum_{i=1}^{m}\ell(b_i) + \gamma_1^T s + \gamma_2^T b \\ & + \frac{\rho}{2}\|a\mathbf{1}_n + s - \boldsymbol{X}^- w\|^2 + \frac{\rho}{2}\|b - a\mathbf{1}_m - \boldsymbol{X}^+ w\|^2 \end{aligned}$$

The variable s is simply updated as follows

$$s \leftarrow \left[(\boldsymbol{X}^- w - a\mathbf{1}_n) - \frac{1}{\rho}\gamma_1\right]_-$$

where $[\cdot]_-$ stands for projection on the negative orthant. Depending on the choice of the loss function $\ell(\cdot)$, the update of b may vary. For the case of the truncated quadratic loss, the update has closed form. More specifically, for $i = 1, 2, \cdots, m$, let $c_i \overset{\text{def}}{=} a - (\boldsymbol{X}^+ w)_i - \frac{1}{\rho}(\gamma_2)_i$, then

$$b_i \leftarrow \begin{cases} c_i & \text{if } c_i \leq -1 \\ \frac{1}{\rho + \frac{2}{m}}\left(-\frac{2}{m} + \rho c_i\right) & \text{if } c_i > -1. \end{cases}$$

Step 3. Update (γ_1, γ_2) according to

$$\begin{aligned} \gamma_1 &\leftarrow \gamma_1 + \rho(a\mathbf{1}_n + s - \boldsymbol{X}^- w) \\ \gamma_2 &\leftarrow \gamma_2 + \rho(b - a\mathbf{1}_m + s + \boldsymbol{X}^+ w) \end{aligned}$$

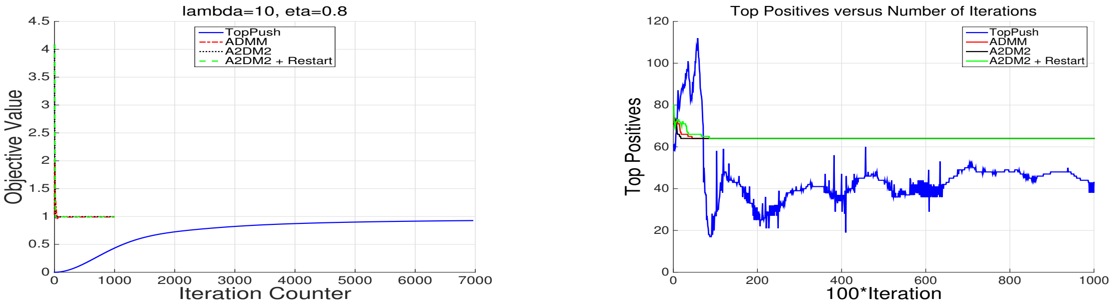
As shown by Theorem 1, A2DM2 requires strong convexity of the objective function with respect to both (w, a) and (b, s) pairs. In order to make this condition hold, we add extra quadratic terms to the ranking objective function in equation (12) and change it to

$$\frac{\lambda}{2}\|w\|^2 + \frac{1}{m}\sum_{i=1}^{m}\ell(b_i) + \frac{\lambda_1}{2}a^2 + \frac{\lambda_2}{2}\|s\|^2 + \frac{\lambda_2}{2}\|b\|^2 \tag{13}$$

where λ_1 and λ_2 are small positive constants. Although adding these terms will slightly change the problem, our experimental results show the ranking performance of the algorithm is not worse than the existing state-of-the-art approaches. We avoid deriving the iterative updates of A2DM2 here as they are quite similar to ADMM and instead summarize them in Algorithm 3.

3.4 Computational Complexity

Updating the pair (w, a) requires $\mathcal{O}(d(m+n))$ operations provided that the matrix $H \overset{\text{def}}{=} \left(\rho A^T A + \begin{bmatrix} \lambda I_d & 0 \\ 0 & 0 \end{bmatrix}\right)^{-1} A^T$ is computed before executing the algorithm and saved in memory. The computational cost of updating (b, s) and (γ_1, γ_2) is of the same order $\mathcal{O}(d(m+n))$. Therefore, in order to achieve an ϵ-accuracy solution, the ADMM and A2DM2 require $\mathcal{O}(d(m+n)/\epsilon)$ and $\mathcal{O}(d(m+n)/\sqrt{\epsilon})$ operations, respectively. Therefore, in terms of the computational complexity, A2DM2 is comparable with the state-of-the-art algorithm in [18].



(a) Objective value vs. number of iterations

(b) Top@Pos vs. number of iterations

Figure 2: Study on `spambase` dataset. (a) Objective value versus number of iterations (b) Top Ranking performance (Pos@Top) on the test set after every 100 iterations of the training phase. A2DM2 converges to its final ranking performance after few hundreds of iteration, whereas TopPush [18] does not seem to achieve a stable number of Pos@Top.

Algorithm 3: A2DM2 for Ranking

```
Require: $b = \hat{b}$, $s = \hat{s}$, $\gamma_1 = \hat{\gamma}_1$, $\gamma_2 = \hat{\gamma}_2$, $\lambda$, $\lambda_1$, $\lambda_2$, $\rho$, $t_0 = 1$

for $k = 0, 1, 2, 3, \cdots$ do
    $\begin{bmatrix} w \\ a \end{bmatrix} \leftarrow -\left(\rho A^T A + \begin{bmatrix} \lambda I_d & 0 \\ 0 & \lambda_1 \end{bmatrix}\right)^{-1} A^T \begin{bmatrix} \rho\hat{b} + \hat{\gamma}_2 \\ \rho\hat{s} + \hat{\gamma}_1 \end{bmatrix}$
    $s \leftarrow \left[\frac{\rho}{\rho+\lambda_2}(\boldsymbol{X}^{-} w - a\mathbf{1}_n) - \frac{1}{\rho+\lambda_2}\hat{\gamma}_1\right]_-$
    $c \leftarrow a\mathbf{1}_n - \boldsymbol{X}^{+} w - \frac{1}{\rho}\hat{\gamma}_2$
    $b_i \leftarrow \begin{cases} \frac{\rho}{\rho+\lambda_2} c_i & \text{if } c_i \leq -1 \\ \frac{1}{\rho+\frac{2}{m}+\lambda_2}\left(-\frac{2}{m} + \rho c_i\right) & \text{if } c_i > -1. \end{cases}$
    $\begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix} \leftarrow \begin{bmatrix} \hat{\gamma}_1 \\ \hat{\gamma}_2 \end{bmatrix} + \rho \begin{bmatrix} a\mathbf{1}_n + s - \boldsymbol{X}^{-} w \\ b - a\mathbf{1}_m + s + \boldsymbol{X}^{+} w \end{bmatrix}$
    $t_0 \leftarrow t$, $t \leftarrow \frac{1+\sqrt{1+4t_0^2}}{2}$
    $\begin{bmatrix} \hat{\gamma}_1 \\ \hat{\gamma}_2 \end{bmatrix} \leftarrow \begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix} + \frac{t_0-1}{t} \begin{bmatrix} \gamma_1 - \gamma_1^0 \\ \gamma_2 - \gamma_2^0 \end{bmatrix}$
    $\hat{s} \leftarrow \frac{1}{\lambda_2}\left[-\hat{\gamma}_1\right]_-$
    $b_i \leftarrow \begin{cases} \frac{-(\hat{\gamma}_2)_i}{\lambda_2} & \text{if } \frac{-(\hat{\gamma}_2)_i}{\lambda_2} \leq -1 \\ \frac{(\hat{\gamma}_2)_i + \frac{2}{m}}{\lambda_2 + \frac{2}{m}} & \text{otherwise.} \end{cases}$
end for
```

4. A2DM2 FOR DIRTY MODELS

In this section we demonstrate the effectiveness of our method for solving an extensively studied family of statistical problems that are often known as "Dirty Models" in the machine learning literature [27, 8].

Dirty Models. In many high dimensional statistical problems, the number of observations is far less than the dimension of the model to be estimated. Without any prior knowledge, the true model is not identifiable. Fortunately, in many practical applications, the model is known to have a low dimensional structure that can be used to resolve the identifiability issue. This prior knowledge can be exploited by adding to the objective function some appropriate convex regularizers which capture the structure of the model. Formally speaking, assume that given the linear observations $b = Ax + By$, where A, B are known matrices, the goal is to estimate x and y. In addition, let us assume that $R_1(\cdot)$ and $R_2(\cdot)$ are the convex penalty functions which encode the prior knowledge of x and y, respectively. Then, the estimation problem can be formulated as follows

$$\min_{x,e} \; R_1(x) + R_2(y) \quad \text{subject to} \quad Ax + By = b. \tag{14}$$

Many famous formulations can be easily interpreted using this model. For example, by specializing $R_2(y) = \frac{1}{2}\|y\|^2$ (the ℓ_2-norm squared), $R_1(x) = \|x\|_1$ (the ℓ_1-norm), A as the design matrix and B as the identity matrix, we obtain the Lasso formulation [24]. When working with real-world data, in order to increase the robustness of the estimation procedure, the elastic net regularizer, given by $R_1(x) = \|x\|_1 + \frac{1}{2}\|x\|^2$, can be used instead of the ℓ_1-norm penalty [28].

Dirty Model with Elastic Net Regularizer. Another interesting special-case of problem (14) is where R_1 and R_2 are both elastic net regularization functions. Given the linear observation model $b = Ax + y$, such a problem can be written as

$$\min_{x,y} F(x,y) \overset{\text{def}}{=} \|x\|_1 + \frac{1}{2}\|x\|^2 + \mu(\|y\|_1 + \frac{1}{2}\|y\|^2) \tag{15}$$
$$\text{subject to} \;\; Ax + y = b.$$

where $\mu > 0$ is a constant. Minimizing the ℓ_1-norm of the variables here is to exploit the prior knowledge about the sparsity of such unknowns [26, 16]. This problem has the same form as (1) with $f_1(x) = \|x\|_1 + \frac{1}{2}\|x\|^2$ and $f_2(y) = \mu(\|y\|_1 + \frac{1}{2}\|y\|^2)$. Note that the objective function components f_1 and f_2 are strongly convex in terms of x and y, respectively. Therefore, our analysis guarantees the fast convergence of A2DM2 for this problem. The application of A2DM2 to this problem is quite straight-forward. However, we omit the details of the variable updates here due to space limits. In the next section,we describe the results of our numerical tests with this problem.

Numerical Experiments. To test the performance of our accelerated method compared to ADMM, we carry out a set of simulations. We randomly generate the observation matrix $A \in \mathbb{R}^{m\times n}$ of size $m = 2^8$ and $n = 2^9$ from a standard

Gaussian distribution. The true target vector x is sparse with only five percent of its entries being non-zero. The non-zero entries are standard Gaussian. The error vector $e \in \mathbb{R}^m$ is generated from an exponential distribution with average 0.01. Figure (1) shows the convergence behavior of ADMM, A2DM2, and A2DM2 + Restart for solving problem (15). Figure 1(a) plots the primal residual sequence $r_k = \|Ax_k - b - y_k\|$, Figure 1(b) shows the primal objective optimality gap $F(x_k, y_k) - F(x^*, y^*)$ and Figure 1(c) shows the dual objective optimality gap $D(\lambda^*) - D(\lambda_k)$. Both of the objective sequences are normalized with their initial values, i.e. with their values at iteration $k = 1$. As the three figures suggest, A2DM2 performs better than ADMM in terms of all three measures. The best performance is observed for A2DM2+Restart. Note that the improved performance is obtained at the cost of defining the auxiliary primal and dual variables $\hat{y}$ and $\hat{\lambda}$ whose updates can be done in $\mathcal{O}(m)$ (the details are omitted for the sake of brevity).

5. EXPERIMENTS ON TOP RANKING

5.1 Settings

Datasets. To evaluate the performance of the proposed A2DM2 algorithm on the problem of learning to rank, we conduct a set of experiments on various datasets. The left column of Table 1 summarizes the datasets used in our experiments. All datasets used are publicly available binary classification datasets[2] having varying sizes and coming from different domains.

Some datasets come from the medical domain (`breast-cancer, diabetes`), ecology (`covtype`), biology (`cod-rna, splice`), others from email spam filtering (`spambase`), web data (`w8a`), census data (`a9a`), and credit card approval (`australian`). Also, competition data on generalization ability and text decoding (`ijcnn1`) were used. The `epsilon` dataset is an artificial data set from the Pascal large scale learning challenge 2008.

Setup & Parameters. On each dataset, we run experiments for ten trials and report the averaged results over those trials. In each run, the dataset is randomly divided into two subsets: 2/3 for training and 1/3 for test. For all algorithms, we set the precision parameter ϵ to 10^{-4}, choose other parameters by 3-fold cross validation (based on the average value of Pos@Top) on training set, and perform the evaluation on the test set. In particular, the regularization parameter λ is chosen from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}\}$ based on cross-validation on the TopPush algorithm. The parameters λ_1, λ_2 were set to the value 0.01. The step size ρ of the proximal operator in the ADMM-based algorithms was cross validated as followed: For ADMM ρ was chosen from the set $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$, for A2DM2 from $\{10^{-4}, 10^{-3}, 10^{-2}\}$ and for A2DM2+Restart from $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$.

Methods. We compared the standard ADMM, the proposed A2DM2 and A2DM2+Restart frameworks with TopPush [18], which is the state-of-the-art top rank algorithm. We implemented the ADMM-based algorithms in MATLAB and used the publicly available source code for TopPush.

Top-Ranking Metric. Since the objective of TopPush, and therefore of all compared approaches, is to push down the top ranked negative example in the ranked list, a natural performance measure is the number of positives ranked on top of the first negative example (Pos@Top) [1]. Larger values in this metric imply better top ranking performance.

Training Efficiency. In order to evaluate the computational efficiency of our proposed algorithms, we also report the average time (in seconds) it takes for the algorithms to be trained. For this experiment, we set the parameters of the different algorithms to be the best ones selected by cross validation and we run them on the training set. We do so for the ten different random runs and average out the training time. Regarding the stopping criterion, all three algorithms i.e., ADMM, A2DM2 and A2DM2+Restart, are stopped when the iteration number is greater than 10 and the sum of the primal and dual residuals is less than ϵ. The TopPush stopping criterion is kept in its original form as given in the source code[3], i.e., when the iteration number is greater than 10 and the relative dual objective gap of the TopPush algorithm is less than ϵ.

5.2 Results: Running Time Comparison

In the right-most column of Table 1 we report the training performance (in seconds) of the algorithms A2DM2, A2DM2 + Restart, compared to the TopPush algorithm and the standard ADMM. One can observe that in most datasets A2DM2 matches the training time of TopPush, and it can be even one order of magnitude faster (`spambase`). In the cases where A2DM2 is slower than TopPush, A2DM2 achieves better ranking performance (`diabetes, a9a`). In general, there is a tradeoff between accuracy at the top and time for convergence. A2DM2 usually manages to balance the tradeoff and achieves good Pos@Top at time comparable (or better) with TopPush, while ADMM often does not.

5.3 Results: Top Ranking Accuracy

In addition, in Table 1 we report the performance of the compared approaches in terms of the average Pos@Top. The A2DM2 algorithm almost always matches the ranking performance of TopPush and in most datasets it results in slightly better results.

From the results of Table 1, we observe that as the size of the datasets increases at the bottom of the table, the acceleration that A2DM2 provides compared to ADMM becomes more considerable. For instance, for the cod-rna dataset, the value of Pos@Top for A2DM2 is around twice that for ADMM and yet A2DM2 is, in average, two orders of magnitude (100 times) faster than ADMM. Also, the results of ADMM for the two larger datasets, epsilon and covtype, are missing from Table 1 since the cross-validation study for ADMM was time consuming.

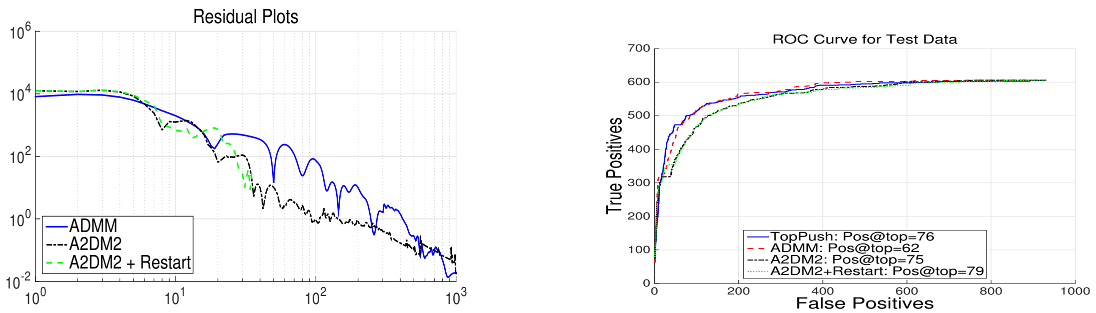
For the `spambase` dataset, for a single random training-test split, Figure 3 (b) shows the Receiver Operating Characteristic (ROC) curves of the four compared algorithms. Since we focus on the ranking model where accuracy at the top is critical, good performance in the left-most part of the ROC curve is necessary. In this regard, one can see similar ranking performance of the compared approaches. In detail, A2DM2 + Restart achieves slightly higher Pos@Top

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[3] http://lamda.nju.edu.cn/code_TopPush.ashx

Data	**Algorithm**	**Pos@top**	**Time (sec.)**
`breast-cancer`	TopPush	$4.90\times10^{1} \pm 1.39\times10^{1}$	$5.58\times10^{-1} \pm 8.01\times10^{-1}$★
239 / 444	ADMM	$5.33\times10^{1} \pm 1.38\times10^{1}$	$1.74\times10^{0} \pm 1.46\times10^{0}$
d:10	A2DM2	$5.25\times10^{1} \pm 1.48\times10^{1}$	$9.24\times10^{-1} \pm 1.08\times10^{0}$★
	A2DM2+Restart	$5.23\times10^{1} \pm 1.50\times10^{1}$	$3.15\times10^{0} \pm 3.53\times10^{0}$
`australian`	TopPush	$1.13\times10^{1} \pm 5.14\times10^{0}$	$2.22\times10^{-1} \pm 2.00\times10^{-1}$★
307 / 383	ADMM	$1.62\times10^{1} \pm 7.18\times10^{0}$	$5.91\times10^{-1} \pm 4.79\times10^{-1}$★
d:14	A2DM2	$1.77\times10^{1} \pm 1.08\times10^{1}$	$5.12\times10^{-1} \pm 5.54\times10^{-1}$★
	A2DM2+Restart	$1.71\times10^{1} \pm 6.76\times10^{0}$	$1.34\times10^{0} \pm 2.05\times10^{0}$
`diabetes`	TopPush	$1.41\times10^{1} \pm 2.14\times10^{1}$	$4.66\times10^{-2} \pm 9.08\times10^{-2}$★
500 / 268	ADMM	$2.36\times10^{1} \pm 2.03\times10^{1}$	$4.58\times10^{-1} \pm 2.51\times10^{-1}$
d:34	A2DM2	$3.19\times10^{1} \pm 1.73\times10^{1}$	$5.03\times10^{-1} \pm 2.41\times10^{-1}$
	A2DM2+Restart	$1.87\times10^{1} \pm 1.89\times10^{1}$	$6.17\times10^{-1} \pm 4.60\times10^{-1}$
`spambase`	TopPush	$5.49\times10^{1} \pm 8.29\times10^{1}$	$1.63\times10^{1} \pm 9.74\times10^{0}$
1,813 / 2,788	ADMM	$4.48\times10^{1} \pm 3.86\times10^{1}$	$2.06\times10^{1} \pm 1.59\times10^{1}$
d:57	A2DM2	$5.02\times10^{1} \pm 3.64\times10^{1}$	$3.35\times10^{0} \pm 1.23\times10^{0}$★
	A2DM2+Restart	$5.48\times10^{1} \pm 3.55\times10^{1}$	$1.51\times10^{1} \pm 8.07\times10^{0}$
`splice`	TopPush	$8.78\times10^{1} \pm 4.85\times10^{1}$	$1.86\times10^{0} \pm 2.58\times10^{0}$★
1,648 / 1,527	ADMM	$9.99\times10^{1} \pm 2.22\times10^{1}$	$7.29\times10^{0} \pm 5.02\times10^{0}$★
d:60	A2DM2	$1.15\times10^{2} \pm 2.63\times10^{1}$●	$3.18\times10^{0} \pm 9.78\times10^{-1}$★
	A2DM2+Restart	$1.14\times10^{2} \pm 2.70\times10^{1}$●	$1.43\times10^{1} \pm 2.13\times10^{0}$
`ijcnn1`	TopPush	$6.08\times10^{1} \pm 2.06\times10^{1}$	$7.39\times10^{0} \pm 1.26\times10^{1}$★
4,853 / 45,137	ADMM	$1.24\times10^{2} \pm 3.76\times10^{1}$●	$1.82\times10^{2} \pm 6.77\times10^{1}$
d:22	A2DM2	$5.56\times10^{1} \pm 9.90\times10^{0}$	$1.50\times10^{0} \pm 2.32\times10^{0}$★
	A2DM2+Restart	$7.34\times10^{1} \pm 3.20\times10^{1}$	$1.13\times10^{2} \pm 1.39\times10^{2}$
`a9a`	TopPush	$1.78\times10^{1} \pm 1.30\times10^{1}$	$8.50\times10^{-1} \pm 1.85\times10^{-1}$★
11,687 / 37,155	ADMM	$4.57\times10^{1} \pm 2.19\times10^{1}$	$1.36\times10^{1} \pm 8.59\times10^{0}$
d:122	A2DM2	$5.47\times10^{1} \pm 2.89\times10^{1}$	$1.44\times10^{1} \pm 6.74\times10^{0}$
	A2DM2+Restart	$5.07\times10^{1} \pm 2.43\times10^{1}$	$5.29\times10^{1} \pm 3.43\times10^{1}$
`w8a`	TopPush	$1.39\times10^{2} \pm 3.42\times10^{1}$	$2.20\times10^{1} \pm 1.33\times10^{1}$★
1,933 / 62,767	ADMM	$1.37\times10^{2} \pm 4.19\times10^{1}$	$1.71\times10^{2} \pm 6.28\times10^{1}$
d:300	A2DM2	$1.38 \times 10^{2} \pm 4.98\times10^{1}$	$5.98\times10^{1} \pm 2.58\times10^{1}$★
	A2DM2+Restart	$1.44\times10^{2} \pm 3.86\times10^{1}$	$2.25\times10^{2} \pm 7.54\times10^{1}$
`covtype`	TopPush	$7.97\times10^{2} \pm 1.52\times10^{2}$●	$1.78 \times10^{1} \pm 4.26 \times 10^{0}$
283,301 / 297,711, d: 54	A2DM2	$2.63 \times 10^{1} \pm 2.48 \times 10^{1}$	$2.02 \times 10^{1} \pm 2.81 \times 10^{0}$
`cod-rna`	TopPush	$1.97\times10^{2} \pm 9.95\times10^{1}$	$8.34 \times10^{2} \pm 4.59 \times 10^{2}$
162,855 / 325,710	ADMM	$1.03\times10^{2} \pm 1.36\times10^{2}$	$6.50\times10^{2} \pm 7.33 \times 10^{2}$
d:8	A2DM2	$2.24 \times 10^{2} \pm 8.91 \times 10^{1}$	$2.01 \times 10^{0} \pm 5.52 \times 10^{-1}$★
	A2DM2+Restart	$1.27 \times 10^{2} \pm 8.38 \times 10^{1}$	$1.71 \times 10^{2} \pm 4.75 \times 10^{2}$
`epsilon`	TopPush	$1.82 \times 10^{3} \pm 3.07 \times 10^{2}$	$5.78 \times 10^{2} \pm 1.85 \times 10^{2}$
249,778 / 250,222	A2DM2	$2.07 \times 10^{3} \pm 4.16 \times 10^{2}$	$4.16 \times 10^{2} \pm 1.79 \times 10^{1}$
d:2,000			

Table 1: Data statistics (left column) and experimental results. The mean and standard deviation of the training time (sec) and the Pos@Top over ten random splits of training-test sets are reported. For each dataset, the number of positive and negative instances is below the data name as m/n, together with dimensionality d. For training time comparison, one or more algorithms are marked as ★ if they are at least an order of magnitude faster compared to the remaining ones. For top-ranking performance (Pos@Top) comparison, the entries marked with ● are those for which the number of positives at top is at least 10 times greater than the Pos@Top achieved by the rest of the algorithms. In most datasets, one can observe that A2DM2 is very competitive with TopPush in terms of the order of magnitude for both top-ranking accuracy and training time.



(a) Residual Plots

(b) ROC curve

Figure 3: Study on `spambase` dataset. (a) Residuals decay faster for the accelerated variants of ADMM compared to ADMM. (b) ROC Curve for test data: One can observe similar top ranking performance for the four approaches.

performance followed by TopPush and A2DM2 and finally ADMM.

5.4 Effect of number of training iterations

In this subsection, we study the ranking performance of A2DM2 versus the number of iterations through some figures and compare it with TopPush, A2DM2 + Restart and ADMM. The shown plots are just for one random training-test split of the `spambase` dataset. However, we observed same trends as what follows with the other datasets and with different training-test splits.

Fixing the regularization parameter $\lambda = 10$ and $\eta = 0.8$, Figure 2 (a) shows how the value of the objective evolves as the number of iterations grows. One can observe that ADMM, A2DM2 and A2DM2+Restart converge in only a few hundred iterations. In contrast, TopPush needs to run for a few thousands of iterations to reach the optimal objective value. In Figure 2 (b), we present how the number of Pos@Top in the test set evolves after every 100 iterations of the training phase. One interesting observation is that A2DM2 converges to its final test top-ranking performance after few hundreds of iteration, whereas TopPush does not seem to achieve a stable number of Pos@Top. Figure 3 (a) shows how the sum of the primal and dual residuals behaves vs. the number of training iterations. As the figure implies, the residuals decay faster for the accelerated variants of ADMM.

6. CONCLUSIONS

In this paper, we propose an Accelerated Alternating Direction Method of Multipliers, named A2DM2. We prove that it has $O(1/k^2)$ convergence rate when the objective terms are strongly convex. This guarantees a faster convergence rate compared to ADMM [4]. A large number of real world machine learning problems formulated under the ADMM framework can benefit from this improvement. We illustrate the applicability of A2DM2 on the problem of learning to rank, and show that it is competitive with the state-of-the-art TopPush algorithm [18] both in terms of ranking accuracy at the top and training efficiency.

7. ACKNOWLEDGEMENTS

This work was supported in part by NSF (IIS-1447566, IIS-1422557,CCF-1451986, CNS-1314560, IIS-0953274, IIS-1029711), NASA (NNX12AQ39A), and a gift from IBM and Yahoo!.

8. REFERENCES

[1] S. Agarwal. The infinite push: A new support vector ranking algorithm that directly optimizes accuracy at the absolute top of the list. In *SDM*, 839–850, 2011.

[2] A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[3] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[5] S. Boyd, C. Cortes, M. Mohri, and A. Radovanovic. Accuracy at the top. In *NIPS*, 953–961, 2012.

[6] C. JC Burges. From RankNet to LambdaRank to LambdaMART: An overview. In *Learning*, 11: 23–581, 2010.

[7] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Yahoo! Learning to Rank Challenge*, 1–24, 2011.9

[8] I. Dhillon and J. Ghosh. Dirty statistical models.

[9] B. O'Donoghue and E. Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics,* 15.3:715-732, 2012.

[10] J. Duchi and Y. Singer. Efficient online and batch learning using forward backward splitting. *The Journal of Machine Learning Research*, 10:2899–2934, 2009.

[11] W. Fenchel. On conjugate convex functions. *Canad. J. Math*, 1(73-77), 1949.

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

[13] D. Goldfarb, S. Ma, and K. Scheinberg. Fast alternating linearization methods for minimizing the

sum of two convex functions. *Mathematical Programming*, 141(1-2):349–382, 2013.

[14] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3), 1588-1623.

[15] A. Juditsky and A. Nemirovski. First order methods for nonsmooth convex large-scale optimization, i: general purpose methods. *Optimization for Machine Learning*, pages 121–148, 2010.

[16] S. P. Kasiviswanathan, H. Wang, A. Banerjee, and P. Melville. Online l1-dictionary learning with application to novel document detection. In *NIPS*, pages 2267–2275, 2012.

[17] Q. Le, and A. Smola. Direct optimization of ranking measures. In *arXiv preprint arXiv:0704.3359*, 2007.

[18] N. Li, R. Jin, and Z-H Zhou. Top rank optimization in linear time. In *NIPS*, 2014.

[19] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3): 225–331, 2009.

[20] Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[21] F. A. Potra and S. J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1):281–302, 2000.

[22] A. Rakotomamonjy. Sparse support vector infinite push. *arXiv preprint arXiv:1206.6432*, 2012.

[23] C. Rudin. The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *JMLR*, 10: 2233–2271, 2009.

[24] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[25] H. Wang and A. Banerjee. Online alternating direction method. *arXiv preprint arXiv:1206.6448*, 2012.

[26] J. Wright and Y. Ma. Dense error correction via l1-minimization. *Information Theory, IEEE Transactions on*, 56(7):3540–3560, 2010.

[27] E. Yang and P. Ravikumar. Dirty statistical models. In *Advances in Neural Information Processing Systems*, pages 611–619, 2013.

[28] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

APPENDIX

Here we prove Lemma 2. This lemma is the core in the analysis of accelerated ADMM algorithm. It is used in proof of Lemma 1 (for details see [14]). Our proof is similar to the proof presented for a similar lemma in [14].

Proof. In order to facilitate the proof let us define $\lambda_k^{1/2} = \hat{\lambda}_k + \tau(c - Ax_k - By_k)$. The optimality condition of update of x_k in accelerated ADMM algorithm (see Algorithm 2) gives

$$\begin{aligned} &\nabla f_1(x_k) - A^T\hat{\lambda}_k - \tau A^T(c - Ax_k - B\hat{y}_k) = 0 \\ \Rightarrow\ &\nabla f_1(x_k) - A^T\lambda_k^{1/2} = 0 \Rightarrow\ x_k = \nabla f_1^*(A^T\lambda_k^{1/2}), \end{aligned} \tag{16}$$

where the last equality is due to the definition of dual functions and strong convexity of f_1. Using the same argument, it is easy to see that $y_k = \nabla f_2^*(B^T\lambda_k)$.

As we mentioned earlier, when the function f_1 is strongly convex, its dual be smooth with Lipschitz gradient. Therefore, for any γ

$$\begin{aligned} &f_1^*(A^T\gamma) - f_1^*(A^T\lambda_{k+1}) = \Big(f_1^*(A^T\gamma) - f_1^*(A^T\lambda_{k+1}^{1/2})\Big) \\ &+ \Big((f_1^*(A^T\lambda_{k+1}^{1/2}) - f_1^*(A^T\lambda_{k+1}))\Big) \\ &\geq\ \Big(f_1^*(A^T\lambda_{k+1}^{1/2}) + \langle A\nabla f_1^*(A^T\lambda_{k+1}^{1/2}), \gamma - \lambda_{k+1}^{1/2}\rangle - f_1^*(A^T\lambda_{k+1}^{1/2})\Big) \\ &- \Big(\langle \lambda_{k+1} - \lambda_{k+1}^{1/2}, A\nabla f_1^*(A^T\lambda_{k+1}^{1/2})\rangle + \frac{\rho^2(A)}{2\sigma_1}\|\lambda_{k+1} - \lambda_{k+1}^{1/2}\|^2\Big) \\ &= \langle \gamma - \lambda_{k+1}, A\nabla f_1^*(A^T\lambda_{k+1}^{1/2})\rangle - \frac{\rho^2(A)}{2\sigma_1}\|\lambda_{k+1} - \lambda_{k+1}^{1/2}\|^2. \end{aligned} \tag{17}$$

Our goal is to bound the term $\|\lambda_{k+1} - \lambda_{k+1}^{1/2}\|$ in (17) using $\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|$. Based on the updates of accelerated ADMM algorithm, it is clear that $\lambda_{k+1} - \lambda_{k+1}^{1/2} = -\tau B(y_{k+1} - \hat{y}_{k+1})$. Thus,

$$\lambda_{k+1} - \lambda_{k+1}^{1/2} = -\tau B(\nabla f_2^*(B^T\lambda_{k+1}) - \nabla f_2^*(B^T\hat{\lambda}_{k+1})),$$

where the equality is due to the optimality conditions of the updates of y_{k+1} and $\hat{y}_{k+1}$. Now we use the Lipschitz continuity of the ∇f_2^* to get

$$\|\lambda_{k+1} - \lambda_{k+1}^{1/2}\| \leq \tau\frac{\rho^2(B)}{\sigma_2}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|. \tag{18}$$

Combining (17) and (18) and using the fact that $\tau \leq \frac{\sigma_1\sigma_2^2}{\rho^2(A)\rho^4(B)}$

$$\begin{aligned} &f_1^*(A^T\gamma) - f_1^*(A^T\lambda_{k+1}) \\ &\geq \langle \gamma - \lambda_{k+1}, A\nabla f_1^*(A^T\lambda_{k+1}^{1/2})\rangle - \frac{1}{2\tau}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|. \end{aligned} \tag{19}$$

Using the convexity of f_2^*, it is clear that

$$f_2^*(B^T\gamma) - f_2^*(B^T\lambda_{k+1}) \geq \langle \gamma - \lambda_{k+1}, B\nabla f_2^*(B^T\lambda_{k+1})\rangle. \tag{20}$$

Combining (19) and (20), we can easily get

$$\begin{aligned} &D(\lambda_{k+1}) - D(\gamma) \geq \langle \gamma - \lambda_{k+1}, A\nabla f_1^*(A^T\lambda_k^{1/2}) \\ &+ B\nabla f_2^*(B^T\lambda_{k+1}) - c\rangle - \frac{1}{2\tau}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|. \end{aligned} \tag{21}$$

From the optimality conditions of the updates of accelerated ADMM it is clear that $x_{k+1} = \nabla f_1^*(A^T\lambda_k^{1/2})$ and $y_{k+1} = \nabla f_2^*(B^T\lambda_{k+1})$. Replacing these in (21) and noting $\tau(\hat{\lambda}_{k+1} - \lambda_{k+1}) = Ax_{k+1} + By_{k+1} - c$ yields

$$\begin{aligned} &D(\lambda_{k+1}) - D(\gamma) \\ &\geq\ \frac{1}{\tau}\langle \gamma - \lambda_{k+1}, \hat{\lambda}_{k+1} - \lambda_{k+1}\rangle - \frac{1}{2\tau}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2 \\ &= \frac{1}{\tau}\langle \gamma - \hat{\lambda}_{k+1} + \hat{\lambda}_{k+1} - \lambda_{k+1}, \hat{\lambda}_{k+1} - \lambda_{k+1}\rangle - \frac{1}{2\tau}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2 \\ &= \frac{1}{\tau}\langle \gamma - \hat{\lambda}_{k+1}, \hat{\lambda}_{k+1} - \lambda_{k+1}\rangle + \frac{1}{2\tau}\|\lambda_{k+1} - \hat{\lambda}_{k+1}\|^2, \end{aligned}$$

which is the desired result. □