ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, IL 60439

A First-Order Smoothing Technique for a Class of Large-Scale Linear Programs

Jieqiu Chen and Samuel Burer

Mathematics and Computer Science Division

Preprint ANL/MCS-P1971-1011

November 7, 2011

A First-Order Smoothing Technique for a Class of Large-Scale Linear Programs*

Jieqiu Chen† Samuel Burer‡

November 7, 2011

Abstract

We study a class of linear programming (LP) problems motivated by large-scale machine learning applications. After reformulating the LP as a convex nonsmooth problem, we apply Nesterov's primal-dual smoothing technique. It turns out that the iteration complexity of the smoothing technique depends on a parameter θ that arises because we need to bound the originally unbounded primal feasible set. We design a strategy that dynamically updates θ to speed up the convergence. The application of our algorithm to two machine learning problems demonstrates several advantages of the smoothing technique over existing methods.

Keywords: smoothing technique, large-scale linear programming, nonsmooth optimization, machine learning

*The research of both authors was supported in part by NSF Grant CCF-0545514. J. Chen was supported in part by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA. Email: jieqchen@mcs.anl.gov

‡Department of Management Sciences, University of Iowa, Iowa City, IA 52242-1994, USA. Email: samuel-burer@uiowa.edu

1 Introduction

We investigate a first-order smoothing technique to solve large-scale instances of the following linear programming (LP) problem:

$$\begin{aligned} \min_{\alpha,\xi} \quad & c^T\alpha + w^T\xi \\ \text{s. t.} \quad & A\alpha - b \leq \xi \\ & \alpha,\ \xi \geq 0, \end{aligned} \tag{P}$$

where $\alpha \in \Re^n$ and $\xi \in \Re^m$ are the decision vectors and $A \in \Re^{m\times n}$, $b \in \Re^m$, $c \in \Re^n_+$ and $w \in \Re^m_+$ are the data. The optimal value of (P) is bounded below by zero, and thus an optimal solution exists. The variable ξ may be interpreted as an error, allowing some of the constraints "$A\alpha \leq b$" to be violated, and the objective term $w^T\xi$ then serves to penalize such violations. Expressed differently, if ξ were fixed to 0, then the above formulation would be similar to a standard-form LP with $c \geq 0$.

Problem (P) is motivated by several machine learning problems. One such example is the LP-based ranking algorithm (Ataman et al., 2006; Ataman, 2007). Another example is the 1-norm support vector machine (Zhu et al., 2003; Mangasarian, 2006). In Section 5, we will demonstrate how to cast such machine learning problems as (P). We are also particularly interested in large instances of (P) for two reasons: (i) while small instances can be solved efficiently by current LP solvers, for problems where A is large and dense, using simplex or interior-point methods might not be feasible because of memory limits; and (ii) the machine learning problems above are often applied to large datasets, making A large. In applications that involve kernel matrices, such as the popular RBF kernels (Hsu et al., 2003), A is also usually completely dense.

Several approaches can be used to solve (P). Standard approaches are the simplex method or interior-point methods. For applications in machine learning, however, (P) is often too large. For example, Ataman (2007) reported that a moderately sized ranking problem formulated as (P) caused CPLEX to run out of memory on a standard PC. Mangasarian (2006) formulated a class of LPs, which includes (P) as a special case; he posed the problem as the unconstrained minimization of a convex differentiable piecewise-quadratic objective function and solved it using a generalized Newton method. As we will see in Section 5, however, this method may not be sufficiently robust in some cases.

Another approach is to treat (P) as the equivalent nonsmooth problem

$$\begin{aligned} \min \quad & c^T\alpha + w^T\,(A\alpha - b)^+ \\ \text{s.t.} \quad & \alpha \geq 0, \end{aligned} \tag{NS$_0$}$$

where $(A\alpha - b)^+$ denotes the nonnegative part of the vector $A\alpha - b$, and then to solve (NS$_0$) using nonsmooth techniques such as the standard subgradient method. Compared with the simplex method and interior-point methods, the subgradient method requires much less memory (basically the memory to store A). It also has low computational costs at each iteration (basically A times a vector). The main drawback of the subgradient method is slow convergence: its worst-case iteration complexity is $O(1/\epsilon^2)$ (Nesterov, 2004), where ϵ is the error tolerance of a calculated solution.

In this paper, we use Nesterov's first-order smoothing method (Nesterov, 2005a,b) to solve (NS$_0$). For bounded feasible sets, the smoothing method has worst-case iteration complexity $O(1/\epsilon)$, which is an order of magnitude faster than the subgradient method. At the same time, its computational cost per iteration and memory requirements are comparable to the subgradient method.

Researchers have successfully applied the smoothing method to many large-scale problems. Banerjee et al. (2006) considered the problem of fitting a large-scale covariance matrix to multivariate Gaussian data, which they solved using the smoothing method. Hoda et al. (2007) and Gilpin et al. (2007) applied the smoothing technique to approximate Nash equilibria of large sequential two-player zero-sum games. Becker et al. (2009) demonstrate that the smoothing method is ideally suited for solving large-scale compressed sensing reconstruction problems. Smoothing techniques also have applications in semidefinite programming (Nesterov, 2007; d'Aspremont, 2008) and general convex optimization (Lan et al., 2009).

Nesterov's first-order smoothing method applies to the following generic problem:

$$\min\{f(x) \;:\; x \in Q\}, \tag{1}$$

where f is a continuous convex function with a certain structure and Q is a compact convex set (see Section 2). To apply the smoothing method to (NS$_0$), we will need the following assumption.

Assumption 1.0.1. *Define $\mathcal{B} := \{i \in \{1,\ldots,n\} : c_i = 0\}$. We assume $\alpha_{\mathcal{B}}$ is bounded in (P). Specifically, we assume knowledge of $h \in \mathbb{R}^{|\mathcal{B}|}_{++}$ such that $\alpha_{\mathcal{B}} \leq h$ is valid for at least one optimal solution of (P).*

This assumption allows us to bound the primal feasible set as follows. First, the assumption

just mentioned bounds the subvector $\alpha_{\mathcal{B}}$. Next, the remaining components of α are naturally bounded by any fixed upper bound on the optimal value of (NS_0). In particular, let θ^* be the optimal value of (NS_0) and $\theta > 0$ be any upper bound on θ^*. We have

$$c^T\alpha^* \leq c^T\alpha^* + w^T(A\alpha^* - b)^+ = \theta^* \leq \theta,$$

and so $c^T\alpha \leq \theta$ is valid for the optimal set of (NS_0). Therefore, (NS_0) is equivalent to the following problem over a compact convex set:

$$\begin{aligned} \min_{\alpha \geq 0} \quad & c^T\alpha + w^T\left(A\alpha - b\right)^+ \\ \text{s.t.} \quad & c^T\alpha \leq \theta,\ \alpha_{\mathcal{B}} \leq h. \end{aligned} \tag{NS}$$

Based on (NS_0) and (NS), we prove that the iteration complexity of Nesterov's smoothing method is a function of θ, and we show that the smoothing algorithm can be improved by dynamically updating θ as it generates better bounds on θ^*. To the best of our knowledge, this is the first application of the smoothing technique to solve problems such as (P) with unbounded feasible sets.

Independently, Zhou et al. (2010) considered applying Nesterov's smoothing technique to several SVM problems, including the 1-norm SVM that we will study in Section 5. However, it is not clear to us how they circumvented the unboundedness issue of the primal feasible set since they provide no discussion of this issue. Moreover, they use Nesterov's primal-only smoothing method (Nesterov, 2005a) whereas our algorithm is based on the primal-dual excessive gap technique (Nesterov, 2005b). Our treatment of specific ingredients of the method is also different, as will be shown in Section 3.

This paper is organized as follows. We summarize the major ingredients of the smoothing technique in Section 2 to facilitate later discussion. In Section 3, we show how to convert (P) into a problem of the form (1), and we specify the major components of the smoothing technique, such as the choice of the so-called prox-functions and derivations of various parameters. In Section 4, we design a strategy to dynamically update the upper bound θ and show how it speeds up the smoothing method. In Section 5, we present two machine learning applications of the smoothing technique. In particular, we compare the smoothing technique with two existing methods, respectively, and demonstrate that the smoothing technique has several advantages.

2 Nesterov's Smoothing Technique

In this section, we review the major ingredients of Nesterov's smoothing technique: the excessive gap condition and convergence rate. We focus on the concepts and results that will be used in our study and leave out technical details. First, we introduce some notation and definitions that will be used throughout the paper.

2.1 Notation and terminology

Let E denote a finite-dimensional real vector space, possibly with an index. This space is equipped with a norm $\|\cdot\|$, which has the same index as the corresponding space. Let $\hat{A}$ be a linear operator from E_1 to E_2, that is, $\hat{A} : E_1 \to E_2$. Define the operator norm of $\hat{A}$, induced by the norms $\|\cdot\|_1$ and $\|\cdot\|_2$, as

$$\|\hat{A}\|_{1,2} = \max_{\|x\|_1=1} \max_{\|u\|_2=1} \langle \hat{A}x, u\rangle,$$

where $\langle\cdot,\cdot\rangle$ refers to the regular inner product. Note $\|\cdot\|_1$ and $\|\cdot\|_2$ do not necessarily represent the standard l_1-norm and l_2-norm; the subscripts are indices only. The operator norm has the following property:

$$\|\hat{A}\|_{1,2} = \max_{\|x\|_1=1} \|\hat{A}x\|_2^* = \max_{\|u\|_2=1} \|\hat{A}^T u\|_1^*,$$

where $\|\cdot\|^*$ denotes the dual norm associated with $\|\cdot\|$ and is defined as

$$\|z\|^* := \max\{\langle z, x\rangle : \|x\| \le 1\}.$$

We use $\hat{A}_i$ to denote the ith row of $\hat{A}$ and $\hat{A}^j$ the jth column of $\hat{A}$. Similarly, we use $\hat{A}_{\mathcal{I}}$ ($\hat{A}^{\mathcal{J}}$) to denote the rows (columns) of $\hat{A}$ indexed by the set $\mathcal{I}$ ($\mathcal{J}$). For a vector $v \in \Re^n$, $v^{-1} \in \Re^n$ denotes the vector whose components are the inverses of the components of v. We let $\mathrm{Diag}(v)$ represent the diagonal matrix with diagonal v. We use e to represent a vector of all ones. The dimension of e may differ but should be clear from the context.

2.2 A primal-dual smoothing method

Consider the following functions $f(x)$ and $\phi(u)$:

$$f(x) = \hat{f}(x) + \max_{u \in Q_2}\{\langle \hat{A}x, u\rangle - \hat{\phi}(u)\} \tag{2}$$

$$\phi(u) = -\hat{\phi}(u) + \min_{x \in Q_1}\{\langle \hat{A}x, u\rangle + \hat{f}(x)\}, \tag{3}$$

where Q_1 and Q_2 are (simple) compact convex sets in finite-dimensional Euclidean spaces E_1 and E_2, respectively; $\hat{A}$ is a linear operator mapping E_1 to E_2; and $\hat{f}(x)$ and $\hat{\phi}(u)$ are continuous convex functions on Q_1 and Q_2, respectively. Thus, $f(x)$ is convex and $\phi(u)$ is concave, but they are *not* necessarily differentiable. For any $\bar{x} \in Q_1$ and $\bar{u} \in Q_2$ we have

$$\phi(\bar{u}) \le f(\bar{x}) \tag{4}$$

because

$$\begin{aligned}
\phi(\bar{u}) &= -\hat{\phi}(\bar{u}) + \min_{x \in Q_1}\{\langle \hat{A}x, \bar{u}\rangle + \hat{f}(x)\} \\
&\le -\hat{\phi}(\bar{u}) + \langle \hat{A}\bar{x}, \bar{u}\rangle + \hat{f}(\bar{x}) \\
&\le \hat{f}(\bar{x}) + \max_{u \in Q_2}\{\langle \hat{A}\bar{x}, u\rangle - \hat{\phi}(u)\} \\
&= f(\bar{x}).
\end{aligned}$$

Nesterov's smoothing technique uses a primal-dual approach to solve the following optimization problems simultaneously:

$$\min\{f(x) : x \in Q_1\}, \tag{5}$$

$$\max\{\phi(u) : u \in Q_2\}. \tag{6}$$

Note that by Fenchel duality (Borwein and Lewis, 2006), (6) is the dual problem of (5) and there is no duality gap. Because of the nondifferentiability, the primal-dual approach does not directly deal with $f(x)$ and $\phi(u)$. Instead it works with the following "smoothed" versions:

$$f_{\mu_2}(x) = \hat{f}(x) + \max_{u \in Q_2}\{\langle \hat{A}x, u\rangle - \hat{\phi}(u) - \mu_2 d_2(u)\} \tag{7}$$

$$\phi_{\mu_1}(u) = -\hat{\phi}(u) + \min_{x \in Q_1}\{\langle \hat{A}x, u\rangle + \hat{f}(x) + \mu_1 d_1(x)\}, \tag{8}$$

where μ_i is a positive *smoothness* parameter and $d_i(\cdot)$ is a *prox-function* on the set Q_i,

which means $d_i(\cdot)$ is continuous and *strongly convex* on Q_i. A strongly convex function $d(\cdot)$ has the following property for some $\sigma > 0$: $d(x) \geq d(x^*) + \frac{1}{2}\sigma\|x - x^*\|^2$ for all $x \in Q$, where $x^* = \arg\min_{x \in Q} d(x)$. The purpose of introducing these prox-functions is to smooth $f(x)$ and $\phi(u)$. The resultant $f_{\mu_2}(x)$ and $\phi_{\mu_1}(u)$ are differentiable, and their gradients are Lipschitz-continuous. When μ_1 and μ_2 are small, $f_{\mu_2} \approx f$ and $\phi_{\mu_1} \approx \phi$.

By definition we have $f_{\mu_2}(x) \leq f(x)$ and $\phi(u) \leq \phi_{\mu_1}(u)$. For sufficiently large μ_1 and μ_2, one can show that there exist some $\bar{x} \in Q_1$ and $\bar{u} \in Q_2$ satisfying the following *excessive gap condition* (EGC):

$$f_{\mu_2}(\bar{x}) \leq \phi_{\mu_1}(\bar{u}). \tag{EGC}$$

(EGC) is like a switched version of (4), which ensures that the primal-dual gap is bounded above, as stated in the following lemma.

Lemma 2.2.1 (Nesterov (2005b))**.** *Let $\bar{x} \in Q_1$ and $\bar{u} \in Q_2$ satisfy (EGC). Then $0 \leq f(\bar{x}) - \phi(\bar{u}) \leq \mu_1 D_1 + \mu_2 D_2$, where $D_1 := \max_{x \in Q_1} d_1(x)$ and $D_2 := \max_{u \in Q_2} d_2(u)$.*

In addition to Lemma 2.2.1, the smoothing technique features three other important ingredients:

(i) A procedure that calculates an initial $(x^0, u^0, \mu_1^0, \mu_2^0)$ satisfying (EGC), namely, $f_{\mu_2^0}(x^0) \leq \phi_{\mu_1^0}(u^0)$;

(ii) Given $(x^k, u^k, \mu_1^k, \mu_2^k)$ satisfying (EGC), a procedure that generates $(x^{k+1}, u^{k+1}, \mu_1^{k+1}, \mu_2^{k+1})$ satisfying (EGC) as well;

(iii) $\mu_i^{k+1} \leq \mu_i^k,\ i = 1, 2$, where one of the two inequalities is strict and $\mu_i^k \to 0$, $i = 1, 2$.

Ingredients (i) and (ii) generate a sequence $\{(x^k, u^k, \mu_1^k, \mu_2^k)\}$ that satisfies (EGC) for each k. Because of Lemma 2.2.1 and (iii), the primal-dual gap goes to zero as k increases; that is,

$$0 \leq f(x^k) - \phi(u^k) \leq \mu_1^k D_1 + \mu_2^k D_2 \to 0. \tag{9}$$

As long as (EGC) is maintained, (9) will hold for all k.

Theorem 2.2.1. *(Nesterov (2005b)) Given $\epsilon > 0$, there is an algorithm based on the smoothing technique that produces a pair $(x^N, u^N) \in Q_1 \times Q_2$ such that*

$$0 \leq f(x^N) - \phi(u^N) \leq \epsilon$$

in

$$N = \frac{4\|\hat{A}\|_{1,2}}{\epsilon}\sqrt{\frac{D_1 D_2}{\sigma_1 \sigma_2}}$$

iterations.

In each iteration of the algorithm, one needs to update $\left(x^k, u^k, \mu_1^k, \mu_2^k\right)$, a process that requires solving several subproblems in the form of the inner max problem in (7) or the inner min problem in (8). Therefore, the solutions of these max and min problems should be easily computable. We omit the generic scheme here; we will describe the specific algorithm with respect to our problem in Section 3.3.

3 Applying the Smoothing Technique

In this section, we show how to convert (NS) into the standard form required by the smoothing technique. After the conversion, we specify all ingredients, including our choice of the prox-functions, the calculation of parameters for the smoothing technique, and the iteration complexity for solving (NS). In the last subsection, we detail the algorithm.

3.1 Reformulation

We define two notations that will be used throughout this section. Let $\square_m$ denote the standard box of dimension m:

$$\square_m := \{u \in \Re^m : 0 \leq u \leq e\}.$$

Let $\triangle_n$ denote the standard simplex of dimension n:

$$\triangle_n := \{x \in \Re^n : x \geq 0,\ \langle e, x\rangle = 1\}.$$

For notational convenience, we will drop the dimension subscript when the dimension is clear from context.

We first reformulate the nonsmooth part of the objective function, $\langle w, (A\alpha - b)^+\rangle$, as a

maximization problem. Since w is nonnegative, we have

$$\begin{aligned}\langle w, (A\alpha - b)^+\rangle &= \left\langle e, \operatorname{Diag}(w)\,(A\alpha - b)^+\right\rangle \\ &= \left\langle e, \Big(\operatorname{Diag}(w)(A\alpha - b)\Big)^+\right\rangle \\ &= \max_{u\in\square_m}\left\{\langle \operatorname{Diag}(w)(A\alpha - b), u\rangle\right\} \\ &= \max_{u\in\square_m}\left\{\langle \operatorname{Diag}(w)A\alpha, u\rangle - \langle \operatorname{Diag}(w)b, u\rangle\right\}.\end{aligned}$$

Next, we transform the primal feasible set $\{\alpha \in \Re^n_+ : \alpha_{\mathcal{B}} \le h, \langle c, \alpha\rangle \le \theta\}$ with $\theta > 0$ into a simpler set by changing variables. The purpose of the transformation is to simplify the problem presentation, especially to facilitate the application of Nesterov's method. Recall that $c_{\mathcal{B}} = 0$ and thus $\langle c, \alpha\rangle = \langle c_{\bar{\mathcal{B}}}, \alpha_{\bar{\mathcal{B}}}\rangle$, where $\bar{\mathcal{B}} := \{1, \ldots, n\}\backslash\mathcal{B}$. Define a new variable $x \in \Re^{n+1}$ as follows:

$$\begin{aligned} x_{\mathcal{B}} &:= \operatorname{Diag}\left(h^{-1}\right)\alpha_{\mathcal{B}} \\ x_{\mathcal{S}} &:= \tfrac{1}{\theta}\begin{pmatrix}\operatorname{Diag}\left(c_{\bar{\mathcal{B}}}\right)\alpha_{\bar{\mathcal{B}}} \\ \theta - \langle c_{\bar{\mathcal{B}}}, \alpha_{\bar{\mathcal{B}}}\rangle\end{pmatrix}, \end{aligned} \tag{10}$$

where $\mathcal{S} := \bar{\mathcal{B}} \cup \{n+1\}$. Because $0 \le \alpha_{\mathcal{B}} \le h$, we see that $x_{\mathcal{B}}$ is inside a box with dimension $|\mathcal{B}|$. Because $\alpha \ge 0$ and $\langle c, \alpha\rangle \le \theta$, $x_{\mathcal{S}}$ resides in a simplex with dimension $|S| = n + 1 - |\mathcal{B}|$. So the primal feasible set becomes

$$\{x \in \Re^{n+1} :\ x_{\mathcal{B}} \in \square,\ x_{\mathcal{S}} \in \triangle\}.$$

The last step of the reformulation involves defining a new set of data $\hat{A} \in \Re^{m\times(n+1)}$, $\hat{b} \in \Re^m$, and $\hat{e} \in \Re^{n+1}$ as follows:

$$\hat{A}^{\mathcal{S}} := \begin{pmatrix}\operatorname{Diag}(w)A^{\bar{\mathcal{B}}}\operatorname{Diag}(c_{\bar{\mathcal{B}}}^{-1}) & 0\end{pmatrix} \tag{11}$$

$$\hat{A}^{\mathcal{B}} := \frac{1}{\theta}\operatorname{Diag}(w)A^{\mathcal{B}}\operatorname{Diag}(h) \tag{12}$$

$$\begin{aligned}\hat{b} &:= \operatorname{Diag}(w)b \\ \hat{e}_{\mathcal{B}} &:= 0,\ \hat{e}_{\bar{\mathcal{B}}} := e,\ \hat{e}_{n+1} := 0.\end{aligned}$$

With the above definition, one can easily verify that the objective function of (NS) can be expressed in terms of x:

$$\langle c, \alpha\rangle + \langle w, (A\alpha - b)^+\rangle = \theta\left[\langle \hat{e}, x\rangle + \max_{u\in\square}\left\{\langle \hat{A}x, u\rangle - \frac{1}{\theta}\langle \hat{b}, u\rangle\right\}\right] =: \theta f(x;\theta), \tag{13}$$

and thus (NS) is equivalent to the following problem

$$\min\{f(x;\theta):\ x_{\mathcal{S}}\in\Delta,\ x_{\mathcal{B}}\in\square\}. \tag{SP}$$

Based on the primal-dual structure of (2) and (3), we immediately have the dual problem

$$\max\{\phi(u;\theta):\ u\in\square\}, \tag{SD}$$

where

$$\phi(u;\theta):=-\frac{1}{\theta}\langle\hat{b},u\rangle+\min_{x_{\mathcal{S}}\in\Delta,x_{\mathcal{B}}\in\square}\Big\{\langle\hat{A}x,u\rangle+\langle\hat{e},x\rangle\Big\}.$$

The smoothing technique we described in Section 2.2 can be used to solve (SP) and (SD).

For any primal feasible solution $\bar{x}$ obtained from the smoothing technique, it is easy to recover a feasible solution $\bar{\alpha}$ to (NS) by just reversing the transformation. Note that by (13), the objective $f(\bar{x};\theta)$ needs to be scaled by θ in order to recover the objective value of (NS). In fact, this property will influence the iteration complexity of the smoothing technique. In particular, the primal error (the absolute difference between the primal objective value and the optimal value) of (NS) is the primal error of (SP) scaled by θ.

Lemma 3.1.1. *Define $p(\alpha):=\langle c,\alpha\rangle+\langle w,(A\alpha-b)^{+}\rangle$, and let x^* be an optimal solution of (SP). Recall θ^* denotes the optimal value of (NS). Suppose $\bar{x}$ is feasible to (SP) and $\bar{\alpha}$ is the corresponding feasible solution to (NS). Then*

$$p(\bar{\alpha})-\theta^*=\theta\Big(f(\bar{x};\theta)-f(x^*;\theta)\Big). \tag{14}$$

Proof. $\bar{\alpha}$ (α^*) can be obtained by $\bar{x}$ (x^*) through the relationship (10). By (13), $p(\bar{\alpha})=\theta f(\bar{x};\theta)$ and $p(\alpha^*)=\theta^*=\theta f(x^*;\theta)$, and the result follows. □

3.2 Specifications

In this subsection, we discuss in detail each ingredient of the smoothing technique. The choice of norms and prox-functions is critical. We select the l_1-norm and the entropy distance function as the prox-function (see Nesterov (2005a)) for the primal space, and the l_2-norm

and a "distance squared" quadratic function for the dual space:

$$\|x\|_1 := \sum_{i=1}^{n+1} |x_i|, \qquad d_1(x) := \ln(|\mathcal{S}|) + |\mathcal{B}| \cdot \exp(-1) + \sum_{i=1}^{n+1} x_i \ln x_i \tag{15}$$

$$\|u\|_2 := \sqrt{\sum_{j=1}^{m} (u_j)^2}, \quad d_2(u) := \frac{1}{2} \sum_{j=1}^{m} \left(u_j - \frac{1}{2} \right)^2. \tag{16}$$

With these choices, we calculate the parameters that determine the iteration complexity of the smoothing technique:

$$D_1 = \max_x \{ d_1(x) : x_{\mathcal{S}} \in \Delta,\ x_{\mathcal{B}} \in \square \} = \ln(|\mathcal{S}|) + |\mathcal{B}| \cdot \exp(-1), \quad \sigma_1 = 1,$$

the derivation of which can be found in Lemma 3 of Nesterov (2005a). The prox-function $d_1(\cdot)$ achieves its minimum at $x_0 = e/(n+1)$, where $d_1(x_0) = 0$. It is easy to verify that

$$D_2 = \max_u \{ d_2(u) : u \in \square_m \} = \frac{m}{8}, \quad \sigma_2 = 1,$$

and $d_2(\cdot)$ achieves its minimum at $u_0 = \frac{1}{2}e$. The operator norm of $\hat{A}$ is thus

$$\begin{aligned}
\|\hat{A}\|_{1,2} &= \max_u \left\{ \|\hat{A}^T u\|_1^* : \|u\|_2 = 1 \right\} \\
&= \max_u \left\{ \max_{i \in \{1,\ldots,n+1\}} \left\{ \langle \hat{A}^i, u \rangle \right\} : \|u\|_2 = 1 \right\} \\
&= \max_{i \in \{1,\ldots,n\}} \left\{ \max_u \left\{ \langle \hat{A}^i, u \rangle : \|u\|_2 = 1 \right\} \right\} \\
&= \max_{i \in \{1,\ldots,n\}} \|\hat{A}^i\|_2.
\end{aligned}$$

The second equality follows because the dual norm of the l_1-norm is the l_∞-norm. The third equality follows from the fact that $\hat{A}$'s last column is zero, as shown by (11).

With all the parameters computed, we are ready to state the iteration complexity of solving (SP) and (SD).

Proposition 3.2.1. *Using Nesterov's smoothing technique, for any $\epsilon > 0$, we obtain a pair of solutions (x^N, u^N) to (SP) and (SD) such that*

$$0 \leq f(x^N; \theta) - \phi(u^N; \theta) \leq \epsilon$$

in

$$N := N(\theta) := \frac{1}{\epsilon}\left(\max_{i\in\{1,\ldots,n\}} \|\hat{A}^i\|_2\right)\sqrt{2\left[\ln(|\mathcal{S}|) + |\mathcal{B}|\cdot\exp(-1)\right]\cdot m} \tag{17}$$

iterations.

Proof. The result is obtained by applying Theorem 2.2.1. □

By (12), $\hat{A}^{\mathcal{B}}$ is dependent on $1/\theta$, and so the number of iterations of the smoothing technique depends on θ. We thus write $N := N(\theta)$ as a function of θ to reflect this dependence.

We comment that different combinations of norms and prox-functions other than (15) and (16) may lead to different parameter values and thus different iteration complexities. We considered several choices for the dual space, and the choice (16) gives us the lowest iteration complexity among those considered. For example, one could choose the l_1-norm and the same prox-function as $d_1(x)$ for the dual space; the resultant iteration complexity is $O(\sqrt{m})$ times larger than (17), which is much worse than the current one if m is large.

In Proposition 3.2.1, the iteration complexity is stated with respect to (SP). Now we state the iteration complexity with respect to (NS).

Proposition 3.2.2. *Using Nesterov's smoothing technique, for any $\epsilon > 0$, we obtain a solution $\bar{\alpha}$ to (NS) such that $0 \le p(\bar{\alpha}) - \theta^* \le \epsilon$ in*

$$N' := \theta N(\theta)$$

iterations, where $N(\theta)$ is given by (17).

Proof. By Proposition 3.2.1, in N' iterations, we obtain a solution $(\bar{x}, \bar{u})$ such that the primal-dual gap is small enough:

$$0 \le f(\bar{x};\theta) - \phi(\bar{u};\theta) \le \frac{\epsilon}{\theta}.$$

We then can construct $\bar{\alpha}$ feasible to (NS). Thus, by (14), we have

$$0 \le p(\bar{\alpha}) - \theta^* = \theta\Big(f(\bar{x};\theta) - f(x^*;\theta)\Big) \le \theta\Big(f(\bar{x};\theta) - \phi(\bar{u};\theta)\Big) \le \epsilon.$$

□

From this proposition, the iteration complexity of solving (NS) is a function of the upper bound θ. This is not so surprising because the primal feasible set is originally unbounded, and the hidden upper bound θ is discovered by exploiting the structure of the objective function. In particular, using (17), we see that $N' = \theta N(\theta)$ is increasing (more precisely,

non-decreasing) in θ. So the smaller θ is, the better. One could always apply some heuristics to get a good θ. In Section 4, we instead introduce a strategy that dynamically updates θ and consequently reduces the iteration complexity for solving (NS) as the algorithm progresses. From now on, we also drop the θ from $N(\theta)$ for notational convenience.

Next we discuss the subproblems associated with our choice of prox-functions. These subproblems will be solved repeatedly in the algorithm presented in Section 3.3, and thus it is important to have closed-form solutions for them. The subproblems are the max and min problems presented within the following smoothed versions of our primal and dual objective functions (recall (7) and (8)):

$$f_{\mu_2}(x;\theta) = \langle \hat{e}, x\rangle + \max_{u\in\square_m}\left\{\langle \hat{A}x, u\rangle - \frac{1}{\theta}\langle \hat{b}, u\rangle - \mu_2 d_2(u)\right\}$$
$$\phi_{\mu_1}(u;\theta) = -\frac{1}{\theta}\langle \hat{b}, u\rangle + \min_{x_{\mathcal{S}}\in\Delta,\ x_{\mathcal{B}}\in\square}\left\{\langle \hat{A}x, u\rangle + \langle \hat{e}, x\rangle + \mu_1 d_1(x)\right\},$$

where $d_1(x)$ and $d_2(u)$ are given in (15) and (16).

Consider the min subproblem first. Its solution is $\texttt{sargmin}(d_1, -\frac{1}{\mu_1}(\hat{A}^T u + \hat{e}))$, where

$$\texttt{sargmin}(d_1, s) := \arg\min_{x\in Q_1}\{-\langle s, x\rangle + d_1(x)\} = \arg\min_{x_{\mathcal{S}}\in\Delta, x_{\mathcal{B}}\in\square}\left\{-\langle s, x\rangle + \sum_{i=1}^{n+1} x_i \ln x_i\right\}.$$

The following lemma establishes a closed-form for $\texttt{sargmin}(d_1, s)$:

Lemma 3.2.1. *Given s, the solution $\texttt{sargmin}(d_1, s)$ is given by*

$$[\texttt{sargmin}(d_1, s)]_i = \begin{cases} \frac{\exp(s_i)}{\sum_{j=1}^{|\mathcal{S}|}\exp(s_j)}, & i\in\mathcal{S} \\ \text{proj}_{[0,1]}\left(\exp\left(s_i\right)\right), & i\in\mathcal{B}\end{cases},$$

where $\text{proj}_{[0,1]}(y)$ projects y to the nearest point between 0 and 1.

Proof. The objective function is separable in $\mathcal{S}$ and $\mathcal{B}$. For optimizing this function over the simplex, see Lemma 4 in Nesterov (2005a). For optimizing over the box, we compute the point at which the first-order derivative vanishes and then project that point back to the feasible region. □

Now consider the max subproblem. In a similar manner, its solution is $\texttt{sargmin}(d_2, \frac{1}{\mu_2}(\hat{A}x - \hat{b}/\theta))$, where

$$\texttt{sargmin}(d_2, s) := \arg\min_{u\in Q_2}\{-\langle s, u\rangle + d_2(u)\} = \arg\min_{u\in\square}\left\{-\langle s, u\rangle + \sum_{j=1}^{m}\left(u_j - \tfrac{1}{2}\right)^2\right\}.$$

This problem also has a closed-form solution:

Lemma 3.2.2. *Given s, the solution* $\texttt{sargmin}(d_2, s)$ *is given by*

$$[\texttt{sargmin}(d_2, s)]_j = \text{proj}_{[0,1]}\left(\tfrac{1}{2}(1+s_j)\right), \qquad j = 1, \ldots, m,$$

where $\text{proj}_{[0,1]}(y)$ *projects y to the nearest point between* 0 *and* 1.

Proof. Observe that

$$-\langle s, u\rangle + \sum_{j=1}^{m}\left(u_j - \tfrac{1}{2}\right)^2 = \sum_{j=1}^{m}\left(u_j^2 - (s_j+1)\,u_j + \tfrac{1}{4}\right).$$

So the problem reduces to solving m one-dimensional quadratic problems whose solutions are as stated. □

3.3 Algorithm

The algorithmic scheme presented by Nesterov (2005b) is generic, and this problem-specific parameters for our problem have been calculated in Sections 3.1 and 3.2. In this subsection, we explicitly state the scheme with respect to (SP) and (SD). In this algorithm, there are three functions: (i) INITIAL initializes all the parameters and the primal-dual solution (x^0, u^0) satisfying (EGC); (ii) UPDATE1 is the primal update; (iii) UPDATE2 is the dual update. UPDATE1 and UPDATE2 are symmetric but entail different subproblems. Here is INITIAL.

Algorithm 1 INITIAL

Input: Data $(m, n, \hat{A}, \hat{b}, \hat{e}, \theta)$

Output: Initialized parameters (μ_1^0, μ_2^0) and solutions (x^0, u^0) that satisfy (EGC)

1: $D_1 = \ln(|\mathcal{S}|) + |\mathcal{B}| \cdot \exp(-1)$, $D_2 = m/8$, $\sigma_1 = \sigma_2 = 1$, $\|\hat{A}\|_{1,2} = \max_{i\in\{1,\ldots,n\}} \|\hat{A}^i\|_2$

2: $\mu_1^0 = 2\,\|\hat{A}\|_{1,2}\sqrt{\frac{D_2}{\sigma_1\sigma_2 D_1}}$, $\mu_2^0 = \|\hat{A}\|_{1,2}\sqrt{\frac{D_1}{\sigma_1\sigma_2 D_2}}$

3: $\bar{x} = \texttt{sargmin}(d_1, 0)$

4: $u^0 = \texttt{sargmin}\left(d_2, \frac{1}{\mu_2^0}(\hat{A}\bar{x} - \hat{b}/\theta)\right)$

5: $x^0 = \texttt{sargmin}\left(d_1, \ln(\bar{x}) + e - \frac{\mu_2^0}{\|\hat{A}\|_{1,2}^2}(A^T u^0 + \hat{e})\right)$

Here is the primal update (when k is even).

Algorithm 2 UPDATE1: primal update

Input: Current solution (x, u) and parameters $(\mu_1, \mu_2, \tau, \theta)$

Output: $(x^+, u^+, \mu_1^+, \mu_2^+)$ that satisfy (EGC)

1: $\bar{x} = \texttt{sargmin}\left(d_1, -\frac{1}{\mu_1}(\hat{A}^T u + \hat{e})\right)$

2: $\hat{x} = (1-\tau)x + \tau\bar{x}$

3: $\bar{u} = \texttt{sargmin}(d_2, \frac{1}{\mu_2}(\hat{A}\,\hat{x} - \hat{b}/\theta))$

4: $\hat{x} = \texttt{sargmin}\left(d_1, \ln(\bar{x}) + e - \frac{\tau}{(1-\tau)\mu_1}(\hat{A}^T\bar{u} + \hat{e})\right)$

5: $x^+ = (1-\tau)x + \tau\hat{x}$

6: $u^+ = (1-\tau)u + \tau\bar{u}$

7: $\mu_1^+ = (1-\tau)\mu_1$, $\mu_2^+ = \mu_2$

Here is the dual update (when k is odd).

Algorithm 3 UPDATE2: dual update

Input: Current solution (x, u) and parameters $(\mu_1, \mu_2, \tau, \theta)$

Output: $(x^+, u^+, \mu_1^+, \mu_2^+)$ that satisfy (EGC)

1: $\bar{u} = \texttt{sargmin}(d_2, \frac{1}{\mu_2}(\hat{A}\,x - \hat{b}/\theta))$

2: $\hat{u} = (1-\tau)u + \tau\bar{u}$

3: $\bar{x} = \texttt{sargmin}(d_1, -\frac{1}{\mu_1}(\hat{A}^T\hat{u} + \hat{e}))$

4: $\hat{u} = \texttt{sargmin}(d_2, \frac{\tau}{(1-\tau)\mu_2}(\hat{A}\,\bar{x} - \hat{b}/\theta) + \bar{u} - \frac{1}{2}e)$

5: $x^+ = (1-\tau)x + \tau\bar{x}$

6: $u^+ = (1-\tau)u + \tau\hat{u}$

7: $\mu_2^+ = (1-\tau)\mu_2$, $\mu_1^+ = \mu_1$

The entire SMOOTH algorithm is as follows.

Algorithm 4 SMOOTH

Input: (i) Data $(m, n, \hat{A}, \hat{b}, \hat{e}, \theta, N')$; (ii) Subroutines $\texttt{sargmin}(d_1, \cdot)$ and $\texttt{sargmin}(d_2, \cdot)$

Output: $(x^{N'}, u^{N'})$

$(x^0, u^0, \mu_1^0, \mu_2^0) =$ INITIAL $(m, n, \hat{A}, \hat{b}, \hat{e}, \theta)$

for $k = 0, 1, \ldots, N' - 1$ **do**

  $\tau = \frac{2}{k+3}$

  if k is even **then**

    $(x^{k+1}, u^{k+1}, \mu_1^{k+1}, \mu_2^{k+1}) =$ UPDATE1$(x^k, u^k, \mu_1^k, \mu_2^k, \tau, \theta)$

  else

    $(x^{k+1}, u^{k+1}, \mu_1^{k+1}, \mu_2^{k+1}) =$ UPDATE2$(x^k, u^k, \mu_1^k, \mu_2^k, \tau, \theta)$

We have two comments regarding the SMOOTH algorithm: First, according to Lemma 3.2.1 and Lemma 3.2.2, the subproblems have closed-form solutions and can be solved quickly. The most time-consuming operations are thus the matrix-vector multiplications $\hat{A}\bar{x}$ and $\hat{A}^T\bar{u}$. Second, θ is treated as an fixed parameter for UPDATE1 and UPDATE2. In the next section, we will discuss a simple procedure that dynamically updates θ, and the algorithm will be valid even with changing values of θ.

4 Speeding Up the Convergence

As shown in Proposition 3.2.2 and its subsequent discussion, the iteration complexity for obtaining an ϵ-solution of (NS) is directly related to θ/ϵ, where $\theta > 0$ is an upper bound on the optimal value of (NS). As an input parameter, the smaller θ is, the better the iteration complexity, and the best possible θ is the optimal value θ^*. We will of course obtain new information on θ^* as Algorithm 4 progresses; in particular, an improved primal objective value will give a better upper bound on θ^* than θ. To take advantage of this information, we consider updating θ dynamically within Algorithm 4.

Suppose we have a better bound $\theta^+ \in [\theta^*, \theta)$ available after running Algorithm 4 for K iterations, where $K < N' = \theta N$. Recall N, as defined in (17), depends on the error tolerance ϵ. With θ^+ on hand, it may be worthwhile to restart the algorithm and input θ^+ for the new run, if the number of iterations required by the new run plus the number of iterations already run is smaller than the iteration estimate under θ, that is, if $K + \theta^+ N(\theta^+) < \theta N(\theta)$ holds.

Even better, we will show that it is possible to improve the iteration complexity to

$\theta^+ N(\theta^+)$—without restarting the algorithm—provided that the condition (EGC) is carefully maintained when updating the parameter θ to θ^+. To achieve this result, let $\theta > \theta^*$ be the current upper bound on the optimal value, and let k be the iteration counter. The following lemma shows, under mild conditions, the existence of $\theta^+ \in (\theta^*, \theta)$ and k such that $(x^k, u^k, \mu_1^k, \mu_2^k)$ satisfies (EGC) with respect to θ^+.

Lemma 4.0.1. *Within Algorithm 4, suppose $(x^k, u^k, \mu_1^k, \mu_2^k)$ satisfies (EGC) strictly with respect to θ; that is, $f_{\mu_2^k}(x^k;\theta) < \phi_{\mu_1^k}(u^k;\theta)$. Suppose also that $f(x^k;\theta) < 1$. Then there exists $\theta^+ \in (\theta^*, \theta)$ such that $(x^k, u^k, \mu_1^k, \mu_2^k)$ also satisfies (EGC) for θ^+:*

$$f_{\mu_2^k}(x^k;\theta^+) \le \phi_{\mu_1^k}(u^k;\theta^+). \tag{18}$$

Proof. Define $\theta^k := \theta f(x^k;\theta)$. Based on the relationship (14), we know $\theta^k \ge \theta^*$, i.e., θ^k is a valid upper bound on θ^*. Then $f(x^k;\theta) < 1$ implies $\theta^k \in [\theta^*, \theta)$. If $f_{\mu_2^k}(x^k;\theta^k) \le \phi_{\mu_1^k}(u^k;\theta^k)$ holds, simply set $\theta^+ := \theta^k$. Otherwise, $g(\tau) := f_{\mu_2^k}(x^k;\tau) - \phi_{\mu_1^k}(u^k;\tau)$ is a continuous function of τ, and we have $g(\theta^k) > 0$ and $g(\theta) < 0$. So there exists $\theta^+ \in (\theta^k, \theta)$ such that $g(\theta^+) \le 0$. In other words, (18) holds. □

We remark that, intuitively, if the error tolerance ϵ is small enough, the condition $f(x^k;\theta) < 1$ in the lemma eventually holds for large enough k since $f(x^*;\theta) = \theta^*/\theta < 1$.

From now on, we use p^k to represent the primal value $p(\alpha^k)$ for notational convenience. By (9), we have

$$f(x^k;\theta) - \phi(u^k;\theta) \le \mu_1^k D_1 + \mu_2^k D_2 =: U^k, \quad k = 1, 2, \ldots,$$

where the bounding sequence $\{U^k\}_{k=1}^{\infty}$ is independent of θ. Thus, by (14), the primal error $p^k - \theta^*$ at iteration k is bounded above by θU^k because

$$p^k - \theta^* = \theta \left(f(x^k;\theta) - f(x^*;\theta)\right) \le \theta \left(f(x^k;\theta) - \phi(u^k;\theta)\right) \le \theta U^k, \quad k = 1, 2, \ldots. \tag{19}$$

On the other hand, because (EGC) holds for θ^+ by Lemma 4.0.1, an identical argument shows that

$$p^k - \theta^* \le \theta^+ U^k. \tag{20}$$

So, instead of θU^k, the improved upper bound $\theta^+ U^k$ is proven to hold at iteration k. Since the smoothing technique guarantees that (EGC) is maintained in the next iteration as long as it holds at the current iteration, immediately switching θ to θ^+ allows us to bound the primal error from the current iteration onwards by the sequence $\{\theta^+ U^l\}_{l=k}^{\infty}$. Said differently, updating θ with θ^+ helps speed up the convergence as the subsequent primal error is bounded

above by the new sequence $\left\{\theta^+ U^l\right\}_{l=k}^{\infty}$. Note that the primal-daul solutions generated in subsequent iterations are different than those that would have been generated with the parameter θ because the parameter θ^+ affects the subroutines UPDATE1 and UPDATE2. Based on this analysis, we have the following proposition.

Proposition 4.0.1. *If, during the course of Algorithm 4, a new upper bound $\theta^+ \in [\theta^*, \theta)$ satisfying (18) is employed for Algorithm 4, then the number of iterations required to obtain an ϵ-solution of (NS) reduces from θN to $\theta^+ N$.*

We have a couple of comments. First, we have demonstrated the existence of θ^+ in Lemma 4.0.1, but there appears to be no closed-form formula for it. One simple procedure to obtain θ^+ is as follows. Periodically check the conditions of Lemma 4.0.1. If they hold, then do the following: (i) set $\theta^+ := \theta f(x^k; \theta)$; (ii) while $f_{\mu_2}(x^k; \theta^+) > \phi_{\mu_1}(u^k; \theta^+)$, set $\theta^+ := 1/2\ (\theta^+ + \theta)$. Steps (i) and (ii) will not affect other parts of the SMOOTH algorithm and can be put into the algorithm conveniently.

Second, to maintain the excessive gap condition, one can easily see that θ^+ should be chosen in a neighborhood of θ, and so the reduction of the number of iterations by updating θ may not be big. However, the procedure of updating θ can be performed repeatedly. As the algorithm converges, the updated parameter value becomes a better and better approximation of θ^*, and thus the cumulative improvements might be significant.

We close this section with an example that illustrates the effects of dynamically updating θ. The example is an instance from the first application that will be discussed in Section 5.1 and is created with the dataset *dermatology* from Asuncion and Newman (2007). The data matrix A in (P) has dimension 6760×358. We did three runs of the SMOOTH algorithm with the following variations:

(i) Initial input of trivial $\theta = \langle w, (-b)^+ \rangle$, and the SMOOTH algorithm is run without updating θ;

(ii) Initial input of trivial $\theta = \langle w, (-b)^+ \rangle$, and the SMOOTH algorithm is run with the dynamic update of θ;

(iii) Initial input of $\theta = \theta^*$, where the optimal value θ^* has been calculated using a standard LP solver, and there is no update on θ because the bound is already optimal.

While case (iii) is not realistic because θ^* is the idealized best upper bound, which is gotten by pre-solving the LP, it serves as a best-case comparison for cases (i) and (ii).

Figure 1 shows the results in log-log scale. The first subplot shows how the primal error $p^k - \theta^*$ changes over time. One can see that dynamically updating θ reduces the error

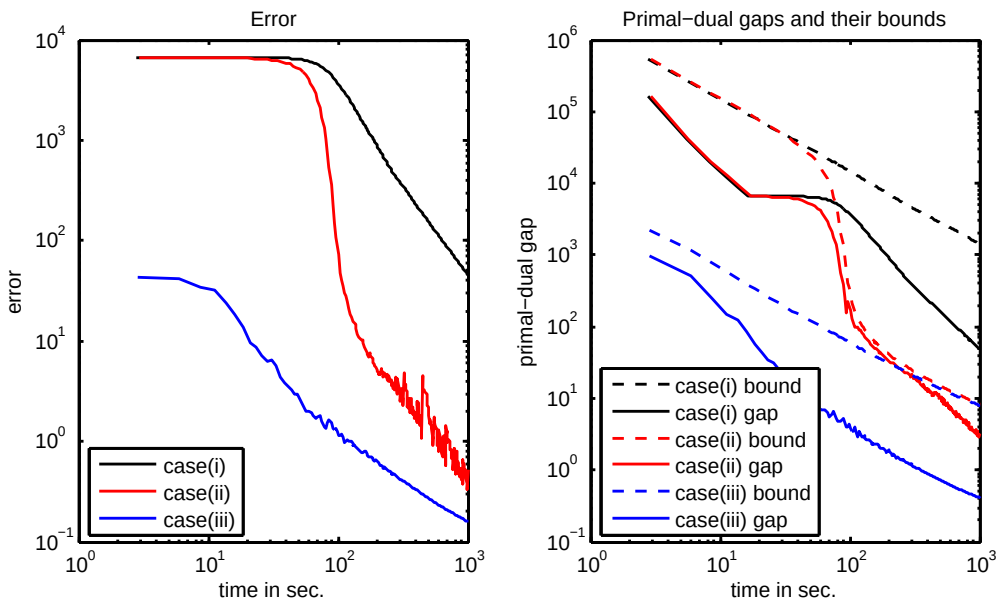

Figure 1: Comparison of running SMOOTH algorithm under cases (i)–(iii). The left subplot shows the change of the primal error $p(\alpha) - \theta^*$ over time; the right subplot shows how the primal-dual gap changes over time and compares it with the theoretical upper bounds.

significantly and its performance is almost as good as the ideal case (iii). We also comment that, in this particular example, $\theta^* \ll \langle w, (-b)^+ \rangle$ and thus the starting value of the error for case (iii) is far smaller than for (i) and (ii). Note also that (i) and (ii) are identical until the iteration when θ is first updated. The second subplot demonstrates how the primal-dual gap $\theta \left(f(x^k; \theta) - \phi(u^k; \theta) \right)$ changes over time for the three cases (depicted as solid lines). It also plots the upper bound sequence θU^k (depicted as dashed lines) to show how θ affects the primal-dual gap. Note that for this example, the upper bound associated with case (ii) changes over time because θ is being updated repeatedly.

5 Applications and Computational Experiments

In this section, we study the application of Algorithm 4 for solving two machine learning problems: (i) a linear programming based ranking method (Ataman et al., 2006; Ataman, 2007); (ii) 1-norm SVMs (see, for example, Mangasarian (2006) for the formulation of the 1-norm SVM and more references). In the following two subsections, we first briefly describe the two applications and reformulate them as (P). We then conduct computational experiments to compare Algorithm 4 with existing algorithms for the respective applications.

5.1 Linear programming-based ranking method

The linear programming-based ranking method proposed in Ataman (2007) is designed to train a scoring function that ranks all positive points higher than all negative points (from data that is assumed to have binary output). This ranking method is reported to perform better than SVM-based ranking algorithms.

Let (x_l, y_l) be an instance in the training set $X \ni l$. The class label is $y_l \in \{1, -1\}$. Let X^+, X^- represent the set of points with positive and negative labels, respectively. Let $K(\cdot, \cdot)$ denote a chosen kernel function, for example, the RBF kernel function. At its core, the ranking method is the following optimization problem:

$$\begin{aligned} \min \quad & \sum_{l \in X} \alpha_l + C \sum_{i \in X^+,\ j \in X^-} w_{i,j} \xi_{i,j} \\ \text{s.t.} \quad & \sum_{l \in X} y_l \left[K(x_i, x_l) - K(x_j, x_l) \right] \alpha_l \geq 1 - \xi_{i,j} \quad \forall\ i \in X^+, j \in X^- \\ & \alpha \geq 0,\ \xi \geq 0, \end{aligned} \tag{LPR}$$

where $\alpha \in \Re^{|X|}$ and $\xi \in \Re^{|X^+| \times |X^-|}$ are the decision vectors and all else is data. It is assumed that $C > 0$ and $w_{i,j} \geq 0$ for all $i \in X^+, j \in X^-$.

To put (LPR) in the form of (P), we define

$$\begin{aligned} A_{i \times j, l} &:= -y_l \left[K(x_i, x_l) - K(x_j, x_l) \right], \quad \forall\ i \times j \in X^+ \times X^-, l \in X \\ m &:= |X^+| \times |X^-| \\ n &:= |X| = |X^+| + |X^-|, \end{aligned}$$

such that $A \in \Re^{m \times n}$. Now (LPR) is readily modeled as (P) with data

$$c = e \in \Re^n, \quad w = Ce \in \Re^m, \quad b = -e \in \Re^m.$$

Notice that m grows quadratically as the number of data points grows, and thus A becomes large-scale even for medium-sized datasets. In addition, since every entry of A is the difference of two kernel functions, A is usually fully dense.

In Ataman (2007), the subgradient method is proposed for (LPR) because the method is memory efficient and thus is able to solve large-scale instances. In our computational study, we compare Algorithm 4 with the subgradient method implemented in Ataman (2007), which is essentially the incremental subgradient method (Nedic and Bertsekas, 2001). We collected 14 datasets from the UCI Machine Learning repository (Asuncion and Newman, 2007) and prepare them in the same way as in Ataman (2007). Table 1 describes the processed data

A. We point out that among the problems in Table 1, *cancer* and *diabetes* cannot be solved by a commercial LP solver, like CPLEX (via either primal or dual LP formulation), on a machine with 4 GB RAM without perhaps some special handling of the memory.

Table 1: Statistics of A: dimension, percentage of nonzeros, and storage size in Matlab format (instances are ordered increasingly by the number of nonzeros).

Instance	m	n	Nonzeros (%)	Size (MB)
wine	6240	178	11%	0.2
iris	5000	150	100%	5.4
glass	5365	214	100%	8.3
ntyroid	5550	215	98%	7.3
sonar	10767	208	100%	16.4
derma	6760	358	93%	6.6
heart	18000	270	100%	32.9
ecoli	14768	336	100%	36.3
spectf	24638	351	93%	20.0
ion	28350	351	100%	72.7
liver	29000	345	100%	60.6
boston	21984	506	100%	50.2
cancer	75684	569	100%	314.6
diabetes	134000	768	96%	409.1

One benefit of the smoothing technique is the availability of the primal-dual gap, which can serve as a good stopping criterion. On the contrary, it is not easy to obtain a primal-dual gap for the subgradient method.[1] So, in order to compare the two methods without a common stopping criterion, we perform the computational experiments in the following way. First, we solve each instance via Algorithm 4 with error tolerance $\epsilon = 1$ and obtain the best objective value found by the smoothing technique. Second, we run the subgradient method until it finds at least as good an objective value as the smoothing technique, or until it reaches the time limit. We set a time limit of 18,000 seconds for both methods. All computations were performed on a Pentium D running at 3.2 GHz under the Linux operating system with 4 GB RAM.

Table 2 presents the CPU times and the best objective values found when the algorithms terminate. Except for the two largest instances, we have the optimal values of the ranking problems available for gauging the quality of the solutions, as listed under the column θ^*. The subgradient method was able to find as good a solution as the smoothing technique in

[1] Nesterov (2009) proposes a primal-dual subgradient scheme, and we have implemented it for comparison. However, empirically we found the estimate (upper bound) of the primal-dual gap of the subgradient method (see (3.3) in Nesterov (2009)) is pessimistic and is usually much larger than the gap of the smoothing technique when both methods achieve similar primal values.

5 out of the 14 instances within the time limit and is faster in 3 out of those 5 instances. For the remaining 9 instances, the smoothing technique either is faster than the subgradient method in achieving the same quality solution or obtains better solutions in the same amount of time.

Table 2: Comparison of Algorithm 4 (SMOOTH) and the subgradient method (SUBG.) when applied to 14 linear ranking problems. θ^* is the optimal value of the ranking problem, but it is not available for the two largest instances. Times are in seconds and rounded to the nearest integers. Here t represents that the time limit (18,000 sec.) was exceeded.

	Time (in sec.)			Best Obj.	
Dataset	SMOOTH	SUBG.	θ^*	SMOOTH	SUBG.
wine	111	**39**	77.82	78.82	78.76
iris	**1104**	7001	3317.72	3318.72	3318.72
glass	**1296**	t	2850.45	2851.45	2852.00
ntyroid	**1232**	4153	1694.65	1695.65	1695.65
sonar	2136	**1055**	191.85	192.85	192.85
derma	119	**7**	28.71	29.67	29.36
heart	**5687**	t	1240.92	1241.92	1249.52
ecoli	t	t	8421.05	**8422.60**	8453.03
spectf	**11203**	t	1580.99	1587.17	1634.22
ion	t	t	2578.49	**2583.15**	2634.61
liver	t	t	11339.9	**11370.30**	11641.30
boston	t	t	889.88	**890.94**	899.03
cancer	t	t	—	**17989.09**	27520.79
diabetes	t	t	—	**14327.73**	28070.29

We also plot the primal objective value errors of both methods versus the CPU times for one instance, *glass*, in Figure 2. We comment that Figure 2 reflects the typical behavior of the two methods. The subgradient method finds good solutions rapidly, but its convergence is slow; the smoothing technique's initial objective value is usually not as good but converges faster, as predicted by theory. This observation suggests that the subgradient method might be a better choice if obtaining a good solution in a short time is important. On the other hand, if the accuracy of the optimization problem is critical, the smoothing technique is the better choice for large-scale problems.

5.2 1-norm support vector machines

Support vector machines (SVMs) are popular techniques for classification, and 1-norm SVMs are known to be effective in reducing input space features. Existing solution approaches for the 1-norm SVM include solving them as LPs and using a generalized Newton method or its

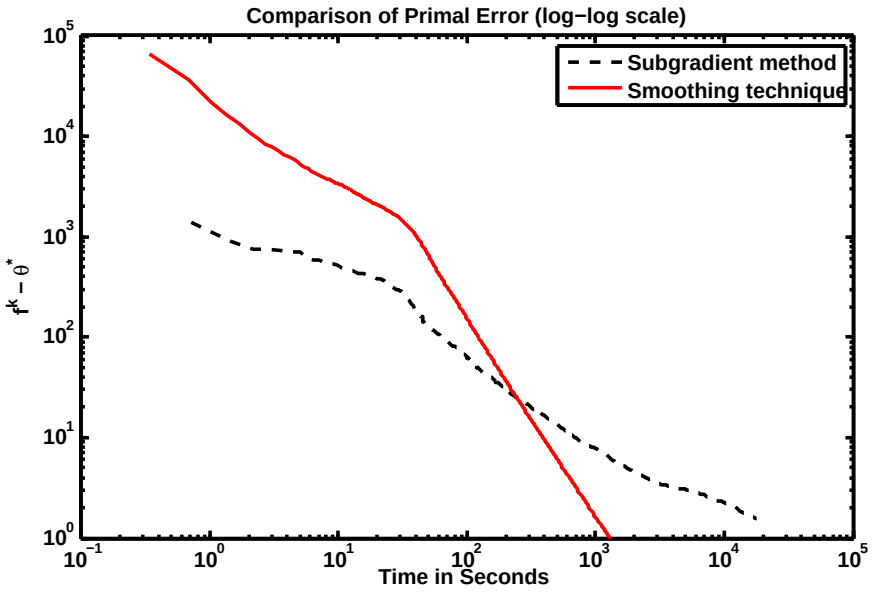

Figure 2: Error versus time (log-log scale): comparison of the smoothing technique and the subgradient method.

variants (Fung and Mangasarian, 2004; Mangasarian, 2006). We show in this subsection that Algorithm 4 can also be applied to solve 1-norm SVMs. First, we introduce the standard form of the 1-norm SVM and reformulate it as (P). Second, we compare Algorithm 4 with the generalized Newton method proposed in Mangasarian (2006) for solving linear 1-norm SVM and nonlinear kernel 1-norm SVM problems. From now on, we will refer to the method of Mangasarian (2006) as *Newton* for short.

A standard 1-norm SVM is the following optimization problem

$$\begin{aligned} \min_{(x,\gamma,\xi)} \quad & \|x\|_1 + C\|\xi\|_1 && \text{(1-norm SVM)} \\ \text{s.t.} \quad & D(\tilde{A}x - e\gamma) + \xi \geq e && (21) \\ & \xi \geq 0, \end{aligned}$$

where $\tilde{A} \in \Re^{m\times n}$ represents m points in $\Re^n$ to be separated by a hyperplane

$$a^T x = \gamma. \tag{22}$$

$D \in \Re^{m\times m}$ is a diagonal matrix with element $D_{ii} \in \{-1, 1\}$ representing the label for the i-th data point, and $C > 0$ is a trade-off parameter.

We claim that the variables (x, γ, ξ) are implicitly bounded. In fact, since the problem is a minimization problem, there exists $M > 0$ such that $\|x^*\|_1 + C\|\xi^*\|_1 < M$, where (ξ^*, x^*)

is an optimal solution to (1-norm SVM). Thus both x^* and ξ^* are bounded. At optimality, (21) can be rewritten as

$$De\,\gamma^* \leq D\tilde{A}\,x^* + \xi^* - e,$$

and it follows from the boundedness of (x^*, ξ^*) that the right-hand side of the above inequality is bounded. Therefore, there exists a scalar $h > 0$ such that $De\,\gamma^* \leq h\,e$, which implies that $\gamma^* \in [-h, h]$ because De's components are either 1 or -1. To actually compute h, note that the absolute value of each component of x^* and ξ^* is also bounded by M, and thus one can compute an upper (if $D_{ii} > 0$) or lower (if $D_{ii} < 0$) estimate of the ith component of $D\tilde{A}x^* + \xi^* - e$ as follows:

$$\beta(i) = \begin{cases} (\tilde{A}_i)^+(M\,e) + (M-1), & \text{if } D_{ii} = 1 \\ -(-\tilde{A}_i)^+(M\,e) + 1, & \text{if } D_{ii} = -1 \end{cases} \qquad \forall\ i = 1, \ldots, m,$$

where $\tilde{A}_i$ is ith row of $\tilde{A}$ and M can be easily estimated. Now we can calculate $h = \max_{i=1,\ldots,m}\{|\beta(i)|\}$.

To reformulate (1-norm SVM) as (P), we first express the variables x and γ as the difference of two nonnegative variables, that is, $x = x^+ - x^-,\ \gamma = \gamma^+ - \gamma^-$. Because of the boundedness of γ, we may enforce $\gamma^+, \gamma^- \in [0, h]$. Define the matrix $A := (De, -De, -D\tilde{A}, D\tilde{A})$ and $\alpha := (\gamma^+;\ \gamma^-;\ x^+;\ x^-) \geq 0$, such that (21) can be equivalently expressed as $A\alpha + e \leq \xi$. Then (1-norm SVM) is equivalent to

$$\begin{aligned} \min_{(\alpha,\xi)} \quad & c^T\alpha + C\,e^T\xi \qquad\qquad \text{(1-norm SVM}'\text{)} \\ \text{s. t.} \quad & A\alpha + e \leq \xi \\ & \alpha,\ \xi \geq 0, \end{aligned}$$

where c is a vector of ones except that the first two entries are zero. Note that Assumption 1.0.1 is satisfied since $\alpha_{\mathcal{B}} = \binom{\gamma^+}{\gamma^-} \leq h\,e$ with $\mathcal{B} = \{1, 2\}$. Therefore, Algorithm 4 is applicable to (1-norm SVM′).

In the following, we compare Algorithm 4 with Newton (Mangasarian, 2006) for solving two types of 1-norm SVMs: the linear 1-norm SVM and the nonlinear kernel 1-norm SVM. The major difference is that the former seeks a separating hyperplane such as (22) while the latter solves for a nonlinear separating surface. According to Mangasarian (2006), the nonlinear kernel SVM can be modeled readily by (1-norm SVM′) with a simple replacement of the data $\tilde{A}$ as follows

$$\tilde{A} \ \longleftarrow\ K(\tilde{A}, \tilde{A}^T)D,$$

where $K(\cdot,\cdot)$ is a kernel function. We discuss the details of the computational experiment in the following paragraphs.

We selected six classification datasets from LIBSVM Data[2] and used them to create both $\tilde{A}$ and $K(\tilde{A},\tilde{A}^T)$. For the linear 1-norm SVM problem, we randomly sampled from each dataset 10,000 or the maximum number of data points, whichever was smaller, to form $\tilde{A}$. For the nonlinear kernel 1-norm SVM problem, we created two sets of $K(\tilde{A},\tilde{A}^T)$ with different sizes. In particular, we randomly sampled 5,000 data points to create the first set of kernel matrices. We then randomly sampled 8,000 or the maximum number of data points, whichever was smaller, to form the second set of kernel matrices. We choose the RBF kernel and followed the procedure as prescribed in Hsu et al. (2003) to prepare $K(\tilde{A},\tilde{A}^T)$. The dataset and the sizes of the data matrices are presented in Table 3. Note that the kernel matrices are square matrices and thus we only show the sizes of their first dimensions.

Table 3: Dimensions of $\tilde{A}$ and $K(\tilde{A},\tilde{A}^T)$

	$\tilde{A}$		$K(\tilde{A},\tilde{A}^T)$	
Dataset	m	n	1st Set, m	2nd Set, m
a6a	10000	122	5000	8000
covtype	10000	54	5000	8000
ijcnn1	10000	22	5000	8000
mushrooms	8124	112	5000	8000
usps	7291	256	5000	7291
w5a	9888	300	5000	8000

In SVMs, C is a user-specified parameter that often takes a set of different values in tuning for the best parameter setting. In our experiment, we tested $C = 1$ and $C = 0.01$ for both the linear SVM and nonlinear kernel SVM.

The stopping criterion for our smoothing technique is that the relative primal-dual gap $r = \frac{p-d}{\max\{1,1/2(|p|+|d|)\}}$ should be smaller than 0.01, where p and d represent the primal and dual objective values, respectively. For Newton, an unconstrained optimization is solved, and its gradient is zero at optimality. Thus we set the stopping criterion for Newton to be when the gradient has norm smaller than $\epsilon = 1e-5$.[3] We also set a time limit of 18,000 seconds for all runs.

Both Algorithm 4 and Newton are implemented in Matlab. For the Newton method, we adopted the code from the author's web site.[4] All computations were performed on the same machine mentioned in the previous subsection.

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

[3]We observe in our computational experiments that if the gradient is not sufficiently small, the resulting solution of the Newton method is infeasible. Therefore, we set a small ϵ.

[4]`http://www.cs.wisc.edu/dmi/svm/lpsvm/`

We summarize the computational results for the linear 1-norm SVM in Tables 4 and 5, corresponding to $C = 1$ and $C = 0.01$, respectively. We compare the smoothing technique and the Newton method in terms of best objective values, CPU times, and optimality. Note that the measures of optimality are different for the two methods and so cannot be compared directly. The smoothing technique was able to solve all instances except for one (*usps*) to within the error tolerance in the given time. The Newton method, however, fails to reduce the gradient to within the error tolerance in three out of six instances when $C = 1$, and in two out of six instances when $C = 0.01$. On the other hand, the Newton method is usually fast when it does converge. Overall, we see that the smoothing technique is efficient and robust compared with the Newton method in solving the linear 1-norm SVM problem. In each table, t signifies that the time limit (18,000 sec.) was exceeded.

Table 4: Linear 1-norm SVM with $C = 1$: comparison of Algorithm 4 (SMOOTH) and the Newton method (NEWTON) in terms of best objective values, CPU times, and measures of optimality.

	Objectives		Times		Optimality	
Data Name	SMOOTH	NEWTON	SMOOTH	NEWTON	Gap	Gradient
a6a	3566.9	87283.5	1191.7	t	1.0e−2	7.5e+1
covtype	5778.1	5916.7	1051.2	t	1.0e−2	6.7e−2
ijcnn1	1741.2	1746.7	305.3	1.6	1.0e−2	1.0e−5
mushrooms	16.2	18.7	1000.3	605.4	1.0e−2	2.0e−6
usps	127.5	37177.5	t	t	1.0e−2	2.1e+2
w5a	324.7	333.2	998.6	2.0	1.0e−2	1.0e−5

Table 5: Linear 1-norm SVM with $C = 0.01$: comparison of Algorithm 4 (SMOOTH) and the Newton method (NEWTON) in terms of best objective values, CPU times, and measures of optimality.

	Objectives		Times		Optimality	
Data Name	SMOOTH	NEWTON	SMOOTH	NEWTON	Gap	Gradient
a6a	43.1	618.8	11.5	t	1.0e−2	2.0e+1
covtype	79.5	81.8	8.1	3757.0	1.0e−2	1.0e−6
ijcnn1	20.8	20.8	3.3	0.3	8.0e−3	2.0e−6
mushrooms	8.5	8.8	9.7	0.5	9.0e−3	2.0e−6
usps	9.6	1436.0	150.7	t	1.0e−2	1.1e+2
w5a	5.7	5.6	6.3	0.1	1.0e−2	1.0e−5

The results for the first set of nonlinear kernel 1-norm SVM problems are summarized in Tables 6 and 7, corresponding to $C = 1$ and $C = 0.01$, respectively. The kernel matrices for this set of problems all have dimension 5000×5000. When $C = 1$, both methods ran out of time on two instances each, but Newton returned much worse objective values when it ran

out of time. The Newton method is faster than the smoothing technique on problems solved by both methods to within the error tolerances. For $C = 0.01$, the smoothing technique solved all problems within the given time while the Newton method ran out of time on four out of the six instances.

Table 6: Nonlinear kernel 1-norm SVM with $C = 1$, first set of data: comparison of Algorithm 4 and the Newton method in terms of best objective values, CPU times, and measures of optimality.

	Objectives		Times		Optimality	
Data Name	SMOOTH	NEWTON	SMOOTH	NEWTON	Gap	Gradient
a6a	1791.2	1791.3	15521.6	3863.6	1.0e−2	0.0e+0
covtype	2844.7	62739.0	10925.6	t	1.0e−2	6.7e−1
ijcnn1	993.2	962.0	t	170.8	1.6e−2	0.0e+0
mushrooms	378.4	376.6	t	4698.5	1.8e−2	0.0e+0
usps	117.5	17053.3	11974.7	t	1.0e−2	1.3e+0
w5a	319.2	314.0	10454.4	1422.4	1.0e−2	0.0e−0

Table 7: Nonlinear kernel 1-norm SVM with $C = 0.01$, first set of data: comparison of Algorithm 4 and the Newton method in terms of best objective values, CPU times, and measures of optimality.

	Objectives		Times		Optimality	
Data Name	SMOOTH	NEWTON	SMOOTH	NEWTON	Gap	Gradient
a6a	23.8	2171.6	1813.8	t	1.0e−2	3.1e−1
covtype	46.0	3499.2	259.4	t	1.0e−2	6.1e−1
ijcnn1	9.8	9.6	3170.3	118.4	1.0e−2	0.0e+0
mushrooms	35.5	1471.2	1379.9	t	1.0e−2	1.1e−1
usps	13.1	2841.6	1428.5	t	1.0e−2	6.9e−1
w5a	3.2	3.1	7253.7	2915.6	1.0e−2	0.0e+0

The computational results for the second set of nonlinear kernel 1-norm SVM are presented in Tables 8 and 9, corresponding to $C = 1$ and $C = 0.01$, respectively. From Table 3, we know that the sizes of these problems are larger than the corresponding linear problems and the first set of nonlinear-kernel problems, and thus they are more difficult to solve. When $C = 1$, the smoothing technique was able to solve only one out of six instances to within the error tolerance in the given amount of time and the Newton method was able to solve two out of six. When $C = 0.01$, the smoothing technique was able to solve four out of six, while the Newton method was able to solve two out of six. In terms of optimality, we find that the optimality gap delivered by the smoothing technique is more consistent (particularly when $C = 0.01$), whereas the Newton method often fails to converge, as indicated by large gradient values.

Table 8: Nonlinear kernel 1-norm SVM with $C = 1$, second set of data: comparison of Algorithm 4 and the Newton method in terms of best objective values, CPU times, and measures of optimality.

	Objectives		Times		Optimality	
Data Name	SMOOTH	NEWTON	SMOOTH	NEWTON	Gap	Gradient
a6a	5533.3	2980.2	t	12341.2	4.3e−1	9.0e−6
covtype	2036.5	22851.5	t	t	8.0e−2	8.4e+1
ijcnn1	3296.2	1542.0	t	172.2	3.6e−1	1.0e−5
mushrooms	732.6	433.6	t	t	5.5e−1	5.9e−5
usps	199.1	7081.2	12782.5	t	1.0e−2	2.9e+2
w5a	1170.1	542.5	t	t	4.8e−1	1.1e−5

Table 9: Nonlinear kernel 1-norm SVM with $C = 0.01$, second set of data: comparison of Algorithm 4 and the Newton method in terms of best objective values, CPU times, and measures of optimality.

	Objectives		Times		Optimality	
Data Name	SMOOTH	NEWTON	SMOOTH	NEWTON	Gap	Gradient
a6a	38.5	1008.5	16413.5	t	1.0e−2	2.1e+1
covtype	24.2	2292.5	2047.8	t	1.0e−2	5.3e+1
ijcnn1	16.3	15.4	t	758.4	2.7e−2	1.0e−5
mushrooms	44.0	1291.0	10520.5	t	1.0e−2	3.5e+1
usps	16.3	1544.0	3741.9	t	1.0e−2	2.4e+2
w5a	5.8	5.6	t	5462.1	1.2e−2	6.0e−6

For both the linear and nonlinear kernel problems, we observe that the smoothing technique is faster with smaller C values, a result that is consistent with the theoretical result in Proposition 3.2.2 because a larger C value corresponds to a larger θ value. Our computational experiments also suggest that the smoothing technique is a favorable choice to solve the 1-norm SVM problem when the parameter C is relatively small. The Newton method is fast when it does converge but is not as robust as the smoothing technique.

6 Conclusion

In this paper, we have developed a first-order smoothing technique for solving (P) and the equivalent problem (NS). To the best of our knowledge, this is the first application of Nesterov's smoothing technique to LPs with unbounded feasible sets. We show that the iteration complexity of this smoothing technique depends on the parameter θ, which arises when bounding the feasible set. We estimate θ as an upper bound on θ^*, the optimal value of (NS). Since a smaller θ means a better iteration complexity, we have designed a strategy

that dynamically updates the value of θ as the algorithm obtains more information about θ^*, resulting in faster convergence. This idea could be extended to other convex nonsmooth problems with unbounded feasible sets.

The smoothing technique is designed for large-scale instances of (P). We have applied the smoothing technique to two problems in machine learning: the linear programming ranking problem and the 1-norm support vector machines. We demonstrate the effectiveness of our method by comparing it with two existing methods: the subgradient method for solving the ranking problem and the Newton method for solving the 1-norm SVM, respectively. Our computational experience indicates that under many circumstances the smoothing technique is a more attractive method because of its balanced efficiency and reliability.

References

A. Asuncion and D. Newman. UCI machine learning repository, 2007. URL `http://archive.ics.uci.edu/ml/`.

K. Ataman. *Learning to rank by maximizing the AUC with linear programming for problems with binary output*. PhD thesis, University of Iowa, 2007.

K. Ataman, W. Street, and Y. Zhang. Learning to rank by maximizing auc with linear programming. In *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pages 123 –129, 2006.

O. Banerjee, L. E. Ghaoui, A. d'Aspremont, and G. Natsoulis. Convex optimization techniques for fitting sparse gaussian graphical models. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 89–96, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: http://doi.acm.org/10.1145/1143844.1143856. URL `http://doi.acm.org/10.1145/1143844.1143856`.

S. Becker, J. Bobin, and E. Candes. Nesta: A fast and accurate first-order method for sparse recovery, 2009. URL `http://www.citebase.org/abstract?id=oai:arXiv.org:0904.3367`.

J. M. Borwein and A. S. Lewis. *Convex analysis and nonlinear optimization*. CMS Books in Mathematics/Ouvrages de Mathématiques de la SMC, 3. Springer, New York, second edition, 2006. ISBN 978-0387-29570-1; 0-387-29570-4.

A. d'Aspremont. Smooth optimization with approximate gradient. *SIAM J. on Optimization*, 19(3):1171–1183, 2008.

G. Fung and O. L. Mangasarian. A feature selection newton method for support vector machine classification. *Computational Optimization and Applications*, 28(2):185–202, July 2004.

A. Gilpin, S. Hoda, J. Peña, and T. Sandholm. Gradient-based algorithms for finding nash equilibria in extensive form games. In *WINE*, pages 57–69, 2007.

S. Hoda, A. Gilpin, and J. Peña. Smoothing techniques for computing nash equilibria of sequential game. Working Paper, Tepper School of Business, Carnegie Mellon Universit, 2007.

C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2003. http://www.csie.ntu.edu.tw/∼cjlin/libsvm/.

G. Lan, Z. Lu, and R. D. C. Monteiro. Primal-dual first-order methods with $o(1/\epsilon)$ iteration-complexity for cone programming. *Mathematical Programming*, pages 1436–4646, 2009. (Online).

O. L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *J. Mach. Learn. Res.*, 7:1517–1530, 2006.

A. Nedic and D. P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic, Boston, MA, 2004.

Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, 2005a.

Y. Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM J. on Optimization*, 16(1):235–249, 2005b.

Y. Nesterov. Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming*, 110(2):245–259, July 2007.

Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120:221–259, 2009. ISSN 0025-5610. URL `http://dx.doi.org/10.1007/s10107-007-0149-x`. 10.1007/s10107-007-0149-x.

T. Zhou, D. Tao, and X. Wu. Nesvm: A fast gradient method for support vector machines. In *IEEE International Conference on Data Mining*, pages 679–688, 2010.

J. Zhu, S. Rosset, T. Hastie, and R. Tibshirani. 1-norm support vector machines. In *Neural Information Processing Systems*, page 16. MIT Press, 2003.