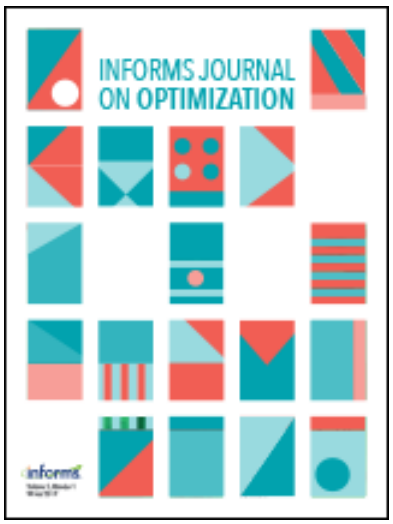


INFORMS Journal on Optimization

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Deterministic Nonsmooth Frank Wolfe Algorithm with Coreset Guarantees

Sathya N. Ravi, Maxwell D. Collins, Vikas Singh

To cite this article:
Sathya N. Ravi, Maxwell D. Collins, Vikas Singh (2019) A Deterministic Nonsmooth Frank Wolfe Algorithm with Coreset Guarantees. INFORMS Journal on Optimization 1(2):120-142. https://doi.org/10.1287/ijoo.2019.0014

http://pubsonline.informs.org/journal/ijoo/

INFORMS JOURNAL ON OPTIMIZATION
Vol. 1, No. 2, Spring 2019, pp. 120–142
ISSN 2575-1484 (print), ISSN 2575-1492 (online)

A Deterministic Nonsmooth Frank Wolfe Algorithm with Coreset Guarantees

Sathya N. Ravi,[a] Maxwell D. Collins,[a] Vikas Singh[a]

[a] Department of Computer Sciences, University of Wisconsin, Madison, Wisconsin 53706

Contact: ravi5@wisc.edu, http://orcid.org/0000-0003-3881-6323 (SNR); mcollins@cs.wisc.edu (MDC); vsingh@cs.wisc.edu (VS)

Received: March 1, 2018
Revised: October 8, 2018
Accepted: December 8, 2018
Published Online in Articles in Advance: May 13, 2019

https://doi.org/10.1287/ijoo.2019.0014

Abstract. We present a new Frank Wolfe type algorithm that is applicable to minimization problems with a nonsmooth convex objective. We provide convergence bounds and show that the scheme yields so-called *coreset* results for various machine learning problems including 1-median, balanced development, sparse principal component analysis, graph cuts, and the ℓ_1-norm-regularized support vector machine. This means that the algorithm provides approximate solutions to these problems in time complexity bounds that are not dependent on the size of the input data. Our framework, motivated by a growing body of work on sublinear algorithms for various data analysis problems, is entirely deterministic and makes no use of smoothing or proximal operators. Apart from these theoretical results, we show experimentally that the algorithm is very practical and in some cases also offers significant computational advantages on large problem instances. We provide an open source implementation that can be adapted for other problems that fit the overall structure.

History: This paper has been accepted for the *INFORMS Journal on Optimization* Special Issue on Machine Learning and Optimization.

Funding: Financial support was received from the Division of Computing and Communication Foundations [Grant 1320755], the National Institutes of Health [Grant U54 AI117924], and the Division of Information and Intelligent Systems [Grant 1252725].

Keywords: nonsmooth convex optimization • Frank Wolfe methods • sublinear algorithms

1. Introduction

The impact of numerical optimization on modern data analysis has been quite significant. Today, these methods lie at the heart of most statistical machine learning applications in domains spanning genomics Banerjee et al. (2006), finance Pennanen (2012), and medicine Liu et al. (2009). The expanding scope of these applications (and the complexity of the associated data) has continued to raise the expectations of the efficiency profile of the underlying algorithms that drive the analyses modules. For instance, the development of "low-order" polynomial time algorithms has long been a central focus of algorithmic research. The availability of a (near) linear time algorithm for a problem was considered a gold standard since any algorithm's runtime should, at a minimum, include the time it takes to evaluate the input data in its entirety. But as applications that generate extremely large data sets become more prevalent, we encounter many practical scenarios where even a procedure that takes only linear time to process the data may be considered impractical (Sarlos 2006, Clarkson and Woodruff 2015). Within the last decade or so, efforts to understand whether the standard notions of an efficient algorithm are sufficient and/or whether linear time algorithms are good enough for various turnkey applications has led to the study of so-called "sublinear" algorithms and a number of interesting results have emerged (see Rudelson and Vershynin 2007, Clarkson et al. 2012, Bachem et al. 2018, and Baykal et al. 2018).

Sublinear computation is based upon the premise that a fast (albeit approximate) solution may, in some cases, be preferable to the optimal solution obtained at a higher computational or financial cost. The strategy is a good fit in at least two different regimes: (a) operations on streaming data where an algorithm must run in real time (an inexact but prompt answer may suffice) (Braverman et al. 2016), or (b) where the appropriate type of data (for the task) is scarce or unavailable at sample sizes necessary, such as distribution testing (Acharya et al. 2015, Daskalakis et al. 2018). In either case, an important feature of most sublinear algorithms is the use of an approximated version of the decision version of the original problem. Many of the technical results for specific problems/tasks have focused on deriving so-called "property testing" schemes as a means to establish what an algorithm can be expected to accomplish without reading through all of the data. This line of work has led to many fast sublinear algorithms for numerical linear algebra problems, including matrix multiplication (Drineas et al. 2006), computing spectral norms and leading singular vectors of the data matrix

(Drineas et al. 2006 among others), and has been deployed in settings where the full data may have a large memory footprint. More recently, several authors have even shown how to extend this idea for solving convex optimization problems. In particular, the work by Clarkson (2010) that motivates our paper obtains sublinear algorithms for a variety of problems where the feasible set is the unit simplex. Next, we briefly describe this framework and its relationship to coresets, an idea often used in the design and analysis of algorithms in computational geometry (see Chan 2018).

In an important result several years ago Clarkson (2010), the author showed that sublinear algorithms can be designed for a variety of problems using the framework of the well-known Frank Wolfe (FW) algorithm (Frank and Wolfe 1956). Recall that FW algorithms can be used to solve smooth convex optimization problems when the feasible set is compact. Using this framework, Clarkson (2010) unified the analysis of various problems in machine learning and statistics, including regression, boosting, and density mixture estimation via reformulations, as optimization problems with a smooth convex objective function and where the unit simplex was the feasible set. This setup was then shown to provide immediate results to important theoretical and practical issues, including sample complexity bounds and faster algorithms for support vector machines (SVMs) and approximation bounds for boosting procedures. This result raises an interesting question: What properties of FW algorithms enable one to design fast algorithms with provable guarantees? Clarkson (2010) addressed this question by drawing a contrast between FW algorithms and a (widely used) alternative strategy, i.e., projected gradient type algorithms. Specifically, instead of a quadratic function, in FW algorithms, we optimize a linear function over the feasible set, which often yields a better per-iteration complexity for many problems (further, the iterates always remain feasible). It turns out that these properties are particularly useful from a optimization point of view since one can frequently obtain pipelines with significantly better memory complexity (Garber and Meshi 2016, Gidel et al. 2016), as we discuss next.

An interesting consequence of the cited work was a result showing a nice relationship of FW-type schemes to a concept predominantly used within computational geometry known as *coresets* (Agarwal et al. 2005). Intuitively, a coreset (typically defined in the context of a problem statement) is a subset of the input data on which an algorithm with provable approximation guarantees for the problem at hand can be obtained, and which has a runtime/memory complexity that is, in some sense, independent of (or only loosely dependent on) the size of the data set (Huggins et al. 2016). A key practical consequence of the analysis in Clarkson et al. (2012) was that coresets for SVMs derived using the proposed procedure were tighter than all earlier results (Tsang et al. 2005). For other machine learning problems that can be solved using convex optimization included in Clarkson et al. (2012), the analysis typically allows bounding the number of nonzero elements in the decision variables (independent of the total number of such variables). Clearly, these are very useful properties, but we see that the setup in Clarkson et al. (2012) requires that the objective function of the optimization problem be differentiable. This is somewhat restrictive for machine learning and computer vision applications, where various nonsmooth regularizers are ubiquitous and serve to impose structural or statistical requirements on the optimal solution (Bach et al. 2012). The overarching goal of our paper is to obtain an algorithm (very similar to the standard FW procedure) that makes no such assumption. Incidentally, we are also able to obtain coreset-type results for a much broader class of problems (that involve nonsmooth objective functions), providing sublinear algorithms for several of these applications and sensible heuristics for others.

1.1. Overview of This Work and Related Results

We first define the problem considered here and give an overview of some of the related works. Let f be a convex and real-valued function (but not necessarily continuously differentiable) over a compact convex domain D. We study a specific class of finite dimensional optimization problems expressed as

$$\min_{x \in D} f(x). \tag{1}$$

Note that we can assume f to be L-Lipschitz for some $0 \le L < \infty$ since D is compact. Our central result gives a new convergence bound for an algorithm derived from a scheme first outlined in White (1993) in the early 1990s, but that seems to have been utilized only sparingly in the literature since. At a high level, our deterministic method generates a sequence of iterates x_k for $k = \{0, 1, 2, \ldots\}$ that provably converges in the primal-dual gap, which implies convergence in the objective function value as well since the problem is convex, by assumption. Specifically, if x_* is a solution to Equation (1), we show that $f(x_k) \le f(x_*) + O(1/\sqrt{k})$, thereby providing an a priori bound on the suboptimality of each iterate. We can restate this bound, for any choice of $\epsilon > 0$ and

$$f(x_k) \le f(x_*) + \epsilon \text{ for any } k \ge K, \tag{2}$$

for some $K \in O(1/\epsilon^2)$.

1.1.1. Relevance. A wide range of problems can be expressed in the form shown in Equation (1), and for some of these problems, bounds as in Equation (2) will also yield a coreset result. As briefly described in the previous section, within a coreset-based algorithm, we can perform training on a very large number of examples while requiring computational resources that depend on the intrinsic hardness of the problem regardless of the amount of data collected. We note that although a subsampling heuristic may show this behavior for some data sets (and for some objective functions), in general, a coreset (if it exists) will typically yield a stronger approximation than ordinary subsampling. For example, coreset results have been used for k-means and k-median clustering (Har-Peled and Kushal 2005), subspace approximation (Feldman et al. 2010), SVMs and its variants (Tsang et al. 2005, Har-Peled et al. 2007, Nathan and Raghvendra 2014), and robotics (Feldman et al. 2012, Balcan et al. 2013). For several problems we discuss here, convergence bounds on the optimization model show that a solution x_K can be found with no more than K nonzero entries, and these nonzeros correspond to the choice of a coreset as mentioned earlier. Another consequence of such a result is that the coreset bound will be deterministic, giving a tight bound on the actual approximation error, as opposed to a bound on the expected approximation error. For some problems, our results also translate well to practice, yielding fast implementations discussed in the experimental section.

1.1.2. Relation to Existing Work. A small set of results pertaining to nonsmooth FW algorithms (and our proposed ideas) have been reported in the literature in the last few years. At the high level, these approaches correspond to three different flavors:

- *Noisy gradients*: Jaggi (2013) considers the case where the FW method uses an inexact gradient such that the linear subproblems provide a solution to the exact problem within a bounded error but assume that the objective function is smooth.
- *Dualizing*: Jaggi (2011) presents a FW dual for nonsmooth functions employing the subgradients of the objective. This can be used to generate a "certificate" of optimality of an approximate solution via the primal-dual gap but the algorithms do not deal with the nonsmoothness directly. In particular, for the applications studied in Jaggi (2011), the nonsmooth parts in the objective function are dualized instead and treated as constraints. This approach becomes inefficient since the subproblems are complicated for many other important applications that are considered here.
- *Smoothing*: Hazan and Kale (2012) present an algorithm for minimization of nonsmooth functions with bounded regret that applies the FW algorithm to a randomly smoothed objective, instead providing probabilistic bounds. This approach closely addresses the setting that we consider; however, it has a practical bottleneck. Observe that (at each iteration) one has to make many queries to the zeroth order oracle of the objective function in order to compute the gradient of the smoothed (approximate) function, which is often expensive when the function depends on the number of data points (may be quite large in many problems). We do observe this behavior in our experimental evaluations in Section 5.

We will point out other related works in the relevant sections that follow. Overall, we see that much of the existing literature has approached this problem either via an implicit randomized smoothing (Hazan and Kale 2012) or by using proximal functions (Argyriou et al. 2014, Pierucci et al. 2014) to yield suitable gradients and provide convergence results for the smoothed objective. Contemporary to our work, Cheung and Li (2017) developed a nonsmooth FW algorithm for minimizing ℓ_1-regularized smooth convex functions over the trace norm ball. The update directions in Cheung and Li (2017) are obtained by considering all affine functions over the ℓ_∞ ball centered around the current iterate with the radius chosen appropriately to ensure convergence. Our approach can be seen as a generalization of Cheung and Li (2017) in that we show that if we choose a neighborhood that is problem specific (instead of ℓ_∞ ball), we can show both convergence and coreset results. To that end, we will discuss a general class of optimization problems for which our algorithm can be viable in Section 4.6. Interestingly, subproblems that arise in our algorithm can also be solved efficiently in theory and practice for all the applications discussed in Cheung and Li (2017) as we will see shortly in Section 4.

1.1.3. Summary of Our Contributions. In Section 2, we discuss why simply using a subgradient is insufficient to prove convergence. Then, we introduce the basic concepts that are used in our algorithm and analysis. The starting point of our development is a specific algorithm mentioned in White (1993), which is of the same general form as Algorithm 1 of our paper in Section 3. White (1993) includes a result equivalent to the approximate weak duality that we describe later. In Section 3, we derive an a priori convergence result for a nonsmooth generalization of the FW method, which also yields a deterministic bound on the approximation error and the size of the constructed coreset. We use a much more general construction for the approximate subdifferential $T(x, \epsilon)$ and show how this can yield novel coreset results analogous to those shown for the FW

method in the smooth case in Clarkson (2010). Using these results, we analyze many important problems in statistics and machine learning in Section 4. Although the algorithm we present may not be a silver bullet for arbitrary nonsmooth problems (which may have specialized algorithms), in several settings, the results do have practical import, as discussed in Section 5. For instance, we show an example where our algorithm enables solving very large problem instances on a single desktop where other reported approaches have deployed distributed optimization schemes on a cluster. On the theoretical side, a useful result is that our scheme can produce coresets with hard approximation bounds.

2. Preliminary Concepts

To introduce our algorithm and the corresponding convergence analysis, we start with some basic definitions. Recall that a fundamental tool for optimization of nonsmooth functions is the subdifferential. It is well known that it is possible to design algorithms with $O(1/\epsilon^2)$ convergence rate for nonsmooth problems (see Lan et al. 2012 and Nesterov 2013). Unfortunately, utilizing just the subdifferential turns out to be insufficient to produce the necessary convergence bounds for our analysis. For instance, in a recent paper, Nesterov (2018) constructively showed that simply replacing the gradient by a subgradient in a FW algorithm is incapable of ensuring convergence—a simple two-dimensional example demonstrated that the FW algorithm does not converge to the optimal solution for any number of iterations. To our knowledge, there are no clear strategies to fix this limitation. As mentioned earlier, one often uses smoothing techniques as a workaround, where an auxiliary function is constructed that is optimized instead (Nesterov 2005, Pierucci et al. 2014). Although this approach has had significant practical impact, it involves the selection of a proximal function. But there are no general recipes to choose the proximal function for a given model and it is often designed based on the specific problem at hand.

We observe that the convergence analysis of FW-type methods relies on the boundedness of an important quantity called the "curvature" constant:

$$C_f^{\text{exact}} := \sup_{\substack{x,s\in D\\ \alpha\in[0,1]\\ y=x+\alpha(s-x)}} \left(\min_{d\in\partial f(x)} \frac{1}{\alpha^2}\left(f(y)-f(x)-\langle y-x,d\rangle\right)\right). \tag{3}$$

It describes how well the first-order information from the approximate subdifferential globally describes the function (when f is smooth, d is simply the gradient of f). It is this quantity that becomes unbounded even for simple nonsmooth functions, making the subsequent analysis difficult. To make this point clear, we now show a simple example of a nonsmooth function for which this quantity is not bounded, though in this "easy" case we can prove convergence with more direct means.

Example 1. Let $f(x) = |x|$ over $D = [-1, 1]$. For any $x \in (0, \frac{1}{2})$, let $s = -1 + x$ and $\alpha = 2x$. Accordingly,

$$y = x + \alpha(s - x) = x + 2x(-1 + x - x) = -x. \tag{4}$$

By definition,

$$C_f^{\text{exact}} \geq \min_{d\in\partial f(x)} \frac{1}{\alpha^2}\left(f(y)-f(x)-\langle y-x,d\rangle\right) = \min_{d\in\partial f(x)} \frac{1}{4x^2}\left(f(-x)-f(x)+\langle 2x,d\rangle\right). \tag{5}$$

Because f is differentiable at x, $\partial f(x) = \{1\}$,

$$C_f^{\text{exact}} \geq \frac{1}{4x^2}(|-x|-|x|+2x) = \frac{1}{2x}. \tag{6}$$

Hence we have that $\lim_{x\to 0^+} \frac{1}{2x} = +\infty$, showing that we cannot obtain an upper bound C_f^{exact} for all $x \in (0, \frac{1}{2})$. This example shows that linear approximations based on ordinary subgradients can be arbitrarily poor in the neighborhood of nondifferentiable points.

2.1. Basic Idea

Intuitively, the subdifferential is a discontinuous multifunction (Robinson 2012) and provides an incomplete description of the curvature C_f of the function in the neighborhood of the nonsmooth points. After making a few technical adjustments, it turns out that we can work around this issue by making use of approximate subdifferentials. We deal with two constructions that yield approximate subdifferentials. Using these definitions,

we will give the formal statement of our convergence theorem in the next section. The first is the *ϵ-subdifferential* (Hiriart-Urruty and Lemaréchal 1993), which is defined at each point x in the domain of a function f as

$$\partial_\epsilon f(x) := \{d \mid f(y) \geq f(x) + \langle d, y - x\rangle - \epsilon \text{ for all } y\}. \tag{7}$$

Notice that the exact subdifferential is recovered when ϵ is zero, $\partial_0 f(x) = \partial f(x)$.

Although the ϵ-subdifferential indeed provides the theoretical properties for our convergence bounds, in practical cases, it requires us to work with carefully chosen problem-specific subsets of $\partial_\epsilon f(x)$. A successive construction, which we present later, produces approximate subdifferentials that have nicer computational properties while preserving the convergence bounds. In this approximation, we must take the subdifferentials over a neighborhood. This idea is summarized next.

Let N be an arbitrary mapping from x to neighborhoods around x such that $N(x,\epsilon) \to \{x\}$ as $\epsilon \to 0$. Formally, we assume that $x \in N(x,\epsilon)$ and there exists some constant R such that $N(x,\epsilon) \subseteq \mathcal{B}(x, R\epsilon)$ for all $\epsilon > 0$. The notation $\mathcal{B}(x,r)$ is the open ball around x of radius r. Assume without loss of generality that $R = 1$. Let T be defined as

$$T(x,\epsilon) := \begin{cases} \{\nabla f(x)\} & \text{if } f \text{ is differentiable on } N(x,\epsilon), \\ \bigcup_{u \in N(x,\epsilon)} \partial f(u) & \text{otherwise.} \end{cases} \tag{8}$$

Here T provides a set of approximate subgradients of f at x. Indeed, if f is L-Lipschitz, we have $T(x, \frac{\epsilon}{2L}) \subseteq \bigcup_{u \in \mathcal{B}(x, \frac{\epsilon}{2L})} \partial f(u) \subseteq \partial_\epsilon f(x)$ by theorem 8.4.4 of Robinson (2012). It turns out that with these constructions, we will shortly obtain a deterministic nonsmooth FW algorithm with accompanying convergence results.

3. Convergence Results

Using the foregoing concepts, we can adapt the procedure in White (1993) with a few technical modifications as shown in Algorithm 1.

Algorithm 1 (Conditional Subgradient (Based on Ideas Introduced in White 1993))

Pick an arbitrary starting point $x_0 \in D$, where D is the compact convex feasible set.
for $k = 0, 1, 2, \ldots$ **do**
Let $\alpha_k := \frac{2}{k+2}$ and $\epsilon_k := \sqrt{\alpha_k}$

$$s \in \arg\min_{z \in D} \max_{d \in T(x, \epsilon_k)} \langle z - x, d\rangle \tag{9}$$

Update $x_{k+1} := x_k + \alpha_k(s - x_k)$
end for

Although both Algorithm 1 and the one proposed in White (1993) are similar in spirit, White (1993) requires us to do exact line search, which limits the applicability in our experimental settings. Moreover, the convergence result in White (1993) is asymptotic (as $k \to \infty$), whereas we provide explicit convergence rate both in primal and in duality gap. Our analysis, arguably, gives a better understanding of the algorithm using the affine invariant curvature concept.

To simplify the presentation, we will assume that the subproblems in Equation (9) are efficiently solvable both in theory and in practice. In general, this is true when $T(x,\epsilon)$ is a polyhedron since any point $d \in T(x,\epsilon)$ can be written as a convex combination of extreme points and extreme rays. Although it might seem like this severely limits the applicability of the proposed approach (to a point where it is hard to imagine when it can be applied other than when the subdifferentials are polytopes), we identify a general class of problems in Section 4.6 for which the subproblems can be solved efficiently in theory even when the data set size is large. We also show shortly that for various statistical machine learning problems, we can solve Equation (9) far more efficiently even when $T(x,\epsilon)$ is not a polyhedron, making the overall algorithm attractive in any case. With this assumption, we may generalize the curvature constant which plays a key role in our convergence analysis:

$$C_f(\epsilon) := \sup_{\substack{x,s \in D \\ \alpha \in [0,1] \\ y = x + \alpha(s-x)}} \min_{d \in T(x,\epsilon)} \frac{1}{\alpha^2}\left(f(y) - f(x) - \langle y - x, d\rangle\right). \tag{10}$$

Importantly, we use approximate subgradients instead of a gradient as in Clarkson (2010), where the value of C_f depends on ϵ. We can choose an ϵ varying with iterations in such a way as to guarantee convergence

although in practical problems, we see that $C_f \in O(1/\epsilon)$. In any case, note that the complexity of the algorithm does not depend on the input data, that is, we can choose ϵ to be very small such that optimizing linear functions over $T(x,\epsilon)$ is tractable in practice. Concretely, input data in our settings correspond to either training examples (or features) depending on the application. In particular, when the decision variables are defined over examples, and the subdifferential can be computed with only the nonzero coordinates of the decision variables, our algorithm ends up being sublinear in the number of training examples (or features) depending on the application. We will see this in detail in the later sections for specific problems that are also empirically verified in Section 5. Our central convergence theorem, stated in terms of this constant, appears next.

Theorem 1. *Suppose f is L-Lipschitz and $C_f(\epsilon) \leq \frac{D_f}{\epsilon}$ for some constant D_f for any $\epsilon \leq 1$. Then Algorithm 1 generates a sequence of iterates x_k, such that for the kth iteration:*

$$f(x_k) - f(x_*) \leq \frac{2^{5/2}L + 2^{3/2}D_f}{\sqrt{k+2}}, \tag{11}$$

where x_* is the optimal solution to the problem in Equation (1).

3.1. Remarks

As the iteration count k increases, the denominator in this upper bound increases. Since the numerator consists of constants depending only on the definition of f and D, this will approach 0 as $k \to \infty$. It is important to notice that although this theorem provides a general convergence bound (which is useful), it does not alone prove that Algorithm 1 will produce a coreset and consequently, will be sublinear. Indeed, it is a known result for smooth functions. Shortly, in example cases covered in Section 4, we show the other key component of the proposed algorithm: a bound on the number of nonzeros for any solution s of the subproblems in Equation (9). The key results here will depend on D and f, and we will seek to show that the number of nonzero entries in s will be $O(1)$ with respect to (w.r.t.) the overall size of the problem. This in turn guarantees that x_k will have no more than $O(k)$ nonzeros. Combined with the previous convergence bound, this in turn will demonstrate that a sparse ϵ-approximate solution can be found with $O(1/\epsilon^2)$ nonzeros, which does not depend on the problem size (e.g., number of samples/examples or dimensions). When x is a vector of the examples in a machine learning problem, the union of nonzeros for each s found by Algorithm 1 will constitute a coreset. Theorem 1 can be used as a general framework that can be extended to show a coreset result for any nonsmooth problem for which a sparsity bound on s can in turn be shown, yielding a valuable tool for sublinear algorithm design.

3.2. Ingredients

The proof of Theorem 1 relies on a bound in the improvement in the objective at each iteration in terms of $C_f(\epsilon)$ (in Equation (10)) and the duality gap between the primal objective at x and a nonsmooth modification of the Wolfe dual. We define this dual ω of f as

$$\omega(x,\epsilon) := \min_{z \in D} \max_{d \in \partial_\epsilon f(x)} f(x) + \langle z - x, d \rangle. \tag{12}$$

This dual gives us a property we call *approximate weak duality*. Up to ϵ, this dual is at all points less than the minimum value of the primal objective at all points.

Lemma 1 (Approximate Weak Duality). *We can bound the approximate weak duality $\omega(x,\epsilon)$ as, $\omega(x,\epsilon) \leq f(y) + \epsilon$ for all $x, y \in D$.*

Proof. Take any $x, y \in D$. Due to the minimization in Equation (12),

$$\omega(x,\epsilon) \leq \max_{d \in \partial_\epsilon f(x)} f(x) + \langle y - x, d \rangle. \tag{13}$$

By the definition of the ϵ-subdifferential, for any $d \in \partial_\epsilon f(x)$, including the one chosen by the maximization,

$$f(x) + \langle y - x, d \rangle \leq f(y) + \epsilon, \tag{14}$$

which gives us the desired result. □

Denote the primal-dual gap at x by

$$g(x,\epsilon) := f(x) - \omega(x,\epsilon) = \max_{y \in D} \min_{d \in \partial_\epsilon f(x)} \langle x - y, d \rangle. \tag{15}$$

Then, by Lemma 1,

$$g(x,\epsilon) \geq f(x) - f(x_*) - \epsilon \geq -\epsilon. \tag{16}$$

The combination of the primal-dual gap and the curvature constant C_f is used to show per-iteration improvement on the objective. This is then used to show the a priori convergence result in Theorem 1. We see that at each step of Algorithm 1, the objective at x_{k+1} improves upon the objective at the previous iterate x_k by a term proportional to the primal-dual gap, up to a curvature error term with C_f. This is stated in the following result.

Theorem 2. *Fix $\epsilon' = 2L\epsilon_k$. For any step $x_{k+1} := x_k + \alpha(s - x_k)$, with arbitrary step size $\alpha \in [0,1]$ and $s \in D$ chosen as in Equation (9), it holds that:*

$$f(x_{k+1}) \leq f(x_k) - \alpha g(x_k, \epsilon') + \alpha^2 C_f(\epsilon_k). \tag{17}$$

Proof. Write $x := x_k, y := x_{k+1} = x + \alpha(s - x)$, and $\epsilon := \epsilon_k$. From the definition of C_f,

$$f(y) \leq f(x) + \left(\max_{d \in T(x,\epsilon)} \langle y - x, d\rangle\right) + \alpha^2 C_f(\epsilon) = f(x) + \alpha\left(\max_{d \in T(x,\epsilon)} \langle s - x, d\rangle\right) + \alpha^2 C_f(\epsilon).$$

By choice of s in Equation (9),

$$f(y) \leq f(x) + \alpha\left(\min_{y \in D} \max_{d \in T(x,\epsilon)} \langle y - x, d\rangle\right) + \alpha^2 C_f(\epsilon).$$

Now, using the fact that $T(x, \frac{\epsilon}{2L}) \subseteq \partial_\epsilon f(x)$,

$$f(y) \leq f(x) + \alpha\left(\min_{y \in D} \max_{d \in \partial_{\epsilon'} f(x)} \langle y - x, d\rangle\right) + \alpha^2 C_f(\epsilon) = f(x) - \alpha\left(\max_{y \in D} \min_{d \in \partial_{\epsilon'} f(x)} \langle x - y, d\rangle\right) + \alpha^2 C_f(\epsilon).$$

By definition in Equation (15), we have that $f(y) \leq f(x) - \alpha g(x, \epsilon') + \alpha^2 C_f(\epsilon)$ as desired. □

Now, we are ready to prove Theorem 1.

Proof of Theorem 1. Finally, we prove the main theorem by inductively applying the per-step improvement in Equation (17) to find the a priori bound in terms of k.

Let $h(x) = f(x) - f(x_*)$. Using Theorem 2, we first restate the stepwise improvement in the objective as follows:

$$h(x_{k+1}) \leq h(x_k) - \alpha_k g(x_k, 2L\epsilon_k) + \alpha_k^2 C_f(\epsilon_k). \tag{18}$$

By Lemma 1,

$$h(x_{k+1}) \leq h(x_k) - \alpha_k(h(x_k) - 2L\epsilon_k) + \alpha_k^2 C_f(\epsilon_k) \leq (1 - \alpha_k)h(x_k) + 2L\alpha_k\epsilon_k + \frac{\alpha_k^2}{\epsilon_k} D_f. \tag{19}$$

Now, if we substitute the constant $E := 2L + D_f$, and the choice $\epsilon_k = \sqrt{\alpha_k}$ as in the algorithm,

$$h(x_{k+1}) \leq (1 - \alpha_k)h(x_k) + \alpha_k^{3/2} E.$$

(We will later see in the experiments how this choice of ϵ_k makes the algorithm practically useful.) Using this notation to rewrite the theorem, we seek to show $h(x_k) \leq \frac{2^{3/2}E}{\sqrt{k+2}} = 2E\sqrt{\alpha_k}$. We prove this by induction. The base case comes from applying Equation (17) with $k = 0$ and $\alpha_k = 1$ to get $h(x_1) \leq E \leq \frac{2^{3/2}E}{\sqrt{3}}$. For $k \geq 1$, making the induction step,

$$\begin{aligned} h(x_{k+1}) &\leq (1 - \alpha_k)(2E\sqrt{\alpha_k}) + \alpha_k^{3/2} E \\ &= \left(1 - \frac{2}{k+2}\right)\frac{2^{3/2}E}{\sqrt{k+2}} + \frac{2^{3/2}}{(k+2)^{3/2}} E \\ &= 2^{3/2}E\sqrt{k+2}\left(\frac{1}{k+2} - \frac{1}{(k+2)^2}\right). \end{aligned}$$

Now, we make an elementary calculation:

$$\left(\frac{1}{k+2}-\frac{1}{(k+2)^2}\right)=\frac{1}{k+2}\frac{k+2-1}{k+2}\leq\frac{1}{k+2}\frac{k+2}{k+3}=\frac{1}{k+3}. \tag{20}$$

Using Equation (20), we can get the desired result as

$$h(x_{k+1})\leq 2^{3/2}E\frac{\sqrt{k+2}}{k+3}\leq 2^{3/2}E\frac{\sqrt{k+3}}{k+3}=\frac{2^{3/2}E}{\sqrt{k+3}}. \quad \square$$

3.3. Comments

A key intuition about the algorithm can be gained from the proof step in Equation (19). This inequality implies that the improvement in the objective for each step has a bounded error with two parts. First is an error linear in α_k that comes from our use of an approximate subgradient rather than an exact subgradient. This error is proportional to ϵ_k. Second, we have a curvature error that arises from the nonlinearity of our objective function and is instead inversely proportional to ϵ_k. By choosing a decaying ϵ_k such that $\frac{1}{\epsilon_k}\in\Theta(1/\sqrt{\alpha_k})$, we send both error terms to 0 as $k\to\infty$.

4. Approximable Nonsmooth Problems

In this section, we demonstrate how the foregoing ideas can be applied and made more specific, on a case-by-case basis, for various problems arising in statistical machine learning. Since many of the steps and quantities in the algorithm and theorems described in the previous section depend on details of the individual problem, we use these examples to show the potential value of our results. In this section, each problem we present yields a choice of objective function f and feasible region D. For these choices, we then describe the approximate subdifferentials and the resulting subproblems to determine the step direction. We show that the antecedents of Theorem 1 are satisfied for a number of machine learning problems and provide $O(1/\epsilon)$ bounds for $C_f(\epsilon)$. A problem-specific bound on C_f is the final component of establishing an iteration bound for that problem. It guarantees that our scheme for selecting ϵ_k will provide convergence, and that we have finite values for the numerator in Equation (11). Therefore, the quotient will approach 0 in the limit.

We first analyze the supremum of linear functions with the goal of getting an expression that bounds C_f. With this in hand, we will present a formulation of ℓ_1 SVM that is suitable for our analysis. In particular, we show C_f bounds using the result from piecewise linear functions and then proceed to discuss how the subproblems in Equation (9) can be efficiently solved. We discuss the tractability and provide tight coreset bounds for this problem. Later, we analyze the multiway graph cuts model by adapting the techniques used in the ℓ_1 SVM problem, and finally, we also describe how the analysis extends to the 1-median problem, sparse principal component analysis (PCA), and balanced development.

4.1. Supremum of Linear Functions

We can define a piecewise linear convex function on $\mathbb{R}^n$ by

$$f(x):=\max_{i=1,\ldots,p} a_i^T x+b_i. \tag{21}$$

If we let $A\in\mathbb{R}^{p\times n}$ be the matrix such that row i is equal to a_i, and $b\in\mathbb{R}^p$ be the vector that has elements b_i. We know that f is Lipschitz-smooth with constant equal to the operator norm $\|A\|$ and $f(x)\equiv\hat{f}(Ax+b)$ for

$$\hat{f}(\hat{x})=\max_{i=1,\ldots,p}\hat{x}_i \Longrightarrow f(x):=\hat{f}(Ax+b). \tag{22}$$

The subdifferentials of $\hat{f}$ are given by

$$\partial\hat{f}(\hat{x})=\left\{e_i \;\middle|\; \left(\max_{j=1,\ldots,p}\hat{x}_j\right)=\hat{x}_i\right\} \text{ where } e_i \text{ is the } i\text{th basis vector.} \tag{23}$$

For the purposes of calculating approximate subdifferentials $\hat{T}$ and T of $\hat{f}$ and f, respectively (defined in Section 2), we use the neighborhoods,

$$\hat{N}(\hat{x},\epsilon) = \{\hat{u} \mid \|\hat{u}-\hat{x}\|_\infty \le \epsilon\}, \quad N(x,\epsilon) = \left\{u \mid \|Au-Ax\|_\infty \le \epsilon, \text{ and } u-x \in \operatorname{im}\left(A^T\right)\right\}, \tag{24}$$

where A^T is the transpose of A and $\operatorname{im}(A^T)$ is the row space of A.

Theorem 3. *Under the setting of Equations* (21), (22), *and* (24), *define*

$$V(\hat{x},\epsilon) = \left\{e_i \;\middle|\; \hat{x}_i \ge \left(\max_j \hat{x}_j\right) - 2\epsilon\right\}, \tag{25}$$

and let CoS denote the convex hull of the set S. Then, the approximate subdifferential is given by $\hat{T}(\hat{x},\epsilon) = \mathrm{Co}V(\hat{x},\epsilon)$.

Proof. Note that both sides will be a singleton set equal to $\{\nabla\hat{f}(\hat{x})\}$ if $\hat{f}$ is differentiable on $\hat{N}(\hat{x},\epsilon)$. We therefore prove only the case that $\hat{f}$ is nondifferentiable somewhere in the neighborhood.

Forward Direction: $\hat{T}(\hat{x},\epsilon) \subseteq \mathrm{Co}V(\hat{x},\epsilon)$.

Take any $\hat{u} \in \hat{N}(\hat{x},\epsilon)$. We have:

$$\partial\hat{f}(\hat{u}) = \mathrm{Co}\left\{e_i \;\middle|\; \hat{u}_i = \left(\max_j \hat{u}_j\right)\right\}. \tag{26}$$

Now take any i such that $e_i \in \operatorname{vert}(\partial\hat{f}(\hat{u}))$ where $\operatorname{vert}(S)$ denotes the vertices of a polyhedral set S. If S is a finite set of linearly independent points, then $\operatorname{vert}(\mathrm{Co}S) = S$ [which in this case is the set of basis vectors in Equation (26)]. Because $\|\hat{u}-\hat{x}\|_\infty \le \epsilon$,

$$\hat{x}_i \ge \hat{u}_i - \epsilon = \left(\max_j \hat{u}_j\right) - \epsilon \ge \left(\max_j \hat{x}_j\right) - 2\epsilon. \tag{27}$$

Therefore,

$$\operatorname{vert}(\partial\hat{f}(\hat{u})) \subseteq V(\hat{x},\epsilon) \Longrightarrow \partial\hat{f}(\hat{u}) \subseteq \mathrm{Co}V(\hat{x},\epsilon). \tag{28}$$

Since Equation (28) is true for all $u \in \hat{N}(\hat{x},\epsilon)$, we see that $\hat{T}(\hat{x},\epsilon) \subseteq \mathrm{Co}V(\hat{x},\epsilon)$.

Reverse Direction: $\hat{T}(\hat{x},\epsilon) \supseteq \mathrm{Co}V(\hat{x},\epsilon)$

Let $\hat{u}$ be the vector defined elementwise by

$$\hat{u}_i = \begin{cases} \left(\max_j \hat{x}_j\right) - \epsilon & \text{if } \hat{x}_i \ge \left(\max_j \hat{x}_j\right) - 2\epsilon, \\ \hat{x}_i & \text{otherwise.} \end{cases} \tag{29}$$

If we take some i such that $\hat{x}_i \ge (\max_j \hat{x}_j) - 2\epsilon$, then

$$\hat{u}_i - \hat{x}_i = \left(\max_j \hat{x}_j\right) - \epsilon - \hat{x}_i \le \left(\max_j \hat{x}_j\right) - \epsilon - \left(\left(\max_j \hat{x}_j\right) - 2\epsilon\right) = \epsilon. \tag{30}$$

Similarly, it also holds that

$$\hat{x}_i - \hat{u}_i \le \left(\max_j \hat{x}_j\right) - \hat{u}_i = \left(\max_j \hat{x}_j\right) - \left(\left(\max_j \hat{x}_j\right) - \epsilon\right) = \epsilon. \tag{31}$$

And $\hat{u}_i = \hat{x}_i$ for the "otherwise" case. Therefore $\|\hat{u}_i - \hat{x}_i\|_\infty \le \epsilon$ and $\hat{u}_i \in \hat{N}(\hat{x},\epsilon)$. Furthermore, note that clearly $\hat{f}(\hat{u}) = \max_j \hat{x}_j - \epsilon$, and for any i such that

$$\hat{x}_i < \left(\max_j \hat{x}_j\right) - 2\epsilon, \tag{32}$$

we have that, for all $\epsilon > 0$,

$$\hat{x}_i < \left(\max_j \hat{x}_j\right) - \epsilon = \hat{f}(\hat{u}). \tag{33}$$

Accordingly,

$$\text{vert}(\partial\hat{f}(\hat{u})) = \hat{V}(\hat{x},\epsilon) \Longrightarrow \hat{T}(\hat{x},\epsilon) \supseteq \partial\hat{f}(\hat{u}) = \text{Co}\hat{V}(\hat{x},\epsilon). \quad \square \tag{34}$$

Lemma 2. *Under the setting of Equations* (21), (22), *and* (24), $T(x,\epsilon) = A^T\hat{T}(Ax+b,\epsilon)$.

Proof. This follows from the fact that $\partial f(u) = A^T\partial\hat{f}(Au+b)$, $AN(x,\epsilon)+b = \hat{N}(Ax+b,\epsilon)$. $\square$

Corollary 1. *Under the setting of Equations* (21), (22), *and* (24), $T(x,\epsilon) = A^T\text{Co}V(Ax+b,\epsilon) = \text{Co}A^TV(Ax+b,\epsilon)$.

Proof. The first equality is simply the combination of Lemma 2 and Theorem 3. The second is clear from the definition of convex hull. $\square$

So the approximate subdifferentials $\hat{T}$ and T, of the functions $\hat{f}$ and f, respectively, are related by

$$\hat{T}(\hat{x},\epsilon) = \text{Co}V(\hat{x},\epsilon) \quad \text{and} \quad T(x,\epsilon) = A^T\text{Co}V(Ax+b,\epsilon), \tag{35}$$

for an appropriate choice of neighborhoods.

Lemma 3. *Under the setting of Equations* (21), (22), *and* (24), *if* $\hat{y} \in \hat{N}(\hat{x},\epsilon)$, *then* $\partial\hat{f}(\hat{y}) \subseteq \hat{T}(\hat{x},\epsilon)$.

Proof. If $\hat{f}$ is nondifferentiable anywhere on $\hat{N}(\hat{x},\epsilon)$, the inclusion follows from the definition of $\hat{T}$. So, we will consider the case when the neighborhood contains no nondifferentiable points. Let $i' := \arg\max_i \hat{x}_i$. If $\hat{f}$ is instead differentiable everywhere on the neighborhood around $\hat{x}$, this means $\hat{u}_i < \hat{u}_{i'}$ for all $u \in \hat{N}(\hat{x},\epsilon)$ for any $i \neq i'$. Therefore, $\nabla\hat{f}(\hat{x}) = \nabla\hat{f}(\hat{y}) = \{e_{i'}\}$, and the lemma is true as an equality. $\square$

Theorem 4. *Under the setting of Equations* (21), (22), *and* (24), *let* $\hat{x},\hat{s} \in \mathbb{R}^p$ *and* $\alpha \in [0,1]$. *Let* $\hat{y} := \hat{x} + \alpha(\hat{s}-\hat{x})$. *Then,*

$$\min_{\hat{d}\in\hat{T}(x,\epsilon)} \frac{1}{\alpha^2}\left(\hat{f}(\hat{y}) - \hat{f}(\hat{x}) - \langle\hat{y}-\hat{x},\hat{d}\rangle\right) \leq \frac{2}{\epsilon}\|\hat{s}-\hat{x}\|_\infty^2. \tag{36}$$

Proof. It is sufficient to look at the following two cases,

Case 1: $\hat{y} \in \hat{N}(\hat{x},\epsilon)$.

Let $i' = \arg\max_i \hat{y}_i$. Then, $e_{i'} \in \partial\hat{f}(\hat{y}) \subseteq \hat{T}(x,\epsilon)$ from Equation (23) and Lemma 3. Then. the left-hand side of Equation (36) is bounded above by

$$\hat{f}(\hat{y}) - \hat{f}(\hat{x}) - \langle\hat{y}-\hat{x},e_{i'}\rangle = \hat{y}_{i'} - \left(\max_i \hat{x}_i\right) - (\hat{y}_{i'} - \hat{x}_{i'}) = \hat{x}_{i'} - \left(\max_i \hat{x}_i\right) \leq 0. \tag{37}$$

Case 2: $\hat{y} \notin \hat{N}(\hat{x},\epsilon)$.

First, note that in this case we have a lower bound on α:

$$\alpha \geq \frac{\epsilon}{\|\hat{s}-\hat{x}\|_\infty}. \tag{38}$$

Then, we use the fact that $\|\hat{d}\|_1 \leq 1$ for any $\hat{d} \in \partial\hat{f}(\hat{u})$ for any $\hat{u} \in \mathbb{R}^p$ to bound the left-hand side of Equation (36) by

$$\begin{aligned}\hat{f}(\hat{y}) - \hat{f}(\hat{x}) + \|\hat{y}-\hat{x}\|_\infty &= \left(\max_i\ \hat{x}_i + \alpha(\hat{s}_i - \hat{x}_i)\right) - \left(\max_i \hat{x}_i\right) + \alpha\|\hat{s}-\hat{x}\|_\infty \\ &\leq \left(\max_i \hat{x}_i\right) + \alpha\left(\max_i \hat{s}_i - \hat{x}_i\right) - \left(\max_i \hat{x}_i\right) + \alpha\|\hat{s}-\hat{x}\|_\infty \leq 2\alpha\|\hat{s}-\hat{x}\|_\infty. \end{aligned} \tag{39}$$

The bound in the lemma then follows from Equations (38) and (39).

Remark 1. Note that the lower bound of α in Equation (38) is only for analyzing the curvature constant $C_f(\epsilon)$, and it has no consequence for implementation. Essentially, $C_f(\epsilon)$ is a quantity that depends only on f, D and does not depend on α; that is, Equation (36) holds for any $\alpha \in [0,1]$. The convergence result follows from Theorem 1 as long as $C_f(\epsilon) < \infty$.

Let us denote the diameter of a set S with respect to the q-norm by $\text{diam}_q(S) = \max_{x,y\in S}\|x-y\|_q$. Then, we can further bound Equation (36) by the diameter of the bounded feasible set.

Corollary 2. *Given the problem of minimizing a piecewise linear function* $f(x) = \max_i(Ax+b)_i$ *over a bounded set* D, *the corresponding* C_f *is bounded above by* $C_f(\epsilon) \leq \frac{2}{\epsilon}(\text{diam}_\infty(AD))^2$.

Proof. This comes from the definition,

$$\operatorname{diam}_p(S) = \max_{x,y\in S} \|x-y\|_p, \tag{40}$$

so that $\|\hat{s}-\hat{x}\|_\infty \le \operatorname{diam}_\infty(AD)$, for all $\hat{s} = As+b, \hat{x} = Ax+b$ with $x, s \in D$. □

4.1.1. ℓ_1-Norm-Regularized SVM. The first machine learning problem we analyze is the ℓ_1-norm-regularized SVM. Since the ℓ_1 norm is sparsity-inducing, the regularization $\|w\|_1$ gives an optimization problem that finds a separating hyperplane in only a small subset of the features. This is expected to perform well in problems where there is a large number of redundant or noisy features included with a few informative features.

Suppose we are given training examples $(x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}$. To train a classifier on this input, we start with a hard-margin variant of the ℓ_1-norm-regularized SVM (Mangasarian 2000):

$$\min_{w\in\mathbb{R}^d, \gamma\in\mathbb{R}} \|w\|_1 \text{ s.t. } y_i\left(w^T x_i - \gamma\right) \ge 1 \ \forall i. \tag{41}$$

Define matrix $A^+ \in \mathbb{R}^{d\times m}$ that has the positive examples in the columns, and $A^- \in \mathbb{R}^{d\times n}$ has the negative examples in the columns. Bennett and Bredensteiner (2000) provide a dual to the soft-margin ℓ_1-norm-regularized SVM with hinge loss, related to polytope distance formulations of the ℓ_∞-norm-regularized SVM (Gärtner and Jaggi 2009):

$$\min_{u\in\Delta_m, v\in\Delta_n} \|A^+u - A^-v\|_\infty \text{ s.t. } 0 \le u, v \le \frac{1}{R}. \tag{42}$$

The elementwise upper bounds on u and v and the parameter R come from the reduced convex hull construction in Bennett and Bredensteiner (2000). When $R = 1$, this yields a dual to the hard-margin problem. In the hard-margin case, the feasible set is $D := \Delta_m \times \Delta_n$, where Δ_m and Δ_n denote the unit simplices in $\mathbb{R}^m$ and $\mathbb{R}^n$, respectively, and the feasible set is a subset of D in the soft-margin case. Therefore, using Corollary 2, we get $C_f(\epsilon) \le \frac{2}{\epsilon}(\operatorname{diam}_\infty(A^+\Delta_m) + \operatorname{diam}_\infty(A^-\Delta_n))$.

Frank Wolfe subproblems. For the dual problem in Equation (42), express our objective as $f(x) := \|Ax\|_\infty$, for A and x such that $Ax = A^+u - A^-v$. We can then write f as in Equation (21), where the linear functions are given by the rows of A and the rows of $-A$. Define the approximate subdifferential as in Equation (35). In this case,

$$V(\hat{x}, \epsilon) = \{e_i \mid \hat{x}_i \ge \|\hat{x}\|_\infty - 2\epsilon\} \cup \{-e_i \mid \hat{x}_i \le -\|\hat{x}\|_\infty + 2\epsilon\}. \tag{43}$$

We now turn to the subproblems in Equation (9) for this $T(x, \epsilon)$ and feasible region D. Observe that the maximum over d will have a solution equal to a vertex of $T(x, \epsilon)$. An optimal z is found by

$$\min_{z\in\mathbb{R}^{m+n}, \mu\in\mathbb{R}} \mu \quad \text{s.t.} \quad \langle Az - Ax, d_v\rangle \le \mu \quad \text{for } d_v \in V(Ax, \epsilon), \mathbf{1}^T z_+ = 1, \mathbf{1}^T z_- = 1, 0 \le z \le \frac{1}{R}, \tag{44}$$

where z_+ and z_- are the indices of the positive and negative examples, respectively. There will be an optimal (z, μ) such that $n + m + 1$ linearly independent constraints are active, and $z_i \le \frac{1}{R}$ will be active for no more than $2\lceil R\rceil$ indices i. This means that there is a solution z with no more than $2\lceil R\rceil + |V(Ax, \epsilon)| - 1$ nonzeros. Clearly, since the number of nonzero entries is strictly less than the total number of dimensions, the algorithm is sublinear.

4.1.1.1. Remarks. Here we note that the construction of coreset depends on the assumption that each iteration involves only $O(1)$ examples. This is true under the conditions that we describe now. Typically, R will be small for well-separated classes. Due to complementary slackness, for small ϵ we expect that $V(Ax, \epsilon)$ at the optimum will correspond to the nonzero features in the sparse separating hyperplane. This provides a central intuition behind the use of coresets. When a problem is easy in a geometric or artificial intelligence sense, having well-separated classes with few relevant features, the resulting coreset is small. Conversely, in problems that are hard, the coreset size may approach the size of the original data set. We also note that the steps s_k may not be at a vertex of D, slightly complicating the coreset construction shown empirically in the later section. However, we still see useful results for nonsmooth f and polyhedral D if s is on a low-dimensional face of D, then s is a convex combination of a small number of vertices of D. Then the step will introduce more than one atom, indeed all of the vertices of the lowest-dimensional face on which s lies, but still a small number.

4.1.1.2. Tractability of Subproblems. If we look at the subproblems from a computational geometry perspective, our algorithm will look roughly like Gilbert's algorithm (Gilbert 1966). This is similar to what is described in Bennett and Bredensteiner (2000), though the ℓ_1-SVM case is not developed fully in that paper and the $T(x,e)$ construction introduces many additional issues. We see that ℓ_1-SVM subproblems, i.e., Equation (44), is solved in $O(R|V(Ax,e)|)$ time, given that example points determine the axis-aligned bounding box of the R-reduced convex hull of each class. The bounding box can be computed in time $O(Rnd)$ and $O(Rd)$ space: a cheap one-time cost performed before the optimization. One can also construct the bounding box lazily in $O(Rnd^*)$ time for d^* the cardinality of the union of all $V(Ax,e)$ across all iterations, and for problems that are easy in a geometric sense, we will have $d^* \ll d$ making the procedure sublinear. Hence, it is important to see that one would not necessarily directly perform the optimization in Equation (9) using a generic solver.

4.1.1.3. Coreset Bound. The previous results for ℓ_1-norm SVM provide the necessary lemmas to show a coreset bound for this problem. The construction of the coreset in this work follows the same intuition as when using the ordinary FW method (Clarkson 2010). For each iterate $x_k \in \mathbb{R}^n$, and the corresponding subproblem solutions s_k, each index into the vector represents an input example. We may take those examples for which the index in s_k is nonzero, and call this set S_k. Using Algorithm 1 to build a coreset, we take the coreset at iteration K to be $\bar{S}_K = \cup_{k=1}^{K} S_k$.

Suppose we seek a coreset that will provide a ϵ-approximation solution. By Theorem 1, if we choose K to ensure that both sides of Equation (11) are (multiplicatively) bounded above by $(1+\epsilon)$, then the union $\bar{S}_K$ will provide this coreset. This approximation bound will be satisfied for

$$K \geq \frac{\left(2^{5/2}L + 2^{3/2}D_f\right)^2}{(1+\epsilon)^2} - 2. \tag{45}$$

The set of examples $\bar{S}_K$ will then be a $(1+\epsilon)$ coreset for any K satisfying this bound. We know from the above description that the ℓ_1-SVM dual objective will be $\|A\|$-Lipschitz, and that $C_f(\epsilon) \leq \frac{D_f}{\epsilon}$ for $D_f = 2(\text{diam}_\infty(A^+\Delta_m) + \text{diam}_\infty(A^-\Delta_n))$.

For a coreset to be empirically useful, we also want it to be of small cardinality. The next step is therefore to bound $|\bar{S}_K|$. We earlier showed that $|S_k| \leq 2\lceil R \rceil + d^* - 1$ for all k. Using a union bound,

$$|\bar{S}_K|(2\lceil R \rceil + d^* - 1).$$

Combining these, we can use Algorithm 1 to construct an ϵ coreset of size $O\left(\frac{1}{\epsilon^2}\right)$.

4.1.2. Balanced Development. A direct application of the piecewise linear discussion in Section 4.1 is the problem of balanced development (see Nesterov 2009). Let $A = (a_1, \ldots, a_N) \in \mathbb{R}^{m \times n}$, where m is the number of attributes and n is the number of products. Hence, the entry A_{ij} is the concentration of attribute i in product j. Let b_i be the minimum concentration of attribute i that we require in our solution. Also, $p_j \geq 0$ is the unitary price of product j. The problem is then to make an investment decision between the products to maximize the minimum amount of each attribute. In Nesterov (2009), the author provides a dual problem:

$$\tau^* = \min_y \{\phi(y) : y \in \Delta_m\} \text{ where } \phi(y) = \max_{1 \leq j \leq n} \frac{1}{p_j} \langle a_j, Ey \rangle, \tag{46}$$

where $E \in \mathbb{R}^{m \times m}$ is diagonal with $E_{ii} = \frac{1}{b_i}$. Corollary 2 implies $C_f(\epsilon) \leq \frac{2}{\epsilon}(\text{diam}_\infty(A\Delta_m))^2$, and combined with the simplicial feasible region this provides a coreset over the attributes.

4.2. Multiway Graph Cuts

We consider the model for multilabel graph cuts in Călinescu et al. (2000) and Recht et al. (2011). This problem seeks to assign one of d labels to each of n nodes. The graph structure is determined by similarity matrix W, with weight w_{uv} between vertices u and v. A convex relaxation of this problem is given by

$$\min_x \quad \sum_{u \sim v} w_{uv} \|x_u - x_v\|_1 \quad \text{s.t.} \quad x_v \in \Delta_d \text{ for } v = 1, \ldots, n. \tag{47}$$

We further constrain a set of seed nodes to have $x_v = e_i$ if the label of seed v is $i \in \{1, \ldots, d\}$. Let $X \in \mathbb{R}^{d \times n}$ be the decision variable, a matrix such that each column is the soft labeling for a node. If we let B be the incidence matrix for an orientation of the graph, and I the $d \times d$ identity matrix, then we can rewrite the objective as

$$f(\mathrm{vec}(X)) = \|(B \otimes I)\mathrm{vec}(X)\|_1 = \|\mathrm{vec}(XB^T)\|_1, \tag{48}$$

where $\otimes$ is the matrix Kronecker product.

Now define $\hat{f}(\hat{x}) := \|\hat{x}\|_1$ so that $f(\mathrm{vec}(X)) = \hat{f}((B \otimes I)\mathrm{vec}(X))$. We start by deriving the subdifferentials for $\hat{f}$. For the approximate subdifferentials of $\hat{f}$, we choose the neighborhoods $\hat{N}(\hat{x}, \epsilon)$ to be the ℓ_1 ball of radius ϵ at x.

Lemma 4. *Similar to Lemma 3, the subdifferential of the neighborhood is a subset of the approximate differential; that is, if $\hat{y} \in \hat{N}(\hat{x}, \epsilon)$, then $\partial\hat{f}(\hat{y}) \subseteq \hat{T}(\hat{x}, \epsilon)$.*

Proof. If $\hat{f}$ is nondifferentiable on $\hat{N}(\hat{x}, \epsilon)$, the result follows from the definition.

Otherwise, if $\hat{f}$ is differentiable at all points on $\hat{N}(\hat{x}, \epsilon)$, then $|\hat{x}_i| > \epsilon$ for all i. Then, because $\|\hat{y} - \hat{x}\|_1 \le \epsilon$, $\mathrm{sign}(\hat{y}_i) = \mathrm{sign}(\hat{x}_i)$ for all i, and $\nabla\hat{f}(\hat{y}) = \nabla\hat{f}(\hat{x})$ and we have the lemma statement with equality. □

Theorem 5. *Consider any $\hat{x}, \hat{s}$ and $\alpha \in [0, 1]$. Let $\hat{y} := \hat{x} + \alpha(\hat{s} - \hat{x})$. Then,*

$$\min_{\hat{d} \in \hat{T}(\hat{x}, \epsilon)} \frac{1}{\alpha^2}\left(\hat{f}(\hat{y}) - \hat{f}(\hat{x}) - \langle \hat{y} - \hat{x}, \hat{d}\rangle\right) \le \frac{2}{\epsilon}\|\hat{s} - \hat{x}\|_1. \tag{49}$$

Proof. We verify two cases,

Case 1: $\hat{y} \in \hat{N}(\hat{x}, \epsilon)$.

If we define the subgradient $\hat{d}$ elementwise by $\hat{d}_i = \mathrm{sign}(\hat{y}_i)$, then $\hat{d} \in \partial\hat{f}(\hat{y})$. Observe that $\langle \hat{y}, \hat{d}\rangle = \|y\|_1$. Then the left-hand side of Equation (49) can be bounded above as follows:

$$\hat{f}(\hat{y}) - \hat{f}(\hat{x}) - \langle \hat{y} - \hat{x}, \hat{d}\rangle = \|\hat{y}\|_1 - \|\hat{x}\|_1 - \|\hat{y}\|_1 + \langle \hat{x}, \hat{d}\rangle = \langle \hat{x}, \hat{d}\rangle - \|\hat{x}\|_1. \tag{50}$$

And because $\|\hat{d}\|_\infty \le 1$, Equation (50) is nonpositive. We therefore only need to bound the second case.

Case 2: $\hat{y} \notin \hat{N}(\hat{x}, \epsilon)$.

Here we have the lower bound on α that

$$\alpha \ge \frac{\epsilon}{\|\hat{s} - \hat{x}\|_1}, \tag{51}$$

and because $\|\hat{d}\|_\infty \le 1$,

$$\hat{f}(\hat{y}) - \hat{f}(\hat{x}) - \langle \hat{y} - \hat{x}, \hat{d}\rangle \le \|\hat{y}\|_1 - \|\hat{x}\|_1 + \|\hat{y} - \hat{x}\|_1 = \|\hat{x} + \alpha(\hat{s} - \hat{x})\|_1 - \|\hat{x}\|_1 + \alpha\|\hat{s} - \hat{x}\|_1 \le 2\alpha\|\hat{s} - \hat{x}\|_1, \tag{52}$$

using the triangle inequality. The theorem statement then follows from this inequality and the bound on α.

Remark 2. Similar to Remark 1, the lower bound of α in Equation (51) has no consequence for implementation.

So, much like in the piecewise linear case, we can conclude the following.

Corollary 3. *Given the problem of minimizing a piecewise linear function $f(x) = \|Ax + b\|_1$ over a bounded set D, the corresponding C_f is bounded above by $C_f(\epsilon) \le \frac{2}{\epsilon}(\mathrm{diam}_1(AD))^2$.*

Note that in the case of multiway graph cuts, D consists of the Cartesian product of n simplices of dimension d. We therefore expect the bound from Corollary 3 will be $\Omega(n)$. The tractability, coreset (and hence the sublinearity) discussion in this case follows directly from the ℓ_1-regularized SVM case.

4.3. 1-Median

We can express the 1-median problem as an optimization over the simplex of convex combinations of the input points. Suppose we are given a (multi)set of points $\mathcal{P} = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$. Let $A \in \mathbb{R}^{d \times n}$ be a matrix with p_i in column i. We can then write the 1-median problem as

$$\min_{x \in \Delta_n} f(x) := \hat{f}(Ax) := \frac{1}{n}\sum_{i=1}^{n} \hat{f}_i(Ax) := \frac{1}{n}\sum_{i=1}^{n} \|Ax - p_i\|_2. \tag{53}$$

The notation $\|\cdot\|$ denotes the ℓ_2 norm. Here, f is nondifferentiable for any x such that $Ax \in \mathcal{P}$, and differentiable elsewhere. We show that $C_f(\epsilon) \leq \frac{2}{\epsilon}\text{diam}_2(A\Delta_n)^2$, and extend this to a coreset result. We note that recently Cohen et al. (2016) developed an stochastic gradient descent algorithm that has a running time of $O(1/\epsilon^2)$ matching our results. The key difference is that the algorithm in Cohen et al. (2016) is randomized, so their result holds only in probability < 1.

The subdifferentials are given by

$$\partial f(x) = A^T \partial \hat{f}(Ax) = \frac{1}{n} A^T \sum_{i=1}^{n} \partial \hat{f}_i(Ax), \quad \partial \hat{f}_i(Ax) = \begin{cases} \left\{ \frac{Ax - p_i}{\|Ax - p_i\|} \right\} & \text{if } Ax \neq p_i, \\ \bar{\mathcal{B}}(0,1) & \text{if } Ax = p_i. \end{cases} \tag{54}$$

Note that the $\frac{1}{n}$ factor in Equation (53) is necessary to produce sensible approximation bounds. If we duplicate each point in our input set, this will double the value of the sum in the objective for any choice of median. Without the $\frac{1}{n}$ factor, any approximation bound expressed in terms of the objective would necessarily be $\Omega(n)$.

The k-median problem has been considered previously in the coreset setting. Har-Peled and Kushal (2005) give a coreset construction by binning the points to derive a coreset of size $O(k\epsilon^{-d} \log n)$. For $k > 1$, the k-median problem is known to be nonconvex, so we here consider the problem of calculating a single median.

For this problem, we choose neighborhoods as

$$N(x, \epsilon) = \{u \mid \|Au - Ax\| < \epsilon \text{ and } u - x \in \text{im}(A^T)\}. \tag{55}$$

Note that the choice of $N(x, \epsilon)$ in Equation (55) meets the assumptions in the previous convergence bounds, as $x \in N(x, \epsilon)$, and $N(x, \epsilon) \subseteq B(x, \|A^+\|\epsilon)$. Here, $\|A^+\|$ is the operator norm of the pseudoinverse. The subdifferentials are described by

$$\partial f(x) = \frac{1}{n} A^T \sum_{i=1}^{n} \partial \hat{f}_i(Ax), \quad \partial \hat{f}_i(Ax) = \begin{cases} \left\{ \frac{Ax - p_i}{\|Ax - p_i\|} \right\} & \text{if } Ax \neq p_i, \\ \bar{\mathcal{B}}(0,1) & \text{if } Ax = p_i. \end{cases} \tag{56}$$

4.4. Iteration Bounds

To bound $C_f(\epsilon)$, we bound the quantity,

$$\frac{1}{n\alpha^2} \min_{u \in N(x,\epsilon)} \left(\sum_{Au \neq p_i} \left(\|Ay - p_i\| - \|Ax - p_i\| - \left\langle Ay - Ax, \frac{Au - p_i}{\|Au - p_i\|} \right\rangle \right) + E(u)(\|Ay - Au\| - \|Ax - Au\| - \|Ay - Ax\|) \right),$$

for any x, s in the simplex and $y = x + \alpha(s - x)$ for $\alpha \in [0, 1]$. Here, $E(u)$ is the number of points such that $Au = p_i$. This is equivalent to the terms in the supremum in $C_f(\epsilon)$, where we have substituted the analytic solution for the minimum element of $\partial f(u)$. By the triangle inequality, for any u,

$$\|Ax - Au\| + \|Ay - Ax\| \geq \|(Ax - Au) + (Ay - Ax)\| = \|Ay - Au\|. \tag{57}$$

Since the nondifferentiable terms are nonpositive, we only need to bound the differentiable terms. First, assume that $y - x \in \text{im}(A^T)$. This is without loss of generality, as we can replace s with its projection $s' \in x + \text{im}(A^T)$ and y with $y' = x + \alpha(s' - x)$. Because $s' - s \in \ker(A)$, this yields $Ay' = Ay$, and the quantity we seek to bound will be identical.

Case 1: $y \in N(x, \epsilon)$.

If $Au = Ay$, the ith differentiable term is given by

$$\begin{aligned} &\|Ay - p_i\| - \|Ax - p_i\| - \left\langle Ay - Ax, \frac{Ay - p_i}{\|Ay - p_i\|} \right\rangle \\ &\quad = \frac{1}{\|Ay - p_i\|} \left(\langle Ay - p_i, Ay - p_i \rangle - \|Ax - p_i\| \, \|Ay - p_i\| - \langle Ay - Ax, Ay - p_i \rangle \right) \\ &\quad = \frac{1}{\|Ay - p_i\|} \left(\langle Ax - p_i, Ay - p_i \rangle - \|Ax - p_i\| \, \|Ay - p_i\| \right), \end{aligned} \tag{58}$$

which by the Cauchy-Schwarz inequality is nonpositive. Therefore, to provide an upper bound on $C_f(\epsilon)$, we need only consider the second case.

Case 2: $y \notin N(x,\epsilon)$.

First, observe that in this case we have a lower bound on α. Specifically, because y lies outside the neighborhood around x defined by Equation (55) (and in the set $x+\mathrm{im}(A^T)$ by assumption), $\|Ay-Ax\| = \alpha\|As-Ax\| \geq \epsilon$,

$$\alpha \geq \frac{\epsilon}{\|As-Ax\|} \geq \frac{\epsilon}{\mathrm{diam}(A\Delta_n)}. \tag{59}$$

Consider in the following, any choice of u from the neighborhood around x. Plugging in $y = x+\alpha(s-x)$ to the differentiable terms,

$$\frac{\sum_{Au\neq p_i} \|Ay-p_i\| - \|Ax-p_i\| - \left\langle Ay-Ax, \frac{Ax-p_i}{\|Ax-p_i\|}\right\rangle}{n\alpha^2} = \frac{\sum_{Au\neq p_i} \|Ax-p_i+\alpha(As-Ax)\| - \|Ax-p_i\| - \alpha\left\langle As-Ax, \frac{Au-p_i}{\|Au-p_i\|}\right\rangle}{n\alpha^2}$$

$$\leq \frac{1}{n\alpha^2}\sum_{Au\neq p_i} 2\alpha\|As-Ax\| \leq \frac{2\|As-Ax\|}{\alpha} \leq \frac{2\mathrm{diam}(A\Delta_n)}{\alpha}, \tag{60}$$

where we used the triangle and Cauchy-Schwarz inequalities for the inequalities. The combination of Equations (59) and (60), taken over all x, s and all α that fall in this case, gives:

$$C_f(\epsilon) \leq \frac{2}{\epsilon}\mathrm{diam}(A\Delta_n)^2. \tag{61}$$

Next, observe that if ϵ is small, as will be the case throughout the optimization if we choose $\epsilon_k \in \Theta(\sqrt{\alpha_k})$ with a small constant term, then with high probability the neighborhood around the iterates for any k that are not near the solution will be differentiable. This will yield $T(x,\epsilon) = \{\nabla f(x)\}$ and the subproblems will be a linear program over the simplex, which will only introduce no more than one nonzero as in Clarkson (2010), which indeed means that the procedure is sublinear.

4.5. Sparse PCA with Constraints

Sparse PCA is an important problem that is widely used for feature extraction and learning. In this problem, we consider the formulation given in Grbovic et al. (2012). The objective of sparse PCA is to decompose a covariance matrix S into near orthogonal principal components $[u_1,\ldots,u_k]$, while constraining the number of nonzero components of each u_k to a required constant $r \in \mathbb{N}$. Hence, the problem of maximizing the variance of component u_k with the cardinality constraint can be written as

$$\max_u \quad u^TSu \quad \text{s.t.} \quad \|u\|_2 = 1,\ \mathrm{card}(u). \tag{62}$$

Writing down the convex relaxation of the problem after setting $U = uu^T$, we get the following optimization problem,

$$\max_U \quad \mathrm{Tr}(SU) \quad \text{s.t.} \quad \mathrm{Tr}(U) = 1,\ 1^T|U|1 \leq r,\ U = U^T,\ U \succeq 0,\ U \in \mathbb{R}^{n\times n}. \tag{63}$$

The last inequality in Equation (63) is the standard linear matrix inequality that requires U to be positive semidefinite. Note that we have an exponential number of linear constraints in the form of $1^T|U|1 \leq r$ constraint. Hence, in general, this problem might practically take very long to solve unless one uses a specialized interior point method that exploits the structure of the constraints. Hence, these constraints are taken to the objective with a penalty ρ associated with it as

$$\max_U \quad \mathrm{Tr}(SU) - \|\rho\mathrm{vec}(U)\|_1 \quad \text{s.t.} \quad \mathrm{Tr}(U) = 1,\ U = U^T,\ U \succeq 0,\ U \in \mathbb{R}^{n\times n}, \tag{64}$$

where $\mathrm{vec}(U)$ denotes the vectorized form of U. To this formulation, Grbovic et al. (2012) suggest adding a class of feature grouping constraints that is motivated by the maintenance scheduling problem (Cui 2008). Let us denote by $\mathbf{1}$ a reliability vector such that $\mathbf{1}_i \in [0,1)$ is a probability that sensor i will need maintenance

during a certain time period. Then, the reliability matrix L is constructed by setting each column to $\mathbf{1}$ and then setting $L_{ii} = 0\ \forall\ 1 \le i \le n$. Using similar techniques we end up with the following formulation:

$$\max_U\ \mathrm{Tr}(SU) - \|\rho \mathrm{vec}(U)\|_1 - \|\ \mathrm{vec}(\eta U \circ \log(1-L))\|_1 \quad \text{s.t. } \mathrm{Tr}(U) = 1, U = U^T, U \ge 0, U \in \mathbb{R}^{n\times n}, \tag{65}$$

where η is the penalty that controls the reliability of the component and $\circ$ is the standard Hadamard product or elementwise product. Let $\vec{U}$ denote the vectorized form of U for simplicity in notation and similarly $\mathrm{vec}(\log(1-L)) = \vec{l}$. Note that $\vec{l}, \vec{U} \in \mathbb{R}^{n^2\times 1}$. Then, the problem can be written as

$$\max_U \quad \mathrm{Tr}(SU) - \sum_{i=1}^{n^2}\left(\rho + \eta\vec{l}_i\right)|\vec{U}_i| \quad \text{s.t.} \quad \mathrm{Tr}(U) = 1,\ U = U^T,\ U \ge 0,\ U \in \mathbb{R}^{n\times n}. \tag{66}$$

Noting that $\mathrm{Tr}(SU) = \sum_{i=1}^n \sum_{j=1}^n S_{ij}U_{ij} = \vec{S}^T\vec{U} = \sum_{i=1}^{n^2} \vec{S}_i\vec{U}_i$, we can write the objective function as a supremum of 2^{n^2} affine functions. Also using the min-max theorem we convert this to a minimization problem. Hence, the problem becomes

$$\min_U \quad \max_i\left(a_i^T\vec{U}\right) \quad \text{s.t.} \quad \mathrm{Tr}(U) = 1,\ U = U^T,\ U \ge 0,\ U \in \mathbb{R}^{n\times n}. \tag{67}$$

Let $S := \{U : \mathrm{Tr}(U) = 1,\ U = U^T,\ U \ge 0,\ U \in \mathbb{R}^{n\times n}\}$.

Lemma 5. *The feasible set of sparse PCA with constraints problem, S, is bounded.*

Proof. We know that $U \ge 0, U = U^T \iff \lambda_i \ge 0$ where λ_is are the eigenvalues of U and $\mathrm{Tr}(U) = 1 \iff \sum_i \lambda_i = 1$. Let $S_\lambda := \{\lambda : \sum_i \lambda_i = 1, \lambda \ge 0, \lambda \in \mathbb{R}^n\}$. Hence, S is bounded if and only if S_λ is bounded. But S_λ is trivially bounded since it is the simplex. □

Let $A \in \mathbb{R}^{2^{n^2}\times n^2}$ be the matrix containing all the a_is stacked on top of each other. Hence, our optimization problem can be written as

$$\min_U \quad \max_i\left(A\vec{U}\right)_i \quad \text{s.t.} \quad U \in S. \tag{68}$$

We can now define the neighborhoods in a similar way as in Section 4.1,

$$\hat{N}\left(\hat{\vec{U}}, \epsilon\right) = \left\{\hat{\vec{X}} \mid \|\hat{\vec{U}} - \hat{\vec{X}}\|_\infty \le \epsilon\right\}; \quad N\left(\vec{U}, \epsilon\right) = \left\{\vec{X} \mid \|A\vec{X} - A\vec{U}\|_\infty \le \epsilon \text{ and } \vec{X} - \vec{U} \in \mathrm{im}(A^T)\right\}, \tag{69}$$

and define

$$V\left(\hat{\vec{U}}, \epsilon\right) = \left\{e_i \mid \hat{\vec{U}}_i \ge \left(\max_j \hat{\vec{U}}_j\right) - 2\epsilon\right\}. \tag{70}$$

With these constructions, similar results as in Section 4.1 are obtained. Note that unlike the other cases, the subproblems here are not as easy to solve and we might require advanced numerical optimization solvers to solve the subproblems efficiently in practice.

4.6. Generic Subproblems

As we saw in the previous sections, specialized algorithms can be used to solve the subproblems that are induced by Equation (9) for many important applications. But occasionally, we might encounter problems that do not possess an inherent structure that can be easily exploited. For these cases, we now provide a generic method (or algorithm) for solving the subproblems. Specifically, we will consider the case when the optimization in Equation (9) is over a union of convex sets $C = \cup_{i=1}^{\mathfrak{n}} C_i$ where C_i is a convex set $\forall i = 1, \ldots, \mathfrak{n}$. Recall that the subdifferential of a proper convex function is a compact set for all the points in the interior of the domain (Robinson 2012), hence, many more problems (that were not considered in Section 4) fall under this case, for example, piecewise quadratic optimization that is commonly used in economic load dispatching problems (see Lee and Kim 2002). Assuming that we have access to the zeroth and first-order oracles, C can be explicitly described by the inequalities $g_{ij}(x) \le 0$, where g_{ij} are convex functions $\forall i, j$ and the total set of constraints has finite cardinality. It is easy to show that the problem of optimizing a convex function g over C can be formulated as the following convex optimization problem:

$$\min_x \quad g(x) \quad \text{s.t.} \quad s_i g_{ij}(z_i/s_i) \le 0, \forall i, j,\ \ 1^T s = 1, s \ge 0, x = \sum_{i=1}^{\mathfrak{n}} z_i. \tag{71}$$

There are two advantages of using the reformulation in Equation (71) instead of solving $\mathfrak{n}$ different convex problems: (1) in certain cases, it might be easy to design algorithms to get approximate solutions much more efficiently (Garber and Hazan 2011); and (2) standard solvers are actively being developed to handle perspective reformulations as described in Günlük and Linderoth (2012). From this discussion, we can conclude that the subproblems can be solved in polynomial time.

5. Experiments

In this section, we present experiments to assess our convergence rate numerically and provide some practical intuition for the bounds obtained in the previous sections. We provide the running time whenever it is significant (more than a few seconds). To keep the extraneous effects of specialized libraries and the number of processors negligible, all our subproblems were solved using a generic linear programming solver on a single core. The results show that the algorithm presented here is not only practical but can also be made competitive using simple alternative standard techniques to solve the subproblems.

5.1. SVMs with ℓ_1 Regularization

We first demonstrate results of our algorithm for ℓ_1-regularized SVM on a collection of four test data sets provided by Yuan et al. (2010) (see Figure 1).

For this evaluation, we also compared our method with the randomized smoothing method of Lan (2013) shown in blue in Figure 1. The top row of plots in Figure 1 (y-axis showing the duality gap) suggest that on all four data sets (leukemia, colon-cancer, ionosphere, w8a), the convergence of the baseline is slower as a function of the number of iterations. The second and the third rows show that the coreset size and the vertices in $|V(Ax, \epsilon)|$ increase at most linearly with the number of iterations as predicted in Section 4.1.1. We see that the number of vertices in $|V(Ax, \epsilon)|$ remains very small throughout our experiments. Since $\alpha_k = 2/(k+2)$ in Figure 1 (which is agnostic of f, D), we observe a little erratic behavior of our algorithm with respect to the duality gap and number of vertices in the earlier iterations. Although the convergence results shown for Lan (2013) are, in expectation, of the same order as the ones we show here, the bound is quite loose for these problems and we find that the practical convergence rate can be slow. This is especially important as the computation time of each iteration of that method in Lan (2013) increases with the iteration count, as more samples of the gradient are drawn in later iterations (unlike our algorithm). Moreover, Figure 2 shows that if we use a bisection line search instead of the step size policy suggested in the proof of Theorem 1, our algorithm converges (measured using the duality gap) in fewer than 40 iterations, making it practically fast while also keeping our theoretical results intact.

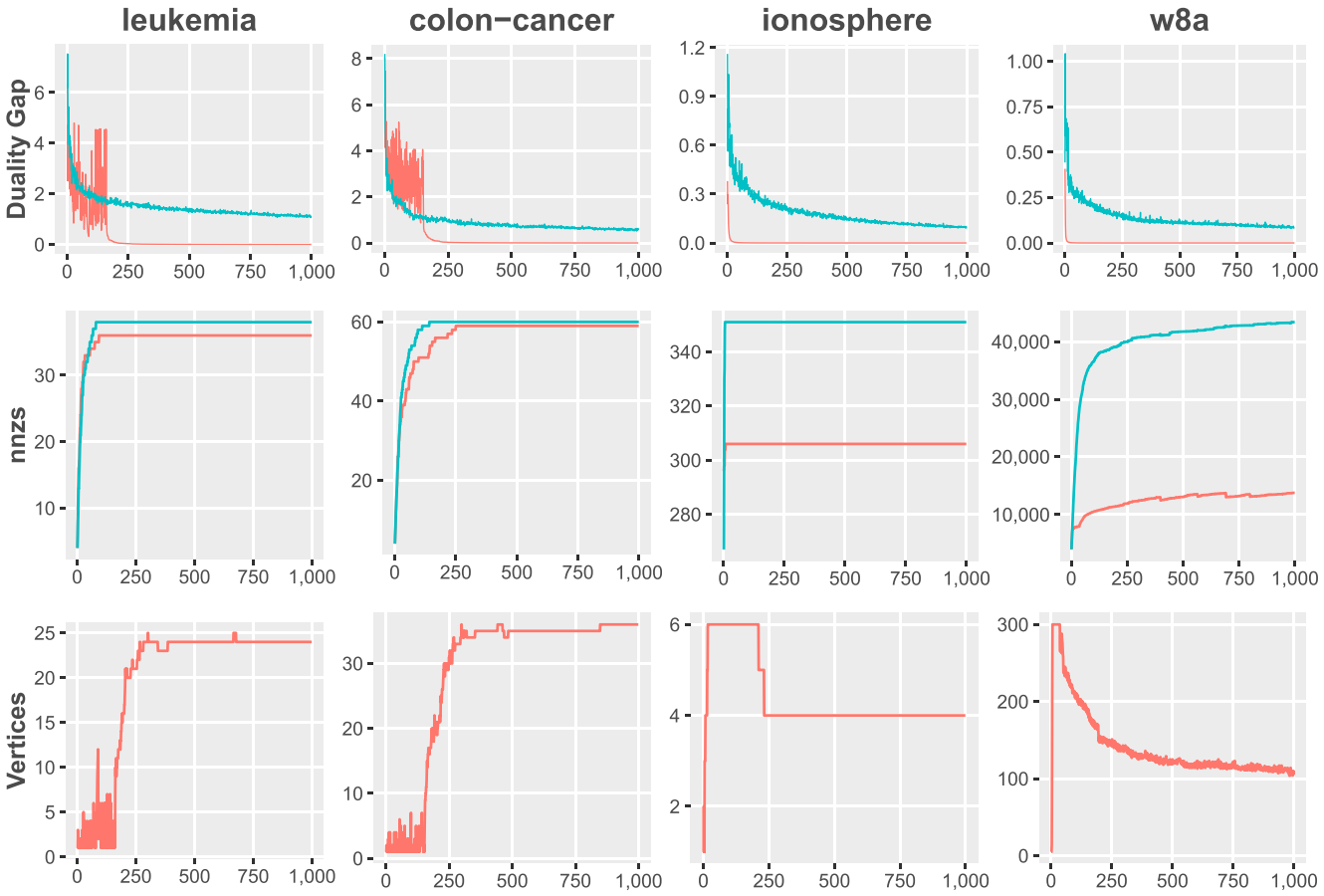
5.2. The 1-Median Problem

We tested a simple FW-based solver for the 1-median problem on synthetic data (similar to existing works for this problem). A set of points of size n is sampled from a multivariate random normal distribution. To keep the presentation succinct, we discuss those parts of this experiment that demonstrate additional useful properties of the algorithm beyond those described for ℓ_1-SVM. The left plot in the top row of Figure 3 shows how the size of the coreset varies as we increase n, whereas the right plot shows the number of iterations required for convergence. Importantly, the size of the coreset is independent of n, as the problem we are solving is fundamentally no harder. The second row shows the computational time taken: the left plot shows the total time taken whereas the right plot shows the time taken per iteration. We see that both of these scale sublinearly with the number of points. The plots agree with the theoretical results from earlier sections concerning the size of coreset and number of iterations required to obtain a solution.

5.3. Multiway Graph Cuts

To demonstrate that our algorithm is scalable to large data sets and hence it is practically applicable, we tested our algorithm on the *dblife* data set (Recht et al. 2011), which consists of 9,168 labels. Therefore, D is the product of n 9,168-dimensional simplices. Note that this problem consist of approximately 20 million decision variables and has been tackled via distributed schemes implemented on a cluster (Recht et al. 2011). Due to the large of number of labels, this data set is ideal for evaluating our algorithm as the analysis implies that the number of labels chosen must increase linearly with the number of iterations. This means that we get a sparse labeling scheme, i.e., the coreset here is the subset of the total number of labels. We evaluated this behavior by using

Figure 1. (Color online) Convergence Results for Algorithm 1 (Red) and Randomized Smoothing of Lan (2013) (Blue) on the Support Vector Machine Data Sets [Equation (42)]



Note. The horizontal axis denotes the iterations.

increasing subsets of the total number of labels. We find that even when the total number of labels is small, this trend becomes more prominent as progressively larger sets of labels are used, as shown in Figure 3 (row 3).

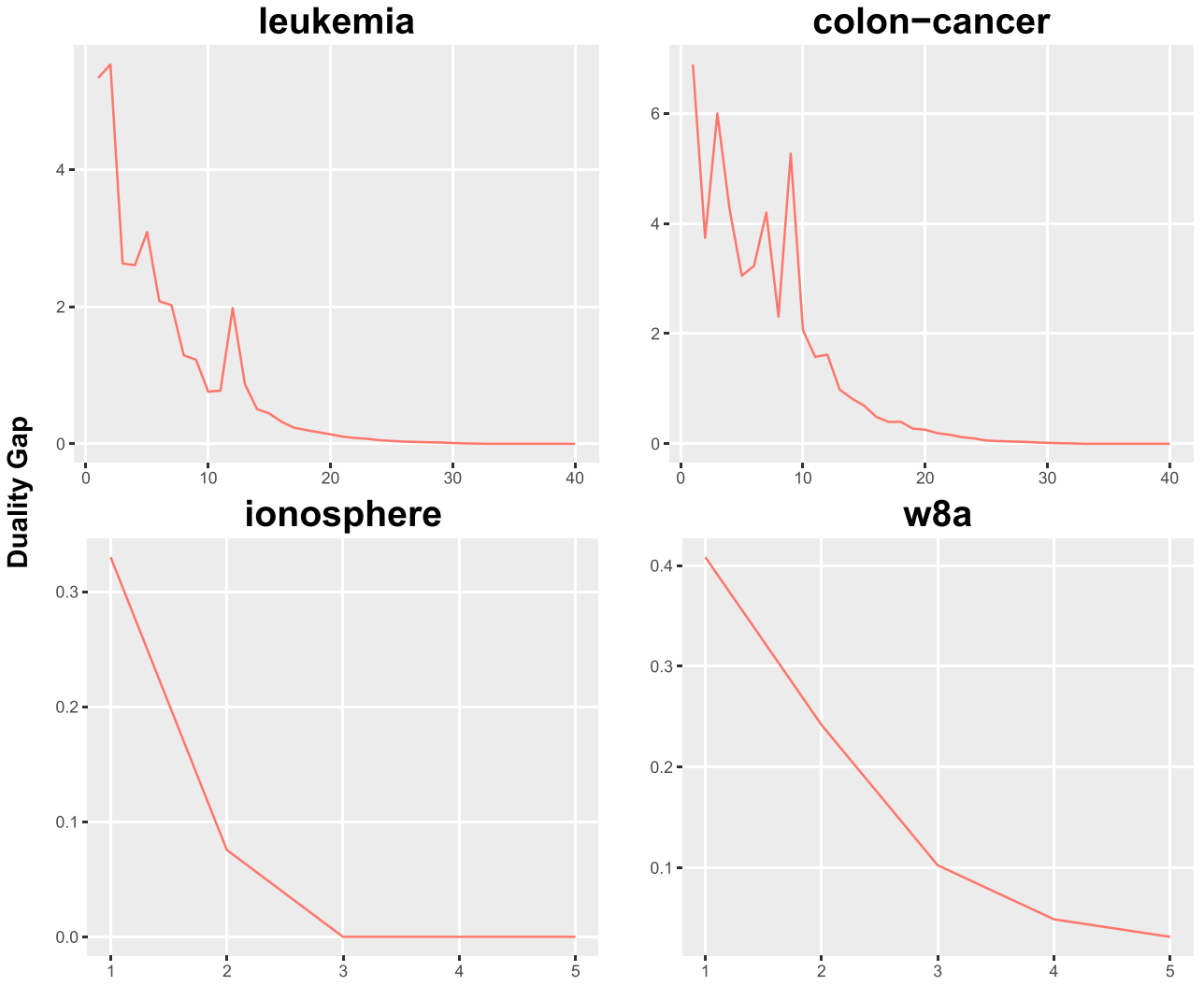
5.4. Interesting Empirical Behavior and Its Potential Relation to Existing Work

The experiments highlight some important practical aspects of the proposed algorithm.

5.4.1. How Often Do We Encounter Poor Subproblems? The number of vertices in $|V(Ax, \epsilon)|$ given by our analysis seems to be an overestimate than what is practically observed. Assuming that we initialize at a point whose neighborhood does not contain any nonsmooth points, using the fact that the set of nondifferentiable points of a convex function on a compact set is of measure zero, we can show that the next iterate of the algorithm also does not contain any nonsmooth points in its neighborhood with high probability. But extending this argument to the next iterate becomes problematic. It seems like a Lovasz local lemma type result (Kolmogorov 2018) will be able to shed some light on this behavior but the independence assumption is inapplicable, so one has to introduce randomness in the algorithm. But a separate sparsity analysis might have to be done to obtain coreset results.

5.4.2. Why Does Our Algorithm Converge So Fast?. The empirical behavior of certain algorithms has been strongly associated with the manifold identification property in recent works (see Lee and Wright 2012). Our algorithm when using a bisection line search strategy, as shown in Figure 3, converges within a few iterations and is faster than the state-of-the-art solvers for that problem. It will be interesting to see if our algorithm satisfies some variation or extension of the properties that are described in Lee and Wright (2012), but one

Figure 2. (Color online) Convergence Results for Algorithm 1 Using Bisection Line Search



issue is that the algorithm presented in Lee and Wright (2012) has randomization incorporated at each iteration, making it hard to adapt that analysis for our algorithm.

5.4.3. Tightness of Convergence. Let us consider the case of ℓ_1-SVM. Assume that the positive and negative labeled instances are generated by a Gaussian distribution with mean 0 and μ, respectively, and variance σI. Using the properties of maximum of sub-Gaussians, we have that

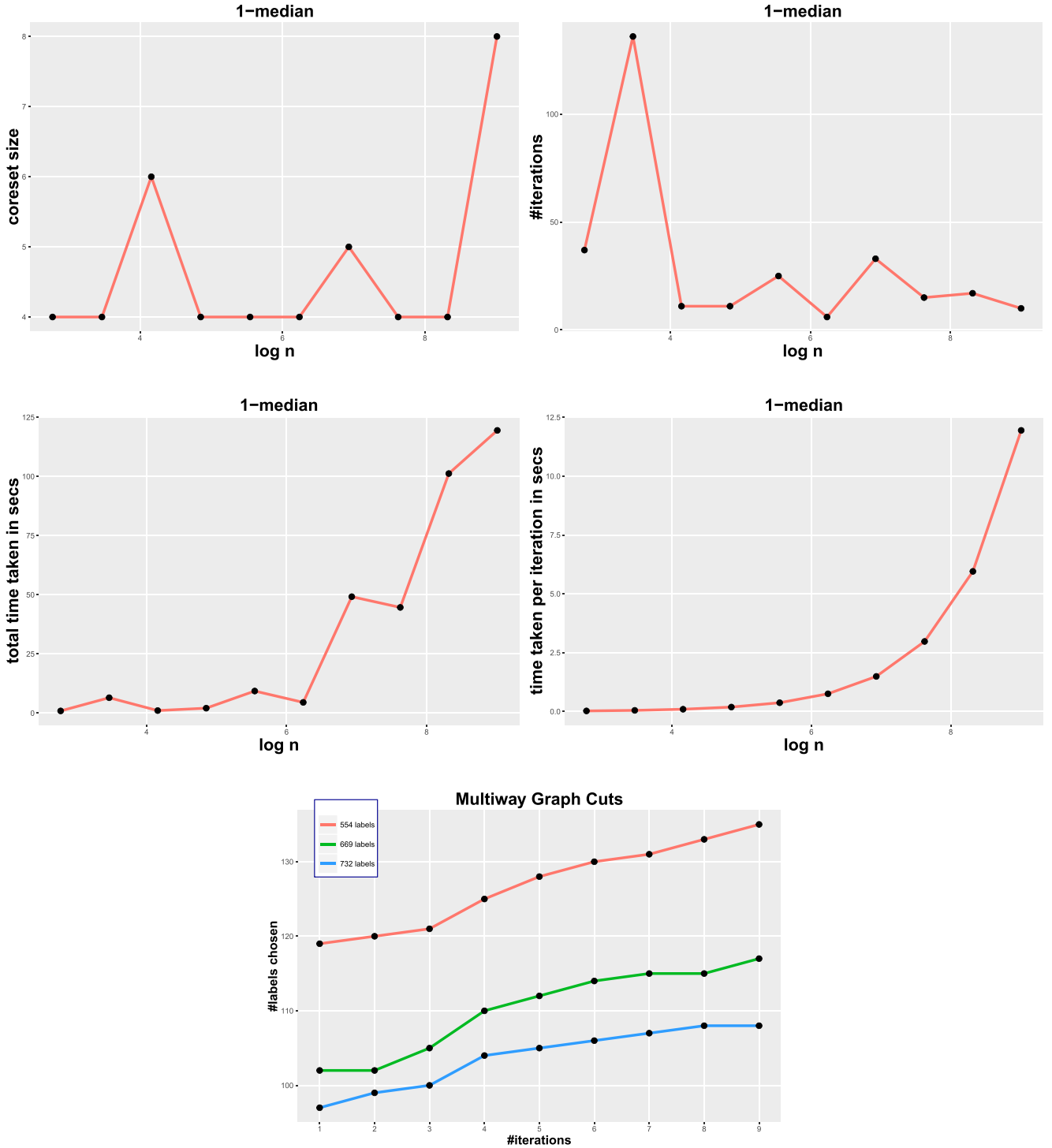
$$\mathbb{E}(\operatorname{diam}_\infty(A^+\Delta_m) + \operatorname{diam}_\infty(A^-\Delta_m)) \le \sigma \log m + \mu + \sigma \log m = 2\sigma \log m + \mu, \tag{72}$$

where the expectation is taken over the randomness in the data: A^+, A^-. This shows that the dependence of C_f is logarithmic in the dimensions. Similarly, we will get $\sqrt{n}, n \log n$ for ℓ_2, ℓ_1 norm diameters, respectively, showing that the $\mathbb{E}C_f$ is near linear in the dimensions. Now we plug this estimate in the convergence result in Theorem 1 to get

$$\mathbb{E}(f(x_k) - f(x_*)) \le \mathbb{E}\left(\frac{2^{5/2}L + 2^{3/2}D_f}{\sqrt{k+2}}\right) = 2^{5/2}\left(\frac{L + \mathbb{E}D_f}{\sqrt{k+2}}\right) = O\left(\frac{L + \log m}{\sqrt{k}}\right). \tag{73}$$

Hence, we can see that by choosing an appropriate neighborhood depending on the application, we can get faster convergence. This aspect is either not present in other FW algorithms that solve nonsmooth problems or involve a looser dependence on the constants involved. For example, Hazan and Kale (2012) achieve a $O\left(\frac{Lm}{\sqrt{k}}\right)$ convergence compared with our $O\left(\frac{L+\log^2 m}{\sqrt{k}}\right)$ here.

Figure 3. (Color online) (Top Row) Plotting the Size of the Coreset (Left) and Number of Iterations (Right) for 1-Median as We Vary the Size n of a Randomly Sampled Set of Points; (Second Row) Total Time Taken (Left) and Time Taken Per Iteration (Right); (Bottom Row) Results for Graph Cuts Problem on dblife



Notes. For the top row, we iterate until achieving a primal-dual gap of 10^{-6}. For results in the bottom row, each linear programming subproblem took approximately 60 seconds using CPLEX solver.

6. Discussion

We have presented novel coreset bounds and optimization schemes for nondifferentiable problems encountered in statistical machine learning and computer vision. The algorithm calculates sparse approximate solutions to the corresponding optimization problems deterministically in a number of iterations independent

Table 1. Summary of the Relevant Quantities Derived for Each Problem Described in the Previous Sections

Problem	C_f	Coreset
ℓ_1-support vector machine	$\frac{2}{\epsilon}(\mathrm{diam}_\infty(A^+\Delta_m) + \mathrm{diam}_\infty(A^-\Delta_m))$	Support vectors
1-median	$\frac{2}{\epsilon}\mathrm{diam}_2(A\Delta_n)^2$	Points
Graph cuts	$\frac{2}{\epsilon}(\mathrm{diam}_1(AD))^2$	Labels
Balanced development	$\frac{2}{\epsilon}(\mathrm{diam}_1(A\Delta_m))^2$	Attributes
Sparse principal component analysis with constraints	$\frac{2}{\epsilon}(\mathrm{diam}_\infty(AS))^2$	Features

of the size of the input problem, depending only on the approximation factor and the sum of the Lipschitz constant and a nonlinearity term. The central result in Theorem 1 applies to any problem of the very general form in Equation (1) with mild conditions, potentially suggesting a number of other applications beyond those considered here (see Table 1 for a summary). Though a general condition to characterize all cases where the internal subproblem is efficiently solvable may not be available, we show a broad/useful class that is efficiently solvable.

Finally, we point out a few technical properties that differentiate our method from existing FW-type methods for nonsmooth problems (Argyriou et al. 2014, Pierucci et al. 2014).

First, many methods rely on smoothing the objective function by a proximal function that needs to be designed piecemeal for specific problems. Although examples exist where these functions can be readily derived, to our knowledge there are no standard recipes for deriving proximal functions in general. Second, several existing methods assume that the proximal iteration can be solved efficiently. However, it is known that in general, the worst-case complexity of a single proximal iteration is the same as solving the original optimization problem (Parikh and Boyd 2014). Third, it is not yet clear how coreset results can be derived for many existing methods, or if it is even possible. For example, (especially) in the very large-scale setting, coreset results enable practical applications where one can store a subset of the (training) data set and still be able to perform nearly as well (on the test data). The basic expectation is that more data should not always make a problem computationally harder. An exciting implication behind coreset results in Clarkson (2010) is that this can be avoided in certain cases. But Clarkson (2010) assumes that f is smooth: an artifact of the optimization rather than a requirement of coresets per se. We showed that this assumption is not necessary. In closing, although the algorithm may not be the de facto off-the-shelf option for all nonsmooth problems, for many problems it offers a very competitive (generally applicable) alternative, and in certain cases, the theory nicely translates into significant practical benefits as well.

Acknowledgments

The authors thank Stephen J. Wright, Shuchi Chawla, Satyen Kale, and Kenneth L. Clarkson for suggestions.

References

Acharya J, Daskalakis C, Kamath GC (2015) Optimal testing for properties of distributions. *Proc. 28th Internat. Conf. Neural Inform. Processing Systems*, vol. 2 (MIT Press, Cambridge, MA), 3591–3599.

Agarwal PK, Har-Peled S, Varadarajan KR (2005) Geometric approximation via coresets. *Combin. Comput. Geometry* 52:1–30.

Argyriou A, Signoretto M, Suykens J (2014) Hybrid conditional gradient-smoothing algorithms with applications to sparse and low rank regularization. Suykens JAK, Signoretto M, Argyriou A, eds. *Regularization, Optimization, Kernels, and Support Vector Machines* (CRC Press, Boca Raton, FL), 53–82.

Bach F, Jenatton R, Mairal J, Obozinski G (2012) Optimization with sparsity-inducing penalties. *Foundations Trends Machine Learn.* 4(1):1–106.

Bachem O, Lucic M, Lattanzi S (2018) One-shot coresets: The case of *k*-clustering. *Proc. Machine Learn. Res.* 84:784–792.

Balcan MF, Ehrlich S, Liang Y (2013) Distributed *k*-means and *k*-median clustering on general topologies. *Proc. 26th Internat. Conf. Neural Inform. Processing Systems*, vol. 2 (Curran Associates, Inc., Red Hook, NY), 1995–2003.

Banerjee O, Ghaoui L, d'Aspremont A, Natsoulis G (2006) Convex optimization techniques for fitting sparse Gaussian graphical models. *Proc. 23rd Internat. Conf. Machine Learn.* (ACM, New York), 89–96.

Baykal C, Liebenwein L, Gilitschenski I, Feldman D, Rus D (2018) Data-dependent coresets for compressing neural networks with applications to generalization bounds. Working paper, Massachusetts Institute of Technology, Cambridge, MA.

Bennett KP, Bredensteiner EJ (2000) Duality and geometry in SVM classifiers. *Proc. 17th Internat. Conf. Machine Learn.* (Morgan Kaufmann Publishers Inc., San Francisco), 57–64.

Braverman V, Chestnut SR, Woodruff DP, Yang LF (2016) Streaming space complexity of nearly all functions of one variable on frequency vectors. *Proc. 35th ACM SIGMOD-SIGACT-SIGAI Sympos. Principles Database Systems* (ACM, New York), 261–276.

Călinescu G, Karloff H, Rabani Y (2000) An improved approximation algorithm for MULTIWAY CUT. *J. Comput. System Sci.* 60(3):564–574.

Chan TM (2018) Applications of Chebyshev polynomials to low-dimensional computational geometry. *J. Comput. Geometry* 9(2):3–20.

Cheung E, Li Y (2017) Nonsmooth Frank-Wolfe using uniform affine approximations. Working paper, University of Waterloo, Waterloo, ON, Canada.

Clarkson KL (2010) Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Trans. Algorithms* 6(4):Article No. 63.

Clarkson KL, Woodruff DP (2015) Input sparsity and hardness for robust subspace approximation. *Proc. 2015 IEEE 56th Annual Sympos. Foundations Comput. Sci.* (IEEE, Piscataway, NJ), 310–329.

Clarkson KL, Hazan E, Woodruff DP (2012) Sublinear optimization for machine learning. *J. ACM* 59(5):1–49.

Cohen MB, Lee YT, Miller G, Pachocki J, Sidford A (2016) Geometric median in nearly linear time. *Proc. 48th Annual ACM Sympos. Theory Comput.* (ACM, New York), 9–21.

Cui L (2008) Maintenance models and optimization. Misra KB, ed. *Handbook of Performability Engineering* (Springer, London), 789–805.

Daskalakis C, Kamath G, Wright J (2018) Which distribution distances are sublinearly testable? *Proc. 29th Annual ACM-SIAM Sympos. Discrete Algorithms* (SIAM, Philadelphia), 2747–2764.

Drineas P, Kannan R, Mahoney MW (2006) Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM J. Comput.* 36(1):132–157.

Feldman D, Monemizadeh M, Sohler C, Woodruff DP (2010) Coresets and sketches for high dimensional subspace approximation problems. *Proc. 21st Annual ACM-SIAM Sympos. Discrete Algorithms* (SIAM, Philadelphia), 630–649.

Feldman D, Sugaya A, Rus D (2012) An effective coreset compression algorithm for large scale sensor networks. *Proc. 2012 ACM/IEEE 11th Internat. Conf. Inform. Processing Sensor Networks (IPSN)* (IEEE, Piscataway, NJ), 257–268.

Frank M, Wolfe P (1956) An algorithm for quadratic programming. *Naval Res. Logist. Quart.* 3(1–2):95–110.

Garber D, Hazan E (2011) Approximating semidefinite programs in sublinear time. Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F, Weinberger KQ, eds. *Advances in Neural Information Processing Systems 24 (NIPS 2011)* (Curan Associates, Inc., Red Hook, NY), 1080–1088.

Garber D, Meshi O (2016) Linear-memory and decomposition-invariant linearly convergent conditional gradient algorithm for structured polytopes. Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R, eds. *Advances in Neural Information Processing Systems 29 (NIPS 2016)* (Curan Associates, Inc., Red Hook, NY), 1001–1009.

Gärtner B, Jaggi M (2009) Coresets for polytope distance. *Proc. 25th Annual Sympos. Comput. Geometry* (ACM, New York), 33–42.

Gidel G, Jebara T, Lacoste-Julien S (2016) Frank-Wolfe algorithms for saddle point problems. arXiv preprint arXiv:1610.07797.

Gilbert EG (1966) An iterative procedure for computing the minimum of a quadratic form on a convex set. *SIAM J. Control* 4(1):61–80.

Grbovic M, Dance CR, Vucetic S (2012) Sparse principal component analysis with constraints. *Proc. 26th AAAI Conf. Artificial Intelligence* (AAAI, Palo Alto, CA), 935–941.

Günlük O, Linderoth J (2012) Perspective reformulation and applications. Lee J, Leyffer S, eds. *Mixed Integer Nonlinear Programming*, The IMA Volumes in Mathematics and its Applications, vol. 154 (Springer, New York), 61–89.

Har-Peled S, Kushal A (2005) Smaller coresets for *k*-median and *k*-means clustering. *Proc. 21st Annual Sympos. Comput. Geometry* (ACM, New York), 126–134.

Har-Peled S, Roth D, Zimak D (2007) Maximum margin coresets for active and noise tolerant learning. *Proc. 20th Internat. Joint Conf. Artifical Intelligence* (Morgan Kaufmann Publishers Inc., San Francisco), 836–841.

Hazan E, Kale S (2012) Projection-free online learning. *Proc. 29th Internat. Conf. Machine Learn.* (Omnipress, Madison, WI), 1843–1850.

Hiriart-Urruty JP, Lemaréchal C (1993) *Convex Analysis and Minimization Algorithms I: Fundamentals*, Grundlehren der mathematischen Wissenschaften, vol. 305 (Springer-Verlag, Berlin, Heidelberg).

Huggins J, Campbell T, Broderick T (2016) Coresets for scalable Bayesian logistic regression. Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R, eds. *Advances in Neural Information Processing Systems 29 (NIPS 2016)* (Curan Associates, Inc., Red Hook, NY), 4080–4088.

Jaggi M (2011) Sparse convex optimization methods for machine learning. PhD thesis, ETH Zurich, Zurich.

Jaggi M (2013) Revisiting Frank-Wolfe: Projection-free sparse convex optimization. *Proc. Machine Learn. Res.* 28(1):427–435.

Kolmogorov V (2018) Commutativity in the algorithmic Lovasz local lemma. *SIAM J. Comput.* 47(6):2029–2056.

Lan G (2013) The complexity of large-scale convex programming under a linear optimization oracle. Working paper, Georgia Institute of Technology, Atlanta.

Lan G, Nemirovski A, Shapiro A (2012) Validation analysis of mirror descent stochastic approximation method. *Math. Programming* 134(2): 425–458.

Lee SC, Kim YH (2002) An enhanced Lagrangian neural network for the ELD problems with piecewise quadratic cost functions and nonlinear constraints. *Electric Power Systems Res.* 60(3):167–177.

Lee S, Wright SJ (2012) Manifold identification in dual averaging for regularized stochastic online learning. *J. Machine Learn. Res.* 13(1): 1705–1744.

Liu H, Palatucci M, Zhang J (2009) Blockwise coordinate descent procedures for the multi-task lasso, with applications to neural semantic basis discovery. *Proc. 26th Annual Internat. Conf. Machine Learn.* (ACM, New York), 649–656.

Mangasarian OL (2000) Generalized support vector machines. Smola AJ, Bartlett P, Schölkopf B, Schuurmans D, eds. *Advances in Large Margin Classifiers* (MIT Press, Cambridge, MA), 135–146.

Nathan V, Raghvendra S (2014) Accurate streaming support vector machines. Working paper, Massachusetts Institute of Technology, Cambridge, MA.

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

Nesterov Y (2009) Primal-dual subgradient methods for convex problems. *Math. Programming* 120(1)221–259.

Nesterov Y (2013) *Introductory Lectures on Convex Optimization: A Basic Course*, Applied Optimization, vol. 87 (Springer Science+Business Media, New York).

Nesterov Y (2018) Complexity bounds for primal-dual methods minimizing the model of objective function. *Math. Programming* 171(1–2): 311–330.

Parikh N, Boyd SP (2014) Proximal algorithms. *Foundations Trends Optim.* 1(3):127–239.

Pennanen T (2012) Introduction to convex optimization in financial markets. *Math. Programming* 134(1):157–186.

Pierucci F, Harchaoui Z, Malick J (2014) A smoothing approach for composite conditional gradient with nonsmooth loss. Research Report 8662, INRIA Grenoble, Montbonnot-Saint-Martin, France.

Recht B, Ré C, Wright SJ, Niu F (2011) Hogwild: A lock-free approach to parallelizing stochastic gradient descent. Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F, Weinberger KQ, eds. *Advances in Neural Information Processing Systems 24 (NIPS 2011)* (Curran Associates, Inc., Red Hook, NY), 693–701.

Robinson SM (2012) Convexity in finite dimensional spaces. Technical Report, University of Wisconsin–Madison, Madison.

Rudelson M, Vershynin R (2007) Sampling from large matrices: An approach through geometric functional analysis. *J. ACM* 54(4):Article No. 21.

Sarlos T (2006) Improved approximation algorithms for large matrices via random projections. *Proc. 47th Annual IEEE Sympos. Foundations Comput. Sci.* (IEEE Computer Society, Washington, DC), 143–152.

Tsang IW, Kwok JT, Cheung PM (2005) Core vector machines: Fast SVM training on very large data sets. *J. Machine Learn. Res*. 8:291–301.

White DJ (1993) Extension of the Frank-Wolfe algorithm to concave nondifferentiable objective functions. *J. Optim. Theory Appl.* 78(2): 283–301.

Yuan GX, Chang KW, Hsieh CJ, Lin CJ (2010) A comparison of optimization methods and software for large-scale L1-regularized linear classification. *J. Machine Learn. Res*. 11:3188–3234.