Journal of Machine Learning Research 23 (2022) 1-41 Submitted 2/19; Revised 12/21; Published 7/22

Solving L1-regularized SVMs and Related Linear Programs: Revisiting the Effectiveness of Column and Constraint Generation

Antoine Dedieu ANTOINE@VICARIOUS.COM
Vicarious AI
Union City, CA 94587, USA

Rahul Mazumder RAHULMAZ@MIT.EDU
MIT Sloan School of Management and Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Haoyue Wang HAOYUEW@MIT.EDU
Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Editor: Mark Schmidt

Abstract

The linear Support Vector Machine (SVM) is a classic classification technique in machine learning. Motivated by applications in high dimensional statistics, we consider penalized SVM problems involving the minimization of a hinge-loss function with a convex sparsity-inducing regularizer such as: the L1-norm on the coefficients, its grouped generalization and the sorted L1-penalty (aka Slope). Each problem can be expressed as a Linear Program (LP) and is computationally challenging when the number of features and/or samples is large—the current state of algorithms for these problems is rather nascent when compared to the usual L2-regularized linear SVM. To this end, we propose new computational algorithms for these LPs by bringing together techniques from (a) classical column (and constraint) generation methods and (b) first order methods for non-smooth convex optimization—techniques that appear to be rarely used together for solving large scale LPs. These components have their respective strengths; and while they are found to be useful as separate entities, they appear to be more powerful in practice when used together in the context of solving large-scale LPs such as the ones studied herein. Our approach complements the strengths of (a) and (b)—leading to a scheme that seems to significantly outperform commercial solvers as well as specialized implementations for these problems. We present numerical results on a series of real and synthetic data sets demonstrating the surprising effectiveness of classic column/constraint generation methods in the context of challenging LP-based machine learning tasks.

Keywords: large scale linear programming, ℓ_1-penalization, slope, hinge loss, column generation, constraint generation, first order methods

1. Introduction

The linear Support Vector Machine (SVM) (Vapnik, 2013; Hastie et al., 2009) is a fundamental tool for binary classification. Given training data $(\mathbf{x}_i, y_i)_{i=1}^n$ with feature vector $\mathbf{x}_i \in \mathbb{R}^p$ and label $y_i \in \{-1, 1\}$, the task is to learn a linear classifier of the form $\text{sign}(\mathbf{x}^T\boldsymbol{\beta} + \beta_0)$ where, $\beta_0 \in \mathbb{R}$ is the offset term. The popular L2-regularized linear SVM (aka L2-SVM) considers the minimization problem

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p, \beta_0\in\mathbb{R}} \sum_{i=1}^n \big(1 - y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0)\big)_+ + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2, \tag{1}$$

where, $(a)_+ := \max\{a, 0\}$ is often noted as the hinge-loss function; and $\lambda \geq 0$ regularizes the L2-norm of the coefficients $\boldsymbol{\beta}$. Several algorithms have been proposed to efficiently solve Problem (1). Popular approaches include stochastic subgradient methods on the primal form (Bottou, 2010; Shalev-Shwartz et al., 2007), coordinate descent methods on a dual (Hsieh et al., 2008) and cutting plane algorithms (Joachims, 2006; Franc and Sonnenburg, 2008).

The L1-SVM estimator: The L2-SVM estimator generally leads to a dense estimate for $\boldsymbol{\beta}$—towards this end, the L1 penalty (Bradley and Mangasarian, 1998; Hastie et al., 2009) is often used as a convex surrogate to encourage sparsity (i.e., few nonzeros) in the coefficients. This leads to one of the problems we consider in this paper, namely, the L1-SVM problem:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p, \beta_0\in\mathbb{R}} \sum_{i=1}^n \big(1 - y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0)\big)_+ + \lambda\|\boldsymbol{\beta}\|_1, \tag{2}$$

which can be written as a Linear Program (LP), as shown in Section 2.2. The regularization parameter $\lambda \geq 0$ controls the L1-norm of $\boldsymbol{\beta}$. Off-the-shelf solvers, including commercial LP solvers (e.g., Gurobi, Cplex) work very well for small/moderate sized problems, but become expensive in solving Problem (2) when n and/or p is large, such as when $n \approx p \approx 10^4$; or $p \approx 10^6$ and n is a few hundred. Some high-quality specialized solvers for Problem (2) include: a homotopy based method to compute the entire piecewise linear regularization path in $\boldsymbol{\beta}$ (Hastie et al., 2004); methods based on Alternating Direction Method of Multipliers (ADMM) or operator splitting (Balamurugan et al., 2016; O'Donoghue et al., 2019). Pang et al. (2017) proposes a parametric simplex (PSM) approach to solve Problem (2) leading to a pair of primal/dual solutions at optimality. The authors demonstrate that their method achieves state-of-the-art performance for some LP-based sparse learning tasks[1] compared to a benchmark ADMM-based implementation "flare" (Li et al., 2015). Our experiments suggest that for problems with small $n \approx 100$ and large $p \approx 50,000$, PSM works well. However, it becomes inefficient as soon as n is large: for example, with $n \approx 10^4$ and $p \approx 100$, PSM can take hours and require large amounts of memory, while the methods we propose here take a few seconds with minimal memory. Mangasarian (2006) propose a perturbation approach that reformulates the L1-SVM problem as an unconstrained smooth minimization problem by including an additional regularization term. They apply Newton-type methods

1. The largest example studied in this work is the Dantzig Selector problem with $n = 200$ samples and $p = 5,000$ features.

to solve the resulting problem. Such Newton-type methods as discussed in Mangasarian (2006) require expensive matrix inversions.

In this paper, our goal is to propose new computational algorithms for the L1-SVM LP by revisiting classical operations research tools such as column and constraint generation with origins in 1950s (Ford Jr. and Fulkerson, 1958)—these methods appear to have been somewhat underutilized in the context of the L1-SVM problem and relatives of the L1-penalty, that we consider here. To improve the performance of column/constraint generation-based methods we use relatively recent first order convex optimization techniques.

We note that there are several appealing L1-regularized classifiers and efficient algorithms that consider a smooth loss function (e.g, logistic, squared hinge loss, etc)—see for example, Friedman et al. (2010a); Yuan et al. (2010). Different loss functions have different operating characteristics: in particular, smooth loss functions lead to estimators that are different from the hinge loss as in the L1-SVM problem (Hastie et al., 2009). Our goal in this paper is not to pursue an empirical analysis of the relative merits/de-merits of different loss functions which have been documented in earlier literature. Rather, we focus on algorithms for the hinge loss function with penalty functions that are representable as linear programs.

The Group-SVM estimator: In several applications, sparsity is structured—the coefficient indices are naturally found to occur in groups that are known a-priori and it is desirable to select or, set to zero, a whole group together as a "unit". In this context, a group version of the usual L1 norm is often used to improve the performance and interpretability of the model (Yuan and Lin, 2006; Huang and Zhang, 2010). We consider the popular L1/L_∞ penalty (Bach et al., 2011) leading to the *Group-SVM* Problem:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p,\ \beta_0\in\mathbb{R}} \sum_{i=1}^{n} \left(1 - y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0)\right)_+ + \lambda \sum_{g=1}^{G} \|\boldsymbol{\beta}_g\|_\infty \tag{3}$$

where, $g = 1, \ldots, G$ denotes a group index and the groups are disjoint. In addition, $\boldsymbol{\beta}_g$ denotes the subvector of coefficients belonging to group g and $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_G)$. Problem (3) can be expressed as an LP and our approach with suitable modifications, applies to this problem as well.

The Slope-SVM estimator: The third problem we study in this paper is of a different flavor and is inspired by the sorted L1-penalty aka the Slope norm (Bogdan et al., 2015; Bellec et al., 2018), popularly used in the context of penalized least squares problems for its useful statistical properties. For a pre-specified sequence of tuning parameters $\lambda_1 \geq \ldots \geq \lambda_p \geq 0$, the Slope-SVM problem is given by:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p,\ \beta_0\in\mathbb{R}} \sum_{i=1}^{n} \left(1 - y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0)\right)_+ + \sum_{j=1}^{p} \lambda_j |\beta_{(j)}|, \tag{4}$$

where $|\beta_{(1)}| \geq \ldots \geq |\beta_{(p)}|$ are the ordered values of $|\beta_i|, i = 1, \ldots, p$. Unlike Problems (2) and (3) where the penalty function is separable or block separable, the penalty function in (4) is not separable in the coefficients. We show in Section 3 that Problem (4) can be expressed as an LP with $O(n+p)$ variables and an exponential number (in p) of constraints,

consequently posing challenges in optimization. Despite the large number of constraints, we show that the LP can be solved with column/constraint generation. We note that using standard reformulation methods (Boyd and Vandenberghe, 2004, See Section A.1), Problem (4) can for example, be modeled with `CVXPY` and solved using a commercial solver like Gurobi for small-sized problems. However, the computations become expensive when λ_is are distinct which is the case in Bogdan et al. (2015)—for these cases, `CVXPY` can handle problems up to $n \approx 100$, $p \approx 200$ whereas, our approach can solve problems with $p \approx 50,000$ within a few seconds.

First order methods: First order methods (Nesterov, 2004) have enjoyed great success in solving large scale structured convex optimization problems arising in machine learning applications. Methods such as proximal gradient and its accelerated variants for example, are appealing candidates for the minimization of smooth functions and also problems of the composite form (Nesterov, 2013), wherein accelerated gradient methods enjoy a convergence rate of $O(1/\sqrt{\epsilon})$ to obtain an ϵ-accurate solution. For the nonsmooth SVM problems i.e., Problems (2), (3) and (4) discussed above, Nesterov's smoothing method (Nesterov, 2005) which replaces the hinge-loss with a smooth approximation can be used to obtain algorithms with a convergence rate of $O(1/\epsilon)$. This approach is explored in Section 4. While this procedure along with additional screening heuristics (Tibshirani et al., 2012) can lead to low accuracy solutions relatively fast; in our experience, the basic version of this algorithm takes a long time to obtain a solution with higher accuracy when n and/or p are large. Similarly, first order methods based on Becker et al. (2011) and O'Donoghue et al. (2019); Balamurugan et al. (2016) also appear to experience increased run times as the problem sizes become large.

What this paper is about: In this paper, we propose an efficient algorithmic framework for L1-SVM, Group-SVM and Slope-SVM using tools in column/constraint generation that make use of some basic structural properties of solutions to these problems, as we discuss below.

Note that large values of λ will encourage an optimal solution to Problem (2), $\hat{\boldsymbol{\beta}}$ (say), to be sparse. This sparsity will be critical to solve Problem (2) when $p \gg n$—we anticipate to solve Problem (2) without having to create an LP model with all p variables. To this end, we use column generation, a classical method in mathematical optimization/operations research originating in the context of solving integer programs during late 1950s (Ford Jr. and Fulkerson, 1958; Dantzig and Wolfe, 1960)—See also Desrosiers and Lübbecke (2005) for a nice review on this topic. We also make use of another structural aspect of a solution to Problem (2) when n is large and p is small. Suppose most of the samples can be classified correctly via a linear classifier—then, at an optimal solution, $1 \leq y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0)$ and hence $\tilde{\alpha}_i := (1 - y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0))_+$ will be zero for many indices $i = 1, \ldots, n$. We leverage this sparsity in $\tilde{\alpha}_i$'s to develop efficient algorithms for Problem (2), using constraint generation (Ford Jr. and Fulkerson, 1958; Desrosiers and Lübbecke, 2005) methods. This allows us to solve (2) without explicitly creating an LP model with n samples.

To summarize, there are two characteristics special to an optimal solution of Problem (2): (a) sparsity in the SVM coefficients, i.e., $\boldsymbol{\beta}$ and/or (b) sparsity in $\tilde{\alpha}_i$'s. Column generation can be used to handle (a); constraint generation can be used to address (b)—in problems where both n, p are large, we propose to combine both column and constraint generation.

To our knowledge, while column generation and constraint generation are used separately in the context of solving large scale LPs, using them together, in the context of the L1-SVM problem is novel. For solving these, usually small, subproblems we rely on powerful LP solvers such as the simplex based algorithms of Gurobi—they lead to a pair of primal-dual solutions and also possess excellent warm-starting capabilities. Our approach applies to the Group-SVM Problem (3) with suitable modifications. We also extend our approach to handle the Slope-SVM problem (4), which requires a fairly involved use of column/constraint generation. Numerical evidence presented here suggests that column/constraint generation methods are modular, simple and powerful tools—they should perhaps be considered more frequently to solve machine learning tasks based on LPs, even beyond the ones studied here.

The column/constraint generation methods mentioned above, are found to benefit from a good initialization. To this end, we use first order optimization methods to get approximate solutions with low computational cost. These solutions serve as decent initializations and are subsequently improved to deliver optimal solutions as a part of our column and/or constraint generation framework. This approach is found to be useful in all the three problems studied here.

To our knowledge, our work is novel in that we bring together first order methods in convex optimization and column/constraint generation algorithms for solving large scale LPs, in the context of solving a problem of key importance in machine learning. A Python implementation of our methods can be found at: `https://github.com/wanghaoyue123/Column-and-constraint-generation-for-L1-SVM-and-cousins`.

Organization of paper: The rest of this paper is organized as follows. Section 2 presents an overview of column/constraint generation methods; and then discusses their instantiation for the L1-SVM and Group-SVM problems. Section 3 discusses the Slope-SVM problem. Section 4 discusses how first order methods can be used to get approximate solutions for these problems. Section 5 presents numerical results.

Notation: For an integer a we use $[a]$ to denote $\{1, 2, \ldots, a\}$. The ith entry of a vector $\mathbf{u}$ is denoted by u_i. For a set $\mathcal{A}$, we use the notation $|\mathcal{A}|$ to denote its size. For a positive semidefinite matrix $\mathbf{A}$, we denote its largest eigenvalue by $\sigma_{\max}(\mathbf{A})$. For a vector $\mathbf{x} \in \mathbb{R}^n$ and a subset $\mathcal{B} \subseteq [n]$, we let $\mathbf{x}_\mathcal{B}$ denote the sub-vector of $\mathbf{x}$ corresponding to the indices in the set $\mathcal{B}$.

2. Column and constraint generation for L1-SVM and its group extension

As we allude to earlier, column and constraint generation algorithms have a long history in mathematical optimization and operations research (Ford Jr. and Fulkerson, 1958; Dantzig and Wolfe, 1960). Here we present an outline of these methods for a generic LP. We subsequently discuss their applications to Problems (2) and (3).

2.1 Methodology for Column and Constraint Generation

The basic idea of column generation is to start with a candidate set of columns and incrementally add new columns into the model until some optimality conditions are met. Consider the *primal* LP problem **(P)** where, $\bar{n}, \bar{p}$ are integers and $\mathbf{A} \in \mathbb{R}^{\bar{n} \times \bar{p}}$, $\mathbf{b} \in \mathbb{R}^{\bar{n}}$, $\mathbf{c} \in \mathbb{R}^{\bar{p}}$ are problem data. We assume that the optimal objective value of **(P)** is finite. The strong

duality theorem (cf Bertsimas and Tsitsiklis (1997), Theorem 4.4) states that this optimum is equal to that of the *dual* problem **(D)**:

$$(\mathbf{P}): \quad \min_{\boldsymbol{\theta}\in\mathbb{R}^{\bar{p}}} \quad \mathbf{c}^T\boldsymbol{\theta} \qquad\qquad (\mathbf{D}): \quad \max_{\mathbf{q}\in\mathbb{R}^{\bar{n}}} \quad \mathbf{q}^T\mathbf{b}$$
$$\text{s.t.} \quad \mathbf{A}\boldsymbol{\theta} \geq \mathbf{b}, \quad \boldsymbol{\theta} \geq \mathbf{0} \qquad\qquad \text{s.t.} \quad \mathbf{q}^T\mathbf{A} \leq \mathbf{c} \quad \mathbf{q} \geq \mathbf{0}.$$

Let us consider the case where $\bar{p}$ is large compared to $\bar{n}$ and we anticipate an optimal solution of $(\mathbf{P})$ to have few nonzeros. Let $\mathbf{A}_j$ denote the jth column of $\mathbf{A}$. Consider a subset of columns $\mathcal{B} \subset [\bar{p}]$ and the corresponding *reduced* primal/dual problems:

$$(\mathbf{P}_\mathcal{B}): \quad \min_{\boldsymbol{\theta}_\mathcal{B}\in\mathbb{R}^{|\mathcal{B}|}} \quad \sum_{j\in\mathcal{B}} c_j\theta_j \qquad\qquad (\mathbf{D}_\mathcal{B}): \quad \max_{\mathbf{q}\in\mathbb{R}^{\bar{n}}} \quad \mathbf{q}^T\mathbf{b}$$
$$\text{s.t.} \quad \sum_{j\in\mathcal{B}} \mathbf{A}_j\theta_j \geq \mathbf{b}, \quad \boldsymbol{\theta}_\mathcal{B} \geq \mathbf{0} \qquad\qquad \text{s.t.} \quad \mathbf{q}^T\mathbf{A}_j \leq c_j, \ j \in \mathcal{B}, \quad \mathbf{q} \geq \mathbf{0}.$$

Let $\tilde{\boldsymbol{\theta}}_\mathcal{B}$ and $\tilde{\mathbf{q}}$ be a pair of primal/dual solutions to the restricted problems $(\mathbf{P}_\mathcal{B})$ and $(\mathbf{D}_\mathcal{B})$. Let $\hat{\boldsymbol{\theta}}$ be an extension of $\tilde{\boldsymbol{\theta}}_\mathcal{B}$ to $\mathbb{R}^{\bar{p}}$ i.e., $\hat{\boldsymbol{\theta}}_\mathcal{B} = \tilde{\boldsymbol{\theta}}_\mathcal{B}$ and $\hat{\boldsymbol{\theta}}_{\mathcal{B}^c} = \mathbf{0}$. While $\hat{\boldsymbol{\theta}}$ is a feasible solution for $(\mathbf{P})$, it may not be optimal for $(\mathbf{P})$. An optimality certificate can be obtained via $\tilde{\mathbf{q}}$, by checking if $\tilde{\mathbf{q}}$ is feasible for $(\mathbf{D})$. Specifically, let the *reduced cost* for variable j be defined as $\bar{c}_j := c_j - \tilde{\mathbf{q}}^T\mathbf{A}_j$. If $\bar{c}_j \geq 0$ for all $j \notin \mathcal{B}$, then $\tilde{\mathbf{q}}$ is optimal for $(\mathbf{D})$; and $\hat{\boldsymbol{\theta}}$ is an optimal solution for $(\mathbf{P})$.

Column generation: Column generation uses the principle outlined above. It is expected to be useful when $\bar{p} \gg \bar{n}$, and an optimal solution to $(\mathbf{P})$ has few nonzeros. We start with a subset of columns say, $\mathcal{B}$—i.e., a guess for the support of a minimizer of $(\mathbf{P})$ for which $(\mathbf{P}_\mathcal{B})$ is feasible. Given $\mathcal{B}$, we solve the *restricted* problem $(\mathbf{P}_\mathcal{B})$; and have a pair of primal/dual solutions for $(\mathbf{P}_\mathcal{B})/(\mathbf{D}_\mathcal{B})$. If all the reduced costs are nonnegative, we declare convergence and stop. Otherwise, we find a column (or a collection of columns) outside $\mathcal{B}$ with the most negative reduced cost(s), update $\mathcal{B}$, and re-solve the updated problem $(\mathbf{P}_\mathcal{B})$ by making use of the warm-start capabilities of a simplex-based LP solver. If $\mathcal{B}$ is only allowed to increase—i.e., we do not drop variables—then this process converges after finitely many iterations. Convergence guarantees of this procedure are formally discussed in Section 2.3. Upon termination, column generation leads to a pair of primal/dual optimal solutions to $(\mathbf{P})/(\mathbf{D})$.

Constraint generation: We now consider the case when $\bar{n} \gg \bar{p}$. Suppose at an optimal solution to **(P)**, only a small fraction of the $\bar{n}$ constraints $\mathbf{a}_i^T\boldsymbol{\theta} \geq b_i$ for $i \in [\bar{n}]$ are active or binding. Then, an optimal solution can be potentially obtained by considering only a small subset of the $\bar{n}$ constraints. This inspires the use of a constraint generation algorithm, which can also be interpreted as column generation (Bertsimas and Tsitsiklis, 1997) on the dual Problem **(D)**.

2.2 Primal and Dual Formulations of L1-SVM

We present an LP formulation for Problem (2):

$$\boxed{\mathcal{P}_\lambda([n],[p])} \qquad \min_{\substack{\boldsymbol{\xi}\in\mathbb{R}^n,\beta_0\in\mathbb{R}\\ \boldsymbol{\beta}^+,\,\boldsymbol{\beta}^-\in\mathbb{R}^p}} \quad \sum_{i=1}^n \xi_i + \lambda\sum_{j=1}^p \beta_j^+ + \lambda\sum_{j=1}^p \beta_j^- \tag{5a}$$

$$\text{s.t.} \quad \xi_i + y_i\mathbf{x}_i^T\boldsymbol{\beta}^+ - y_i\mathbf{x}_i^T\boldsymbol{\beta}^- + y_i\beta_0 \geq 1 \quad i\in[n] \tag{5b}$$

$$\boldsymbol{\xi}\geq 0,\ \boldsymbol{\beta}^+\geq 0,\ \boldsymbol{\beta}^-\geq 0.$$

Above, the positive and negative parts of β_i are denoted as $\beta_i^+ = \max\{\beta_i, 0\}$ and $\beta_i^- = \max\{-\beta_i, 0\}$ respectively, and ξ_i's are auxiliary continuous variables corresponding to the hinge-loss function. The feasible set of Problem (5) is nonempty. A dual of (5) is the following LP:

$$\boxed{\mathcal{D}_\lambda([n],[p])} \qquad \max_{\boldsymbol{\pi}\in\mathbb{R}^n} \quad \sum_{i=1}^n \pi_i$$

$$\text{s.t.} \quad -\lambda \leq \sum_{i=1}^n y_i x_{ij}\pi_i \leq \lambda \qquad j\in[p] \tag{6}$$

$$\mathbf{y}^T\boldsymbol{\pi} = 0$$

$$0\leq \pi_i \leq 1 \qquad i\in[n].$$

For Problems (5) and (6), standard complementary slackness conditions lead to:

$$(1-\pi_i)\xi_i = 0, \qquad \pi_i\left(\xi_i + y_i\mathbf{x}_i^T\boldsymbol{\beta} + y_i\beta_0 - 1\right) = 0 \qquad i\in[n]. \tag{7}$$

Let $(\boldsymbol{\beta}^*(\lambda), \beta_0^*(\lambda))$ and $\boldsymbol{\pi}^*(\lambda)$ denote optimal solutions for Problems (5) and (6). In what follows, for notational convenience, we will drop the dependence of an optimal solution on λ when there is no confusion. We make a few observations regarding the geometry of an L1-SVM solution following standard SVM terminology (Hastie et al., 2009). For easier notation, we denote $\alpha_i = y_i\mathbf{x}_i^T(\boldsymbol{\beta}^+ - \boldsymbol{\beta}^-) + y_i\beta_0$ for all i. Note that $\xi_i = \max\{1-\alpha_i, 0\}$ for all i. If a point i is correctly classified, we have $\xi_i = 0$; and if this point is away from the margin, then we have $0 > 1-\alpha_i$ and hence $\pi_i = 0$ (from (7)). Note that if point i is misclassified, then $\xi_i > 0$ and $\pi_i = 1$. Furthermore, based on the value of π_i, we have the following cases: (i) If $\pi_i = 0$, then $\alpha_i \geq 1$; (ii) If $\pi_i = 1$ then $1\geq\alpha_i$ and (iii) If $\pi_i\in(0,1)$ then $\alpha_i = 1$. The SVM coefficients can be estimated from the samples lying on the margin i.e., for all i such that $\alpha_i = 1$. In particular, if an optimal solution to the L1-SVM problem has κ-many nonzeros in $\boldsymbol{\beta}$, then $(\boldsymbol{\beta}, \beta_0)$ can be computed based on $(\kappa+1)$-many samples lying on the margin[2].

2. Note that here we assume that the corresponding feature columns form a full rank matrix.

2.3 Column and Constraint Generation for L1-SVM

We discuss how column and constraint generation applies to the L1-SVM Problem (2) for a given λ. Given a set of candidate features $\mathcal{J} \subseteq [p]$ and a subset of samples $\mathcal{I} \subseteq [n]$, we form the *column and constraint restricted* L1-SVM problem as

$$\boxed{\mathcal{P}_\lambda(\mathcal{I},\mathcal{J})} \quad \begin{array}{ll} \min\limits_{\substack{\boldsymbol{\xi}\in\mathbb{R}^{|\mathcal{I}|},\ \beta_0\in\mathbb{R} \\ \boldsymbol{\beta}^+,\ \boldsymbol{\beta}^-\in\mathbb{R}^{|\mathcal{J}|}}} & \sum\limits_{i\in\mathcal{I}} \xi_i + \lambda \sum\limits_{j\in\mathcal{J}} \beta_j^+ + \lambda \sum\limits_{j\in\mathcal{J}} \beta_j^- \\ \text{s.t.} & \xi_i + \sum\limits_{j\in\mathcal{J}} y_i x_{ij}\beta_j^+ - \sum\limits_{j\in\mathcal{J}} y_i x_{ij}\beta_j^- + y_i\beta_0 \geq 1 \qquad i\in\mathcal{I} \\ & \boldsymbol{\xi}\geq 0,\ \boldsymbol{\beta}^+\geq 0,\ \boldsymbol{\beta}^-\geq 0, \end{array} \tag{8}$$

and the corresponding dual problem is

$$\boxed{\mathcal{D}_\lambda(\mathcal{I},\mathcal{J})} \quad \begin{array}{ll} \max\limits_{\boldsymbol{\pi}\in\mathbb{R}^{|\mathcal{I}|}} & \sum\limits_{i\in\mathcal{I}} \pi_i \\ \text{s.t.} & -\lambda \leq \sum\limits_{i\in\mathcal{I}} y_i x_{ij}\pi_i \leq \lambda \qquad j\in\mathcal{J} \\ & \sum\limits_{i\in\mathcal{I}} y_i\pi_i = 0 \\ & 0\leq \pi_i \leq 1 \qquad i\in\mathcal{I}. \end{array} \tag{9}$$

Let $(\hat{\beta}_j^+,\hat{\beta}_j^-)_{j\in\mathcal{J}}$, $\hat{\beta}_0$ and $(\hat{\xi}_i)_{i\in\mathcal{I}}$ be an optimal solution of $\mathcal{P}_\lambda(\mathcal{I},\mathcal{J})$, and let $(\hat{\pi}_i)_{i\in\mathcal{I}}$ be an optimal solution of $\mathcal{D}_\lambda(\mathcal{I},\mathcal{J})$. Define $\hat{\boldsymbol{\beta}}^+$ and $\hat{\boldsymbol{\beta}}^-$ to be the vectors in $\mathbb{R}^p$ which are obtained from $(\hat{\beta}_j^+,\hat{\beta}_j^-)_{j\in\mathcal{J}}$ by padding coordinates outside $\mathcal{J}$ with zeros. Similarly, define $\hat{\boldsymbol{\xi}}$ and $\hat{\boldsymbol{\pi}}$ to be the vectors in $\mathbb{R}^p$ obtained from $(\hat{\xi}_i)_{i\in\mathcal{I}}$ and $(\hat{\pi}_i)_{i\in\mathcal{I}}$ respectively by padding coordinates outside $\mathcal{I}$ with zeros. Using this notation, the tuple $(\hat{\boldsymbol{\beta}},\hat{\beta}_0)$ is a candidate approximate solution to the L1-SVM problem i.e., Problem (2). (Recall that, $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}^+ - \hat{\boldsymbol{\beta}}^-$, and the candidate solution $(\hat{\boldsymbol{\beta}},\hat{\beta}_0)$ is obtained from the column-and-constraint restricted Problem (8)).

For $j\in[p]\setminus\mathcal{J}$ and $i\in[n]\setminus\mathcal{I}$, define:

$$\begin{aligned} &\bar{\beta}_j^+ = \lambda - \sum_{i\in\mathcal{I}} y_i x_{ij}\hat{\pi}_i, \qquad \bar{\beta}_j^- = \lambda + \sum_{i\in\mathcal{I}} y_i x_{ij}\hat{\pi}_i, \\ &\bar{\pi}_i = 1 - y_i\Big(\sum_{j\in\mathcal{J}} x_{ij}\hat{\beta}_j + \hat{\beta}_0\Big). \end{aligned} \tag{10}$$

Above, $\bar{\beta}_j^+$, $\bar{\beta}_j^-$ are reduced costs of variables β_j^+ and β_j^- respectively. A similar notation applies to $\bar{\pi}_i$. Note that if

$$\min\Big\{\bar{\beta}_j^+,\bar{\beta}_j^-\Big\}\geq 0 \quad \forall\ j\in[p]\setminus\mathcal{J}; \quad \text{and} \quad \bar{\pi}_i\leq 0 \quad \forall\ i\in[n]\setminus\mathcal{I}, \tag{11}$$

then $(\hat{\boldsymbol{\beta}},\hat{\beta}_0,\hat{\boldsymbol{\xi}})$ is an optimal solution of $\mathcal{P}_\lambda([n],[p])$ and $\hat{\boldsymbol{\pi}}$ is an optimal solution of $\mathcal{D}_\lambda([n],[p])$. To see this, we first note that the condition in (11) ensures that $(\hat{\boldsymbol{\beta}},\hat{\beta}_0,\hat{\boldsymbol{\xi}})$ and $\hat{\boldsymbol{\pi}}$ are feasible solutions to $\mathcal{P}_\lambda([n],[p])$ and $\mathcal{D}_\lambda([n],[p])$ respectively. The objective value of $\mathcal{P}_\lambda([n],[p])$ at a

solution $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0, \hat{\boldsymbol{\xi}})$ is the same as the objective value of $\mathcal{P}_\lambda(\mathcal{I}, \mathcal{J})$ with $(\hat{\beta}_j^+, \hat{\beta}_j^-)_{j\in\mathcal{J}}$, $\hat{\beta}_0$ and $(\hat{\xi}_i)_{i\in\mathcal{I}}$. Also, the objective value of $\mathcal{D}_\lambda([n],[p])$ with $\hat{\boldsymbol{\pi}}$ is the same as the objective value of $\mathcal{D}_\lambda(\mathcal{I}, \mathcal{J})$ with $(\hat{\pi}_i)_{i\in\mathcal{I}}$. Hence, by strong duality we know that $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0, \hat{\boldsymbol{\xi}})$ is an optimal solution of $\mathcal{P}_\lambda([n],[p])$ and $\hat{\boldsymbol{\pi}}$ is an optimal solution of $\mathcal{D}_\lambda([n],[p])$.

When the condition in (11) does not hold, $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$ is not an optimal solution of Problem (2). The following theorem provides an upper bound on the primal optimality gap for the solution $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$.

Theorem 1 *Let z^* denote the optimal objective value of $\mathcal{P}_\lambda([n],[p])$. Denote*

$$\hat{z} := \sum_{i=1}^{n} \Big(1 - y_i(\mathbf{x}_i^T \hat{\boldsymbol{\beta}} + \hat{\beta}_0)\Big)_+ + \lambda \|\hat{\boldsymbol{\beta}}\|_1. \tag{12}$$

Using the notation in (10)*, let us define*

$$\tilde{\epsilon}_1 := \max_{i\in[n]\setminus\mathcal{I}} \{\max\{\bar{\pi}_i, 0\}\}, \quad \tilde{\epsilon}_2 := -\min_{j\in[p]\setminus\mathcal{J}} \Big\{\min\{\bar{\beta}_j^+, \bar{\beta}_j^-, 0\}\Big\}. \tag{13}$$

If $(\boldsymbol{\beta}^*, \beta_0^*)$ is an optimal solution to Problem (2)*, then the following holds*

$$z^* \le \hat{z} \le z^* + \tilde{\epsilon}_1(n - |\mathcal{I}|) + \tilde{\epsilon}_2 \|\boldsymbol{\beta}^*\|_1. \tag{14}$$

Proof *The first inequality follows by noting that z^* is the optimal objective value for Problem* (2)*. Below we prove the second inequality in* (14)*. By the definition of $\tilde{\epsilon}_2$ we have*

$$\lambda - \Big|\sum_{i=1}^{n} y_i x_{ij} \hat{\pi}_i\Big| = \lambda - \Big|\sum_{i\in\mathcal{I}} y_i x_{ij} \hat{\pi}_i\Big| \ge -\tilde{\epsilon}_2 \quad \forall j \in [p] \setminus \mathcal{J}. \tag{15}$$

On the other hand, since $(\hat{\pi}_i)_{i\in\mathcal{I}}$ is a feasible solution of $\mathcal{D}_\lambda(\mathcal{I}, \mathcal{J})$ we have

$$\lambda - \Big|\sum_{i=1}^{n} y_i x_{ij} \hat{\pi}_i\Big| = \lambda - \Big|\sum_{i\in\mathcal{I}} y_i x_{ij} \hat{\pi}_i\Big| \ge 0 \quad \forall j \in \mathcal{J}. \tag{16}$$

Combining the two inequalities above we have

$$-\lambda - \tilde{\epsilon}_2 \le \sum_{i=1}^{n} y_i x_{ij} \hat{\pi}_i \le \lambda + \tilde{\epsilon}_2 \quad \forall j \in [p]. \tag{17}$$

In addition, we have $\sum_{i=1}^n y_i \hat{\pi}_i = \sum_{i\in\mathcal{I}} y_i \hat{\pi}_i = 0$ and $0 \le \hat{\pi}_i \le 1$ for all $i \in [n]$. So $(\hat{\pi}_i)_{i=1}^n$ is a feasible solution for the problem $\mathcal{D}_{\lambda+\tilde{\epsilon}_2}([n],[p])$. Let $(\boldsymbol{\xi}^*, \boldsymbol{\beta}^{*+}, \boldsymbol{\beta}^{*-}, \beta_0^*)$ be an optimal solution of $\mathcal{P}_\lambda([n],[p])$, then it is also a feasible solution of $\mathcal{P}_{\lambda+\tilde{\epsilon}_2}([n],[p])$. Hence by weak duality we have

$$\sum_{i=1}^{n} \hat{\pi}_i \le \sum_{i=1}^{n} \xi_i^* + (\lambda + \tilde{\epsilon}_2) \sum_{j=1}^{p} (\beta_j^{*+} + \beta_j^{*-}) = z^* + \tilde{\epsilon}_2 \|\boldsymbol{\beta}^*\|_1. \tag{18}$$

On the other hand, by (10)*, we have*

$$\Big(1 - y_i(\mathbf{x}_i^T \hat{\boldsymbol{\beta}} + \hat{\beta}_0)\Big)_+ = \max\{\bar{\pi}_i, 0\} \quad \forall\ i \in [n] \setminus \mathcal{I}. \tag{19}$$

From the definition of $\hat{z}$ in (12)*, we have the following:*

$$\begin{aligned}\hat{z} &= \sum_{i\in\mathcal{I}}\Big(1-y_i(\mathbf{x}_i^T\hat{\boldsymbol{\beta}}+\hat{\beta}_0)\Big)_+ + \sum_{i\in[n]\setminus\mathcal{I}}\max\{\bar{\pi}_i,0\}+\lambda\|\hat{\boldsymbol{\beta}}\|_1\\ &\leq \sum_{i\in\mathcal{I}}\Big(1-y_i(\mathbf{x}_i^T\hat{\boldsymbol{\beta}}+\hat{\beta}_0)\Big)_+ + \tilde{\epsilon}_1(n-|\mathcal{I}|)+\lambda\|\hat{\boldsymbol{\beta}}\|_1\\ &= \sum_{i\in\mathcal{I}}\Big(1-y_i(\mathbf{x}_i^T\hat{\boldsymbol{\beta}}+\hat{\beta}_0)\Big)_+ + \tilde{\epsilon}_1(n-|\mathcal{I}|)+\lambda\sum_{j\in\mathcal{J}}|\hat{\beta}_j|, \qquad (20)\end{aligned}$$

where, the first line above uses (19)*, and the inequality above uses the definition of $\tilde{\epsilon}_1$.*

Since $\sum_{i\in\mathcal{I}}(1-y_i(\mathbf{x}_i^T\hat{\boldsymbol{\beta}}+\hat{\beta}_0))_+ + \lambda\sum_{j\in\mathcal{J}}|\hat{\beta}_j|$ is the optimal objective value of $\mathcal{P}_\lambda(\mathcal{I},\mathcal{J})$, it holds by strong duality

$$\sum_{i\in\mathcal{I}}\Big(1-y_i(\mathbf{x}_i^T\hat{\boldsymbol{\beta}}+\hat{\beta}_0)\Big)_+ + \lambda\sum_{j\in\mathcal{J}}|\hat{\beta}_j| \;=\; \sum_{i\in\mathcal{I}}\hat{\pi}_i = \sum_{i=1}^{n}\hat{\pi}_i. \qquad (21)$$

Combining (18)*,* (20) *and* (21) *we complete the proof.* ■

Theorem 1 provides an upper bound on the primal optimality gap $\hat{z}-z^*$ in terms of $\tilde{\epsilon}_1$ and $\tilde{\epsilon}_2$ — suggesting termination criteria for column-and-constraint generation.

Remark 2 *Note that the values $\tilde{\epsilon}_1$ and $\tilde{\epsilon}_2$ defined in* (13) *quantify the violation of the conditions in* (11)*. By definition, it holds $\tilde{\epsilon}_1 \geq 0$ and $\tilde{\epsilon}_2 \geq 0$. If $\tilde{\epsilon}_1 = \tilde{\epsilon}_2 = 0$, then the condition in* (11) *is satisfied, and the inequality in* (14) *reduces to $z^* = \hat{z}$.*

In the following, we discuss three important special cases of the column-and-constraint generation algorithm discussed above: (a) $n \ll p$; (b) $n \gg p$; and (c) $n \approx p$ with both n and p large.

2.3.1 Column Generation when $p \gg n$

For the case when p is much larger than n (and n is small), we set $\mathcal{I} = [n]$ in Problem (8) and consider a column generation method that adjusts $\mathcal{J}$ in each iteration. More precisely, in each iteration, we solve a subproblem $\mathcal{P}([n],\mathcal{J})$. Then we compute the reduced cost according to (10), and expand $\mathcal{J}$ by adding the indices j for which $\min\{\bar{\beta}_j^+,\bar{\beta}_j^-\}$ is lower than a pre-specified (negative) threshold. We summarize the algorithm below.

Algorithm 1: **Column generation for L1-SVM**
Input: $\mathbf{X}$, $\mathbf{y}$, regularization parameter λ, a convergence threshold $\epsilon \geq 0$, an initial set of columns $\mathcal{J}$.
Output: A near-optimal solution $(\hat{\boldsymbol{\beta}},\hat{\beta}_0)$ for the L1-SVM Problem (2).

1. Repeat Steps 2 to 3 until $\mathcal{J}$ stabilizes.
2. Solve the problem $\mathcal{P}_\lambda([n],\mathcal{J})$.

3. Form the set $\mathcal{J}^\epsilon := \{j \in [p] \backslash \mathcal{J} \mid \min\{\bar{\beta}_j^+, \bar{\beta}_j^-\} \leq -\epsilon\}$. Update $\mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{J}^\epsilon$; and go to Step 2 (with LP warm-starting enabled).

Algorithm 1 expands with no deletion the set of columns in the restricted problem—this converges in a finite number of iterations, bounded above by p. The cost of Algorithm 1 also depends upon the size of $\mathcal{J}$—if this becomes comparable to p, then the cost of solving $\mathcal{P}_\lambda([n], \mathcal{J})$ will be large. However, assuming that a solution to the L1-SVM problem corresponds to a sparse $\boldsymbol{\beta}$, then using a good initialization for $\mathcal{J}$, the worst case behavior may not be observed in practice—see our results in Section 5 for empirical support. In addition, thanks to simplex warm-start capabilities, one can compute a solution to the updated version of the restricted L1-SVM problem quite efficiently.

The following corollary presents a special case of Theorem 1 when we use Algorithm 1 (column generation).

Corollary 3 *Let $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$ be the solution obtained by Algorithm 1 and $\hat{z}$ be the corresponding objective value of Problem* (2)*, as defined in* (12)*. Let z^* be the optimal objective value and $(\boldsymbol{\beta}^*, \beta_0^*)$ be an optimal solution to Problem* (2)*. Then it holds that*

$$0 \leq \hat{z} - z^* \leq \epsilon \|\boldsymbol{\beta}^*\|_1.$$

There are variants of the column generation procedure where, instead of continually expanding the set of columns in the restricted problem, one can also drop variables—see Desrosiers and Lübbecke (2005). This is useful if the size of $\mathcal{J}$ becomes so large that the restricted problem becomes difficult to solve. If we only expand the set of columns in the restricted problem, as in Algorithm 1, we obtain a sequence of decreasing objective values across the column generation iterations. If one were to both add and delete columns, one may not obtain a monotone sequence of objective values. In this case, additional care is needed to ensure convergence of the resulting procedure (Desrosiers and Lübbecke, 2005).

Initializing column generation with a candidate set of columns: In practice, Algorithm 1 is found to benefit from a good initial choice for $\mathcal{J}$. To obtain a reasonable estimate of $\mathcal{J}$ with low computational cost, we list a couple of options that we found to be useful.

(i) *First order methods:* Section 4 discusses first order methods to obtain an approximate solution to the L1-SVM problem, which can be used to initialize $\mathcal{J}$.

(ii) *Regularization path:* We compute a path or grid of solutions to the L1-SVM problem using Algorithm 1, for a decreasing sequence of λ values: $\lambda \in \{\lambda_0, \ldots, \lambda_M\}$. This is discussed below.

Computing a regularization path with column generation. Note that the subgradient condition of optimality for the L1-SVM Problem (2) is given by: $\lambda \cdot \text{sign}(\beta_j^*) = \sum_{i=1}^n y_i x_{ij} \pi_i^*$ where, for a scalar u, $\text{sign}(u)$ denotes a subgradient of $u \mapsto |u|$. When λ is larger than $\lambda_{\max} := \max_{j \in [p]} \sum_{i=1}^n |x_{ij}|$, an optimal solution to Problem (2) is zero: $\boldsymbol{\beta}^*(\lambda) = \mathbf{0}$.

Let $\mathcal{I}_+$, $\mathcal{I}_-$ denote the sample indices corresponding to the classes with labels $+1$ and -1, respectively; and let N_+, N_- denote their respective sizes. If $N_+ \geq N_-$, then for

$\lambda \geq \lambda_{\max}$ a solution to Problem (6) is $\pi_i(\lambda) = N_-/N_+, \forall i \in \mathcal{I}_+$ and $\pi_i(\lambda) = 1, \forall i \in \mathcal{I}_-$. For $\lambda = \lambda_{\max}$ using (10), the minimum of the reduced costs of the variables β_j^+ and β_j^- is

$$\min\left\{\bar{\beta}_j^+(\lambda_{\max}), \bar{\beta}_j^-(\lambda_{\max})\right\} = \lambda_{\max} - \Big|\frac{N_-}{N_+}\sum_{i\in\mathcal{I}_+} y_i x_{ij} + \sum_{i\in\mathcal{I}_-} y_i x_{ij}\Big|. \tag{22}$$

When $\lambda = \lambda_1$ is slightly smaller than $\lambda_{\max}$, we initialize column generation by selecting $\mathcal{J}$ as a small subset of variables that minimize the right-hand side of (22). Once we obtain a solution to Problem (2) at λ_1, we compute a solution for a smaller value of λ by using LP warm-start along with column generation. Consequently, this leads to solutions for a grid of λ values, as summarized below.

Algorithm 2: **Regularization path algorithm for L1-SVM**
Input: $\mathbf{X}$, $\mathbf{y}$, convergence tolerance ϵ (for column generation), a grid of decreasing λ values: $\{\lambda_0 = \lambda_{\max}, \ldots, \lambda_M = \lambda\}$, a small integer j_0.
Output: A sequence of solutions $(\boldsymbol{\beta}^*(\lambda_i), \beta_0^*(\lambda_i))$ for $i = 0, \ldots, M$ on a grid of λ-values for the L1-SVM Problem (2).

1. Let $\boldsymbol{\beta}^*(\lambda_0) = \mathbf{0}$ and assign $\mathcal{J}(\lambda_0)$ to the j_0 variables minimizing the rhs of (22).

2. For $\ell \in \{1, \ldots, M\}$ initialize $\mathcal{J}(\lambda_\ell) \leftarrow \mathcal{J}(\lambda_{\ell-1})$, $\boldsymbol{\beta}^*(\lambda_\ell) \leftarrow \boldsymbol{\beta}^*(\lambda_{\ell-1})$ and $\beta_0^*(\lambda_\ell) \leftarrow \beta_0^*(\lambda_{\ell-1})$. Run column generation to obtain the new estimate $\boldsymbol{\beta}^*(\lambda_\ell)$ with $\mathcal{J}(\lambda_\ell)$ denoting the corresponding columns.

2.3.2 Constraint Generation when $n \gg p$

For the case when n is much larger than p (and p is small), we set $\mathcal{J} = [p]$ in Problem (8) and consider a constraint generation method that updates $\mathcal{I}$ in each iteration. In particular, we solve the subproblem $\mathcal{P}(\mathcal{I}, [p])$, and subsequently update $\mathcal{I}$ by adding the indices $i \in [n] \setminus \mathcal{I}$ with $\bar{\pi}_i$ larger than a threshold. We repeat this process till convergence, as formally summarized below.

Algorithm 3: **Constraint generation for L1-SVM**
Input: $\mathbf{X}$, $\mathbf{y}$, regularization parameter λ, a tolerance threshold $\epsilon \geq 0$, an initial set of constraints indexed by $\mathcal{I}$.
Output: A near-optimal solution $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$ for the L1-SVM Problem (2).

1. Repeat Steps 2 and 3 until $\mathcal{I}$ stabilizes.

2. Solve the problem $\mathcal{P}_\lambda(\mathcal{I}, [p])$.

3. Let $\mathcal{I}^\epsilon := \{i \in [n] \setminus \mathcal{I} \mid \bar{\pi}_i > \epsilon\}$. Update $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}^\epsilon$, and go to Step 2 (with LP warm-starting enabled).

The following corollary is a special case of Theorem 1 when we use Algorithm 3 (constraint generation), and provides an upper bound on the optimality gap of the solution computed by Algorithm 3.

Corollary 4 *Let $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$ be a solution obtained by Algorithm 3 and $\hat{z}$ be the corresponding objective value of Problem* (2) *as defined in* (12)*. Let $\hat{\mathcal{I}}$ be the index set delivered by Algorithm 3 upon termination, and let z^* be the optimal objective value of Problem* (2)*. Then it holds that*

$$0 \leq \hat{z} - z^* \leq (n - |\hat{\mathcal{I}}|)\epsilon.$$

Initialization: Similar to the case of column generation discussed above, the constraint generation procedure also benefits from a good initialization scheme. To this end, we use first order methods as described in Section 4—specifically, the method we use for large n (and small p), is discussed in Section 4.4.2.

Alternatively, similar to the case of column generation, we can also compute a regularization path with LP warm-start continuation.

2.3.3 Column and Constraint Generation when Both n and p Are large

When both n and p are large, we use a combination of column and constraint generation to solve L1-SVM. In particular, we update both the sets $\mathcal{I}$ and $\mathcal{J}$, as outlined below.

Algorithm 4: **Combined column and constraint generation for L1-SVM**
Input: $\mathbf{X}$, $\mathbf{y}$, a regularization coefficient λ, tolerance thresholds $\epsilon_1, \epsilon_2 \geq 0$, initial subsets $\mathcal{I}$ and $\mathcal{J}$.
Output: A near-optimal solution $(\hat{\boldsymbol{\beta}}, \hat{\beta}_0)$ for the L1-SVM Problem (2).

1. Repeat Steps 2 to 4 until $\mathcal{I}$ and $\mathcal{J}$ stabilize.
2. Solve the problem $\mathcal{P}_\lambda(\mathcal{I}, \mathcal{J})$.
3. Let $\mathcal{I}^{\epsilon_1} := \{i \in [n] \setminus \mathcal{I} \mid \bar{\pi}_i > \epsilon_1\}$. Update $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}^{\epsilon_1}$.
4. Let $\mathcal{J}^{\epsilon_2} := \{j \in [p] \setminus \mathcal{J} \mid \min\{\bar{\beta}_j^+, \bar{\beta}_j^-\} \leq -\epsilon_2\}$. Update $\mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{J}^{\epsilon_2}$; and go to Step 2.

By the updating rule of Algorithm 4, it holds $\tilde{\epsilon}_1 \leq \epsilon_1$ and $\tilde{\epsilon}_2 \leq \epsilon_2$ upon termination. Recall that $(\boldsymbol{\beta}^*, \beta_0^*)$ is an optimal solution to Problem (2) with corresponding optimal objective value z^*. Therefore, by Theorem 1, the optimality gap has the upper bound $\hat{z} - z^* \leq \epsilon_1(n - |\hat{\mathcal{I}}|) + \epsilon_2\|\boldsymbol{\beta}^*\|_1$, where $\hat{\mathcal{I}}$ is the set $\mathcal{I}$ upon termination of Algorithm 4. Section 4.4.3 discusses the use of first order optimization methods to initialize $\mathcal{I}$ and $\mathcal{J}$.

2.4 Application to the Group-SVM Problem

We now discuss application of column/constraint generation to the Group-SVM Problem (3). We first introduce some notation that we will use—we let $\mathcal{I}_g \subset [p]$ denote the indices corresponding to group g for $g \in [G]$.

Column generation: Below we present an LP formulation for Problem (3). We introduce auxiliary variables $\boldsymbol{v} = (v_g)_{g\in[G]}$ such that v_g refers to the L_∞-norm of the coefficients $\boldsymbol{\beta}_g$:

$$\begin{array}{lll} \text{(Group-SVM)} & \min\limits_{\substack{\boldsymbol{\xi}\in\mathbb{R}^n,\beta_0\in\mathbb{R},\\ \boldsymbol{\beta}^+,\,\boldsymbol{\beta}^-\in\mathbb{R}^p,\boldsymbol{v}\in\mathbb{R}^G}} & \sum\limits_{i=1}^n \xi_i + \lambda \sum\limits_{g=1}^G v_g \\ & \text{s.t.} & \xi_i + y_i\mathbf{x}_i^T\boldsymbol{\beta}^+ - y_i\mathbf{x}_i^T\boldsymbol{\beta}^- + y_i\beta_0 \geq 1 \qquad i\in[n] \\ & & v_g - \beta_j^+ - \beta_j^- \geq 0 \qquad j\in\mathcal{I}_g,\ g\in[G] \\ & & \boldsymbol{\xi}\geq 0,\ \boldsymbol{\beta}^+\geq 0,\ \boldsymbol{\beta}^-\geq 0,\ \boldsymbol{v}\geq 0. \end{array} \tag{23}$$

A dual of Problem (23) is given by:

$$\begin{array}{lll} \text{(Dual-Group-SVM)} & \max\limits_{\boldsymbol{\pi}\in\mathbb{R}^n} & \sum\limits_{i=1}^n \pi_i \\ & \text{s.t.} & \sum\limits_{j\in\mathcal{I}_g}\left|\sum\limits_{i=1}^n y_i x_{ij}\pi_i\right| \leq \lambda \qquad g\in[G] \\ & & \mathbf{y}^T\boldsymbol{\pi} = 0 \\ & & 0\leq\pi_i\leq 1 \qquad i\in[n]. \end{array} \tag{24}$$

Following the description in Section 2.1, we apply column generation on the groups. Here, the reduced cost of group g is given as:

$$\bar{\beta}_g = \lambda - \sum_{j\in\mathcal{I}_g}\left|\sum_{i=1}^n y_i x_{ij}\pi_i\right|, \tag{25}$$

and we include into the model (a subset of) groups g for which $\bar{\beta}_g$ is smaller than a small negative tolerance.

Computing a regularization path: The regularization path algorithm (Algorithm 2) presented in Section 2.3.1 can be adapted to the Group-SVM problem. First, note that:

$$\boldsymbol{\beta}^*(\lambda) = \mathbf{0}, \quad \forall\lambda\geq\lambda_{\max} = \max_{g\in[G]}\sum_{j\in\mathcal{I}_g}\sum_{i=1}^n |x_{ij}|. \tag{26}$$

For $\lambda = \lambda_{\max}$, the reduced cost of variables corresponding to group g is given by the "group" analogue of (22):

$$\bar{\beta}_g = \lambda_{\max} - \sum_{j\in\mathcal{I}_g}\left|\frac{N_-}{N_+}\sum_{i\in\mathcal{I}_+} y_i x_{ij} + \sum_{i\in\mathcal{I}_-} y_i x_{ij}\right|. \tag{27}$$

We can obtain a small set of groups maximizing the rhs of (27). We use these groups to initialize the LP solver to solve Problem (23) for the next small value of λ, using column generation.

We repeat this process for smaller values of λ using warm-start continuation.

Constraint generation and column generation: When n is large (but the number of groups is small) constraint generation can be used for the Group-SVM problem in a manner

similar to that used for the L1-SVM problem. Similarly, column and constraint generation can be applied together to obtain computational savings when both n and the number of groups are large.

Initialization: As discussed for the L1-SVM problem, we can use first order methods to obtain a low-accuracy solution for the Group-SVM problem—these are discussed in Section 4. This can be used to initialize the set of nonzero groups (for column generation), relevant constraints (for constraint generation), and both groups and constraints (for the combined column-and-constraint generation procedure).

3. Column and constraint generation for Slope-SVM

Here we discuss the Slope-SVM estimator i.e., Problem (4). For a p-dimensional regularization parameter $\boldsymbol{\lambda}$ with coordinates sorted as: $\lambda_1 \geq \ldots \geq \lambda_p \geq 0$, we let

$$\|\boldsymbol{\beta}\|_S := \sum_{j=1}^{p} \lambda_j |\beta_{(j)}| \tag{28}$$

denote the Sorted L1-norm or the *Slope-norm*—we borrow this term inspired by the "Slope estimator" (Bogdan et al., 2013) and acknowledge a slight abuse in terminology. Note that for convenience, we drop the dependence on $\boldsymbol{\lambda}$ in the notation $\|\cdot\|_S$.

We note that the epigraph of $\|\boldsymbol{\beta}\|_S$ i.e., $\{(\boldsymbol{\beta}, \eta) \mid \|\boldsymbol{\beta}\|_S \leq \eta\}$ admits an LP formulation using $O(p^2)$ many variables and $O(p^2)$ many constraints (cf Section A.1)—given the problem-sizes we seek to address, we do not pursue this route. Rather, we make use of the observation that the epigraph can be expressed with exponentially many linear inequalities (Section 3.1) when using $O(p)$-many variables. The large number of constraints make column/constraint generation methodology for the Slope penalty quite different than the L1-SVM example. However, as we will discuss, our procedure does *not* require us to explicitly enumerate the exponentially many inequalities. To our knowledge, the procedure we present here for the Slope-SVM problem is novel. Section 3.1 discusses a constraint generation method that greatly reduces the number of constraints needed to model the epigraph. Section 3.2 discusses the use of column generation to exploit sparsity in $\boldsymbol{\beta}$ when p is large. Finally, Section 3.3 combines these two features to address the Slope SVM problem. We note that even for large p and small n settings, both column and constraint generation methods are needed for the Slope penalty, making it different from the L1-penalty, where column generation alone suffices. In what follows, we concentrate on the case where n is small but p is large—if n is also large, a further layer of constraint generation might be needed to efficiently handle sparsity arising from the hinge-loss.

3.1 Constraint Generation for Slope-SVM

Reformulation of Slope-SVM: Note that Problem (4) can be expressed as:

$$\boxed{\mathcal{M}_S(\mathcal{C},[p])} \qquad \min_{\substack{\boldsymbol{\xi}\in\mathbb{R}^n,\ \beta_0,\eta\in\mathbb{R},\\ \boldsymbol{\beta}^+,\ \boldsymbol{\beta}^-\in\mathbb{R}^p}} \quad \sum_{i=1}^n \xi_i + \eta$$

$$\text{s.t.} \qquad \xi_i + y_i\mathbf{x}_i^T\boldsymbol{\beta}^+ - y_i\mathbf{x}_i^T\boldsymbol{\beta}^- + y_i\beta_0 \geq 1 \quad i\in[n] \tag{29a}$$

$$(\boldsymbol{\beta}^+,\ \boldsymbol{\beta}^-,\ \eta)\in\mathcal{C} \tag{29b}$$

$$\boldsymbol{\xi}\geq 0,\ \boldsymbol{\beta}^+\geq 0,\ \boldsymbol{\beta}^-\geq 0,$$

where, $\boldsymbol{\beta}=\boldsymbol{\beta}^+-\boldsymbol{\beta}^-$; and $\boldsymbol{\beta}^+,\ \boldsymbol{\beta}^-\in\mathbb{R}^p_+$ denote the positive and negative parts of $\boldsymbol{\beta}$, respectively. In line (29b) we express the Slope penalty in the epigraph form with $\mathcal{C}$ defined as:

$$\mathcal{C}:=\left\{(\boldsymbol{\beta}^+,\ \boldsymbol{\beta}^-,\ \eta)\ \middle|\ \eta\geq\sum_{j=1}^p\lambda_j\beta^+_{(j)}+\sum_{j=1}^p\lambda_j\beta^-_{(j)},\ \ \boldsymbol{\beta}^+,\ \boldsymbol{\beta}^-\in\mathbb{R}^p_+\right\},$$

where, we use the notation $\beta^+_{(1)}+\beta^-_{(1)}\geq\ldots\geq\beta^+_{(p)}+\beta^-_{(p)}$ and remind ourselves that $|\beta_i|=\beta_i^+ + \beta_i^-$ for all i. Below we show that (29b) can be expressed via linear inequalities involving $(\boldsymbol{\beta}^+,\boldsymbol{\beta}^-)$.

We first introduce some notation. Let $\mathcal{S}_p$ denote the set of all permutations of $\{1,\ldots,p\}$, with $|\mathcal{S}_p|=p!$. For a permutation $\phi\in\mathcal{S}_p$, we let $(\phi(1),\ldots,\phi(p))$ denote the corresponding rearrangement of $(1,\ldots,p)$. Using this notation, the Slope norm can be expressed as:

$$\|\boldsymbol{\beta}\|_S=\sum_{j=1}^p\lambda_j|\beta_{(j)}|=\max_{\phi\in\mathcal{S}_p}\sum_{j=1}^p\lambda_j|\beta_{\phi(j)}|=\max_{\psi\in\mathcal{S}_p}\sum_{j=1}^p\lambda_{\psi(j)}|\beta_j|. \tag{30}$$

As a consequence, we have the following lemma:

Lemma 5 *The Slope norm $\|\boldsymbol{\beta}\|_S$ admits the following representation*

$$\|\boldsymbol{\beta}\|_S=\max_{\boldsymbol{w}\in\mathcal{W}^{[p]}}\boldsymbol{w}^T(\boldsymbol{\beta}^+-\boldsymbol{\beta}^-)=\max_{\boldsymbol{w}\in\mathcal{W}_0^{[p]}}\boldsymbol{w}^T(\boldsymbol{\beta}^+-\boldsymbol{\beta}^-),$$

where, $\mathcal{W}_0^{[p]}:=\mathrm{Conv}\left(\mathcal{W}^{[p]}\right)$ is the convex hull of $\mathcal{W}^{[p]}$, where

$$\mathcal{W}^{[p]}:=\left\{\boldsymbol{w}\in\mathbb{R}^p\ \middle|\ \exists\psi\in\mathcal{S}_p\ \ s.t.\ \ \ w_j=\lambda_{\psi(j)},j\in[p]\right\}. \tag{31}$$

Proof Note that a linear function maximized over a bounded polyhedron reaches its maximum at one of the extreme points of the polyhedron—this leads to:

$$\max_{\boldsymbol{w}\in\mathcal{W}_0^{[p]}}\boldsymbol{w}^T(\boldsymbol{\beta}^+-\boldsymbol{\beta}^-)=\max_{\boldsymbol{w}\in\mathcal{W}^{[p]}}\boldsymbol{w}^T(\boldsymbol{\beta}^+-\boldsymbol{\beta}^-). \tag{32}$$

Using the definition of $\mathcal{W}^{[p]}$, we get that the rhs of (32) is $\max_{\psi\in\mathcal{S}_p}\sum_{j=1}^p\lambda_{\psi(j)}|\beta_j|$ which is in fact the Slope norm $\|\boldsymbol{\beta}\|_S$. ■

The following remark provides a description of $\mathcal{W}^{[p]}$ for special choices of $\boldsymbol{\lambda}$.

Remark 6 ***(a)*** *If all the coefficients are equal i.e.,* $\lambda_1 = \ldots = \lambda_p$ *and* $\|\boldsymbol{\beta}\|_S = \lambda\|\boldsymbol{\beta}\|_1$*, then* $\mathcal{W}^{[p]}$ *is a singleton.* ***(b)*** *If all the coefficients are distinct i.e.,* $\lambda_1 > \ldots > \lambda_p$*, then each permutation* $\psi \in \mathcal{S}_p$ *is associated with a unique vector in* $\mathcal{W}^{[p]}$ *and* $\mathcal{W}^{[p]}$ *contains* $p!$ *elements.*

Using Lemma 5, we can derive an LP formulation of Problem (29) by modeling $\mathcal{C}$ in (29b) as:

$$\mathcal{C} = \left\{ \left(\boldsymbol{\beta}^+,\ \boldsymbol{\beta}^-,\ \eta\right) \ \middle|\ \boldsymbol{\beta}^+, \boldsymbol{\beta}^- \in \mathbb{R}^p,\ \ \eta \geq \max_{\boldsymbol{w} \in \mathcal{W}^{[p]}} \boldsymbol{w}^T(\boldsymbol{\beta}^+ + \boldsymbol{\beta}^-) \right\}, \tag{33}$$

where, $\mathcal{W}^{[p]}$ is defined in (31). The resulting LP formulation (29) has at most n constraints from (29a). There are at most $p!$ constraints associated with (29b) by virtue of (33). We note that many constraints in (33) are redundant: for example, the maximum is attained corresponding to the inverse of permutation ϕ (denoted by ϕ^{-1}), where $|\beta_{\phi(1)}| \geq \ldots \geq |\beta_{\phi(p)}|$. This motivates the use of constraint generation techniques.

Constraint generation: We proceed by replacing $\mathcal{W}^{[p]}$ with a smaller subset and solve the resulting LP. We subsequently refine this approximation if (29b) is violated. Formally, let us consider a collection of vectors $\mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(t)} \in \mathcal{W}^{[p]}$ leading to a superset $\mathcal{C}_t$ of $\mathcal{C}$:

$$\mathcal{C} \subseteq \mathcal{C}_t := \left\{ \left(\boldsymbol{\beta}^+,\ \boldsymbol{\beta}^-,\ \eta\right) \ \middle|\ \boldsymbol{\beta}^+, \boldsymbol{\beta}^- \in \mathbb{R}^p,\ \ \eta \geq \sum_{j=1}^p w_j^{(\ell)} \beta_j^+ + \sum_{j=1}^p w_j^{(\ell)} \beta_j^-,\ \forall \ell \leq t \right\}. \tag{34}$$

By replacing $\mathcal{C}$ in (29b) by $\mathcal{C}_t$, we get an LP denoted by $\mathcal{M}_S(\mathcal{C}_t, [p])$ which is a relaxation of $\mathcal{M}_S(\mathcal{C}, [p])$. Let $(\boldsymbol{\beta}^*, \eta^*)$ be a solution of $\mathcal{M}_S(\mathcal{C}_t, [p])$. If this is not an optimal solution at a tolerance threshold $\epsilon \geq 0$, we will expand $\mathcal{C}_t$ if

$$\eta^* + \epsilon < \sum_{j=1}^p \lambda_j |\beta^*_{(j)}| = \|\boldsymbol{\beta}^*\|_S.$$

To this end, consider a permutation $\psi_{t+1} \in \mathcal{S}_p$ such that $|\beta^*_{\psi_{t+1}(1)}| \geq \ldots \geq |\beta^*_{\psi_{t+1}(p)}|$. If ψ_{t+1}^{-1} denotes the inverse of ψ_{t+1}, we obtain $\mathbf{w}^{(t+1)} \in \mathcal{W}^{[p]}$ such that:

$$w_j^{(t+1)} = \lambda_{\psi_{t+1}^{-1}(j)}\ \forall j \in [p], \tag{35}$$

and solve the resulting LP. We continue adding these additional constraints (aka cuts), till no further cuts need to be added—this leads to a (near)-optimal solution to $\mathcal{M}_S(\mathcal{C}, [p])$. We note that the first vector $\mathbf{w}^{(1)}$ can be obtained by applying (35) on an estimator obtained from the first order optimization schemes (cf Section 4). Our algorithm is summarized below for convenience.

Algorithm 5: **Constraint generation for Slope-SVM**
Input: $\mathbf{X}$, $\mathbf{y}$, a vector of Slope coefficients $\{\lambda_j\}_{j \in [p]}$, a tolerance threshold $\epsilon > 0$, a vector $\mathbf{w}^{(1)} \in \mathcal{W}^{[p]}$.
Output: A near-optimal solution $(\boldsymbol{\beta}^*, \beta_0^*)$ for the Slope-SVM Problem (4).

1. Repeat Steps 2 to 3 (for $t \geq 1$) till no further cuts need to be added.

2. Solve the model $\mathcal{M}_S(\mathcal{C}_t, [p])$ with $\mathcal{C}_t$ as in (34). Let $(\boldsymbol{\beta}^*, \eta^*)$ be the corresponding decision variables.

3. Let $\psi_{t+1} \in \mathcal{S}_p$ be such that $|\beta^*_{\psi_{t+1}(1)}| \geq \ldots \geq |\beta^*_{\psi_{t+1}(p)}|$. If condition $\eta^* + \epsilon \geq \sum_{j=1}^p \lambda_j |\beta^*_{\psi_{t+1}(j)}|$ is not satisfied, we add a new cut $\mathbf{w}^{(t+1)} \in \mathcal{W}^{[p]}$ as per (35); update $\mathcal{C}_{t+1}$ and go to Step 2.

3.2 Dual Formulation and Column Generation for Slope-SVM

When the amount of regularization is high, the Slope penalty (with $\lambda_i > 0$ for all i) may lead to many zeros in the coefficient vector $\boldsymbol{\beta}$ at an optimal solution to Problem (4)—computational savings are possible if we can leverage this sparsity when p is large. To this end, we use column generation along with the constraint generation algorithm described in Section 3.1. In particular, given a set of columns $\mathcal{J} = \{\mathcal{J}(1), \ldots, \mathcal{J}(|\mathcal{J}|)\} \subset [p]$, we consider a restricted version of Problem (4) with $\beta_j = 0, j \notin \mathcal{J}$:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} \quad \sum_{i=1}^n \left(1 - y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0)\right)_+ + \|\boldsymbol{\beta}\|_S \quad \text{s.t.} \quad \beta_j = 0, j \notin \mathcal{J}.$$

The above can be expressed as an LP similar to Problem (29) but with fewer columns

$$\boxed{\mathcal{M}_S\left(\mathcal{C}^{\mathcal{J}}, \mathcal{J}\right)} \quad \begin{array}{ll} \min\limits_{\substack{\boldsymbol{\xi} \in \mathbb{R}^n,\ \beta_0,\ \eta \in \mathbb{R}, \\ \boldsymbol{\beta}^+_{\mathcal{J}},\ \boldsymbol{\beta}^-_{\mathcal{J}} \in \mathbb{R}^{|\mathcal{J}|}}} & \sum\limits_{i=1}^n \xi_i + \eta \\ \text{s.t.} & \xi_i + \sum\limits_{j \in \mathcal{J}} y_i x_{ij} \beta_j^+ - \sum\limits_{j \in \mathcal{J}} y_i x_{ij} \beta_j^- + y_i \beta_0 \geq 1 \qquad i \in [n] \\ & (\boldsymbol{\beta}^+_{\mathcal{J}},\ \boldsymbol{\beta}^-_{\mathcal{J}},\ \eta) \in \mathcal{C}^{\mathcal{J}} \\ & \boldsymbol{\xi} \geq 0,\ \boldsymbol{\beta}^+_{\mathcal{J}} \geq 0,\ \boldsymbol{\beta}^-_{\mathcal{J}} \geq 0, \end{array} \tag{36}$$

where, $\boldsymbol{\beta}_{\mathcal{J}}$ is a sub-vector of $\boldsymbol{\beta}$ restricted to $\mathcal{J}$ and $\mathcal{C}^{\mathcal{J}}$ is the adaption of (33) restricted to $\boldsymbol{\beta}_{\mathcal{J}}$:

$$\mathcal{C}^{\mathcal{J}} := \left\{ (\boldsymbol{\beta}^+_{\mathcal{J}},\ \boldsymbol{\beta}^-_{\mathcal{J}},\ \eta) \quad \middle| \quad \boldsymbol{\beta}^+_{\mathcal{J}}, \boldsymbol{\beta}^-_{\mathcal{J}} \in \mathbb{R}^{|\mathcal{J}|},\ \eta \geq \max_{\boldsymbol{w}_{\mathcal{J}} \in \mathcal{W}^{\mathcal{J}}} \boldsymbol{w}_{\mathcal{J}}^T (\boldsymbol{\beta}^+_{\mathcal{J}} + \boldsymbol{\beta}^-_{\mathcal{J}}) \right\}, \tag{37}$$

where, $\boldsymbol{w}_{\mathcal{J}} \in \mathbb{R}^{|\mathcal{J}|}$ and $\mathcal{W}^{\mathcal{J}}$ is defined as:

$$\mathcal{W}^{\mathcal{J}} := \left\{ \boldsymbol{w}_{\mathcal{J}} \quad \middle| \quad \exists \psi \in \mathcal{S}_{|\mathcal{J}|} \ \text{ s.t. } \ w_{\mathcal{J}(j)} = \lambda_{\psi(j)},\ \forall j \leq |\mathcal{J}| \right\}.$$

Since column generation is equivalent to constraint generation on the dual problem, to determine the set of columns to add to $\mathcal{J}$ in Problem (36), we need the dual formulation of Slope-SVM.

Dual formulation for Slope-SVM: We first present a dual (Zeng and Figueiredo, 2014) of the Slope norm:

$$\max \left\{ \boldsymbol{\beta}^T \boldsymbol{z} \quad \middle| \quad \boldsymbol{\beta} \in \mathbb{R}^p,\ \|\boldsymbol{\beta}\|_S \leq 1 \right\} = \max_{k \leq p} \left\{ \left(\sum_{j=1}^k \lambda_j \right)^{-1} \sum_{j=1}^k |z_{(j)}| \right\}. \tag{38}$$

The identity (38) follows from the observation that the maximum will be attained at an extreme point of the polyhedron $\mathcal{P}_S = \{\boldsymbol{\beta} \mid \|\boldsymbol{\beta}\|_S \leq 1\} \subset \mathbb{R}^p$. We describe these extreme points. We fix $k \in [p]$, and a subset $A \subset \{1, \ldots, p\}$ of size k—the extreme points of $\mathcal{P}_S$ having support A have their nonzero coefficients to be equal, with absolute value $\left(\sum_{j=1}^k \lambda_j\right)^{-1}$. Finally, (38) follows by taking a maximum over all $k \in [p]$.

A dual of Problem (36) is given by:

$$\begin{aligned} \max_{\boldsymbol{\pi} \in \mathbb{R}^n, \mathbf{q} \in \mathbb{R}^p} \quad & \sum_{i=1}^n \pi_i \\ \text{s.t.} \quad & \max_{k=1,\ldots,|\mathcal{J}|} \left\{ \left(\sum_{j=1}^k \lambda_j \right)^{-1} \sum_{j=1}^k |q_{(j)}| \right\} \leq 1 \\ & q_j = \sum_{i=1}^n y_i x_{ij} \pi_i, \quad j \in [p] \\ & \mathbf{y}^T \boldsymbol{\pi} = 0 \\ & 0 \leq \pi_i \leq 1, \quad i \in [n]. \end{aligned} \tag{39}$$

We now discuss how additional columns can be appended to $\mathcal{J}$ in Problem (36) to perform column generation. Let $\boldsymbol{\pi}^* \in \mathbb{R}^n$ be an optimal solution of Problem (39). We compute the associated $\mathbf{q}^*$ and sort its entries such that $|q^*_{(1)}| \geq \ldots \geq |q^*_{(|\mathcal{J}|)}|$. The first constraint in (39) leads to:

$$\max_{k=1,\ldots,|\mathcal{J}|} \left\{ \sum_{j=1}^k |q^*_{(j)}| - \sum_{j=1}^k \lambda_j \right\} \leq 0. \tag{40}$$

Now, for each column $j \notin \mathcal{J}$, we compute its corresponding $q^*_{(j)}$ and insert it into the sorted sequence $|q^*_{(1)}| \geq \ldots \geq |q^*_{(|\mathcal{J}|)}|$. This insertion costs at most $O(|\mathcal{J}|)$ flops: we update $\mathcal{J} \leftarrow \mathcal{J} \cup \{j\}$ and denote the sorted entries by: $|q^*_{\overline{(1)}}| \geq \ldots \geq |q^*_{\overline{(|\mathcal{J}|+1)}}|$. We add a column $j \notin \mathcal{J}$ to the current model if:

$$\max_{k=1,\ldots,|\mathcal{J}|+1} \left\{ \sum_{j=1}^k |q^*_{\overline{(j)}}| - \sum_{j=1}^k \lambda_j \right\} > \epsilon, \tag{41}$$

and this costs $O(|\mathcal{J}| + 1)$ flops. Therefore, the total cost of sorting the vector $\mathbf{q}^*$ and scanning through all columns that are not in the current model to identify negative reduced costs, is of the order $\mathcal{O}\left(|\mathcal{J}| \log |\mathcal{J}| + 2(p - |\mathcal{J}|)|\mathcal{J}|\right)$. This approach can be computationally expensive. To this end, we propose an alternative method having a smaller cost with $\mathcal{O}\left(|\mathcal{J}|\right)$ flops. Indeed, by combining Equations (40) and (41), a column $j \notin \mathcal{J}$ will be added to the model if it satisfies:

$$|q_j| \geq \lambda_{|\mathcal{J}|+1} + \epsilon. \tag{42}$$

This shows that the cost of adding a new column for Slope-SVM is the same as that in L1-SVM. The column generation algorithm is summarized below.

Algorithm 6: **Column generation for Slope-SVM**
Input: $\mathbf{X}$, $\mathbf{y}$, a sequence of Slope coefficients $\{\lambda_j\}$, a threshold $\epsilon > 0$, an initial set of columns $\mathcal{J}$.
Output: A near-optimal solution $(\boldsymbol{\beta}^*, \beta_0^*)$ for the Slope-SVM Problem (4).

1. Repeat Steps 2 and 3 until convergence.
2. Solve the model $\mathcal{M}_S\left(\mathcal{C}^{\mathcal{J}}, \mathcal{J}\right)$ in Problem (36) with warm-start (if available).
3. Identify the columns $\mathcal{J}^\epsilon \subset \{1, \ldots, p\} \setminus \mathcal{J}$ that need to be added by using criterion (42). Update $\mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{J}^\epsilon$, and go to Step 2.

3.3 Pairing Column and Constraint Generation for Slope-SVM

We discuss how to combine the column generation (Section 3.2) and constraint generation (Section 3.1) methods outlined above to solve the Slope SVM problem.

For a set of columns $\mathcal{J}$ and constraints associated with $\boldsymbol{w}_{\mathcal{J}}^{(1)}, \ldots, \boldsymbol{w}_{\mathcal{J}}^{(t)} \in \mathcal{W}^{\mathcal{J}}$, we consider the following problem

$$\boxed{\mathcal{M}_S\left(\mathcal{C}_t^{\mathcal{J}}, \mathcal{J}\right)} \quad \begin{array}{ll} \min\limits_{\substack{\boldsymbol{\xi}\in\mathbb{R}^n,\ \beta_0\in\mathbb{R},\ \eta\in\mathbb{R} \\ \boldsymbol{\beta}_{\mathcal{J}}^+,\ \boldsymbol{\beta}_{\mathcal{J}}^-\in\mathbb{R}^{|\mathcal{J}|}}} & \sum\limits_{i=1}^n \xi_i + \eta \\ \text{s.t.} & \xi_i + \sum\limits_{j\in\mathcal{J}} y_i x_{ij}\beta_j^+ - \sum\limits_{j\in\mathcal{J}} y_i x_{ij}\beta_j^- + y_i\beta_0 \geq 1 \qquad i \in [n] \\ & (\boldsymbol{\beta}_{\mathcal{J}}^+,\ \boldsymbol{\beta}_{\mathcal{J}}^-,\ \eta) \in \mathcal{C}_t^{\mathcal{J}} \\ & \boldsymbol{\xi} \geq 0,\ \boldsymbol{\beta}_{\mathcal{J}}^+ \geq 0,\ \boldsymbol{\beta}_{\mathcal{J}}^- \geq 0, \end{array} \tag{43}$$

where, $\mathcal{C}_t^{\mathcal{J}}$ (and $\mathcal{C}^{\mathcal{J}}$) is a restriction of $\mathcal{C}_t$ (and $\mathcal{C}$, respectively) to the columns $\mathcal{J}$. Formally,

$$\mathcal{C}_t^{\mathcal{J}} := \left\{ \left(\boldsymbol{\beta}_{\mathcal{J}}^+,\ \boldsymbol{\beta}_{\mathcal{J}}^-,\ \eta\right) \;\middle|\; \boldsymbol{\beta}_{\mathcal{J}}^+,\ \boldsymbol{\beta}_{\mathcal{J}}^- \in \mathbb{R}^{|\mathcal{J}|}, \quad \eta \geq (\boldsymbol{w}_{\mathcal{J}}^{(\ell)})^T(\boldsymbol{\beta}_{\mathcal{J}}^+ + \boldsymbol{\beta}_{\mathcal{J}}^-),\ \forall \ell \leq t \right\} \supset \mathcal{C}^{\mathcal{J}}.$$

We use the method in Section 3.1 to refine $\mathcal{C}_t^{\mathcal{J}}$. We use the method of Section 3.2—see criterion (42)—to add a set of columns to $\mathcal{J}$. Let $\mathcal{J}^\epsilon$ denote these additional columns with coordinates $\mathcal{J}^\epsilon(k)$ for $k = 1, \ldots, |\mathcal{J}^\epsilon|$—we will also assume that the elements have been sorted by increasing reduced costs. For notational purposes, we will need to map[3] the existing cuts of $\mathcal{W}^{\mathcal{J}}$ onto $\mathcal{W}^{\mathcal{J}\cup\mathcal{J}^\epsilon}$. To this end, we make the following definition:

$$w_m^{(\ell)} = \lambda_{|\mathcal{J}|+k},\ \forall m \in \mathcal{J}^\epsilon,\ \forall \ell \leq t. \tag{44}$$

We summarize our algorithm below.

Algorithm 7: **Column-and-constraint generation for Slope-SVM**
Input: $\mathbf{X}$, $\mathbf{y}$, a sequence of Slope coefficients $\{\lambda_j\}$, a convergence threshold $\epsilon \geq 0$. Initialization of $\boldsymbol{\beta}^*$ and $\mathcal{J}$ (e.g., using the first order method in Section 4.3). Define $\boldsymbol{w}_{\mathcal{J}}^{(1)}$ as per (35).
Output: A near-optimal solution $\boldsymbol{\beta}^*$ for the Slope-SVM Problem (4).

3. In other words, the existing vectors $\mathbf{w}_{\mathcal{J}}^{(\ell)}$ are in $\mathbb{R}^{|\mathcal{J}|}$ and we need to extend them to $\mathbb{R}^{|\mathcal{J}|+|\mathcal{J}^\epsilon|}$. Therefore, we need to define the coordinates corresponding to the new indices $\mathcal{J}^\epsilon$.

1. Repeat Steps 2 to 4 until no cut can be added and $\mathcal{J}$ stabilizes.
2. Solve the model $\mathcal{M}_S\left(\mathcal{C}_t^{\mathcal{J}}, \mathcal{J}\right)$ in Problem (43) (with warm-starting enabled).
3. If $\eta < \sum_{j=1}^{|\mathcal{J}|} \lambda_j |\beta^*_{(j)}| - \epsilon$, add a new cut $\boldsymbol{w}_{\mathcal{J}}^{(t+1)} \in \mathcal{W}^{\mathcal{J}}$ as in Equation (35) and define $\mathcal{C}_{t+1}^{\mathcal{J}}$.
4. Identify columns $\mathcal{J}^\epsilon \subset \{1, \ldots, p\} \setminus \mathcal{J}$ that need to be added (based on criterion (42)). Map the cuts $\boldsymbol{w}_{\mathcal{J}}^{(1)}, \ldots \boldsymbol{w}_{\mathcal{J}}^{(t+1)}$ to $\mathcal{W}^{\mathcal{J}\cup\mathcal{J}^\epsilon}$ via (44). Update $\mathcal{J} \leftarrow \mathcal{J} \cup \mathcal{J}^\epsilon$ and go to Step 2.

4. First order methods to obtain a low-accuracy solution

The computational performance of the column/constraint generation methods described above for Problems (2), (3) and (4), is found to benefit from a good initialization. For example, a good estimate of the initial set of columns can improve the overall efficiency of a column generation algorithm for Problem (2). We refer the reader to Desrosiers and Lübbecke (2005) for additional discussions on the importance of having a good initialization for column generation. Finding a good and computationally inexpensive initialization for general problems can be challenging. In our case, we propose to leverage low-accuracy solutions[4] available from first order methods (Nesterov, 2004).

Problems (2), (3) and (4) are nonsmooth: We use Nesterov's smoothing technique (Nesterov, 2005) to smooth the nonsmooth hinge-loss function and use proximal gradient descent on the composite version (Nesterov, 2013) of the problem[5]. These solutions serve as reasonable initial estimates for the sets of columns (constraints) necessary for the column (respectively, constraint) generation methods. When the number of samples and/or features become larger, a direct application of the first order methods becomes expensive and we use additional heuristics for scalability as discussed in Section 4.4.

4.1 Solving the Composite Form with Nesterov's Smoothing

Note that for a scalar u, we have $\max\{0, u\} = \frac{1}{2}(u + |u|) = \max_{|w|\le 1} \frac{1}{2}(u + wu)$ and this maximum is achieved when $w = \text{sign}(u)$. Hence, the hinge-loss can be expressed as:

$$\sum_{i=1}^{n} (z_i)_+ = \max_{\|\boldsymbol{w}\|_\infty \le 1} \sum_{i=1}^{n} \frac{1}{2} \left[z_i + w_i z_i\right], \tag{45}$$

where $z_i = 1 - y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0)$. One can obtain a smooth approximation of (45) as follows:

$$H^\tau(\mathbf{z}) := \max_{\|\mathbf{w}\|_\infty \le 1} \sum\nolimits_{i\in[n]} \frac{1}{2} \left[z_i + w_i z_i\right] - \frac{\tau}{2}\|\boldsymbol{w}\|_2^2, \tag{46}$$

where, $\tau > 0$ is a parameter that controls the amount of smoothness in $H^\tau(\mathbf{z})$ and how well it approximates $H^0(\mathbf{z}) = \sum_{i=1}^n (z_i)_+$. This is formalized in the following lemma adapted from Nesterov (2005):

4. As our experiments demonstrate, obtaining high accuracy solutions via first order methods can become prohibitively expensive especially, when compared to the column/constraint algorithms presented here.
5. For the Group-SVM problem, we use proximal block coordinate methods instead of proximal gradient methods as they lead to better numerical performance.

Lemma 7 *The function* $\mathbf{z} \mapsto H^\tau(\mathbf{z})$ *is an* $O(\tau)$*-approximation for the hinge loss* $H^0(\mathbf{z})$ *i.e.,* $H^0(\mathbf{z}) \in [H^\tau(\mathbf{z}), H^\tau(\mathbf{z}) + n\tau/2]$ *for all* $\mathbf{z}$*. Furthermore,* $H^\tau(\mathbf{z})$ *has Lipschitz continuous gradient with parameter* $1/(4\tau)$*, i.e.,* $\|\nabla H^\tau(\mathbf{z}) - \nabla H^\tau(\mathbf{z}')\|_2 \leq 1/(4\tau)\|\mathbf{z} - \mathbf{z}'\|_2$ *for all* $\mathbf{z}, \mathbf{z}'$*.*

Let us define:

$$F^\tau(\boldsymbol{\beta}, \beta_0) = \max_{\|\mathbf{w}\|_\infty \leq 1} \left\{ \sum_{i=1}^n \frac{1}{2} \left[1 - y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) + w_i(1 - y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0)) \right] - \frac{\tau}{2} \|\boldsymbol{w}\|_2^2 \right\}. \quad (47)$$

By Lemma 7, it follows that $F^\tau(\boldsymbol{\beta}, \beta_0)$ is a uniform $O(\tau)$-approximation to the hinge-loss function. The gradient of F^τ is given by:

$$\nabla F^\tau(\boldsymbol{\beta},\ \beta_0) = -\frac{1}{2} \sum_{i=1}^n (1 + w_i^\tau) y_i \tilde{\mathbf{x}}_i \in \mathbb{R}^{p+1}, \quad (48)$$

where $\boldsymbol{w}^\tau$ is the optimal solution in (47). In addition, $(\boldsymbol{\beta}, \beta_0) \mapsto \nabla F^\tau(\boldsymbol{\beta},\ \beta_0)$ is Lipschitz continuous with parameter $C^\tau = \sigma_{\max}(\tilde{\mathbf{X}}^T \tilde{\mathbf{X}})/(4\tau)$, where $\tilde{\mathbf{X}}_{n \times (p+1)}$ is a matrix with ith row $(\mathbf{x}_i, 1)$.

We use a proximal gradient method (Beck and Teboulle, 2009) to the following composite form of the smoothed-hinge-loss SVM problem with regularizer $\Omega(\boldsymbol{\beta})$

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} F^\tau(\boldsymbol{\beta}, \beta_0) + \Omega(\boldsymbol{\beta}), \quad (49)$$

where, $\Omega(\boldsymbol{\beta}) = \lambda \|\boldsymbol{\beta}\|_1$ for L1-SVM, $\Omega(\boldsymbol{\beta}) = \lambda \sum_{g=1}^G \|\boldsymbol{\beta}_g\|_\infty$ for Group-SVM and $\Omega(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_S$ for Slope-SVM. For these choices, the proximal/thresholding operators can be computed efficiently, as we discuss next.

4.2 Thresholding Operators

For notational convenience we set $\boldsymbol{\gamma} = (\boldsymbol{\beta}, \beta^0) \in \mathbb{R}^{p+1}$. Following Nesterov (2004); Beck and Teboulle (2009), for $L \geq C^\tau$ we have that $\boldsymbol{\gamma} \mapsto Q_L(\boldsymbol{\gamma}; \boldsymbol{\alpha})$ is an upper bound to $\boldsymbol{\gamma} \mapsto F^\tau(\boldsymbol{\gamma})$, i.e, for all $\boldsymbol{\alpha}, \boldsymbol{\gamma} \in \mathbb{R}^{p+1}$:

$$F^\tau(\boldsymbol{\gamma}) \leq Q_L(\boldsymbol{\gamma}; \boldsymbol{\alpha}) := F^\tau(\boldsymbol{\alpha}) + \nabla F^\tau(\boldsymbol{\alpha})^T (\boldsymbol{\gamma} - \boldsymbol{\alpha}) + \frac{L}{2} \|\boldsymbol{\gamma} - \boldsymbol{\alpha}\|_2^2. \quad (50)$$

The proximal gradient method requires solving the following problem:

$$\hat{\boldsymbol{\gamma}} = \arg\min_{\boldsymbol{\gamma}} \{Q_L(\boldsymbol{\gamma}; \boldsymbol{\alpha}) + \Omega(\boldsymbol{\gamma})\} = \arg\min_{\boldsymbol{\gamma}} \frac{1}{2} \left\| \boldsymbol{\gamma} - \left(\boldsymbol{\alpha} - \frac{1}{L} \nabla F^\tau(\boldsymbol{\alpha}) \right) \right\|_2^2 + \frac{1}{L} \Omega(\boldsymbol{\gamma}). \quad (51)$$

We denote: $\hat{\boldsymbol{\gamma}} = (\hat{\boldsymbol{\beta}}, \hat{\beta}^0)$. Note that $\hat{\beta}_0$ is simple to compute and $\hat{\boldsymbol{\beta}}$ can be computed via the following thresholding operator:

$$\mathcal{S}_{\mu\Omega}(\boldsymbol{\eta}) := \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{2} \|\boldsymbol{\beta} - \boldsymbol{\eta}\|_2^2 + \mu \Omega(\boldsymbol{\beta}) \quad (52)$$

for some $\mu > 0$. Computation of the thresholding operator is discussed below for specific choices of Ω.

Thresholding operator when $\Omega(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_1$: In this case, $\mathcal{S}_{\mu\Omega}(\boldsymbol{\eta})$ is available via componentwise soft-thresholding where, the scalar soft-thresholding operator is given by:

$$\arg\min_{u\in\mathbb{R}} \ \frac{1}{2}(u-c)^2 + \mu|u| = \text{sign}(c)(|c|-\mu)_+.$$

Thresholding operator when $\Omega(\boldsymbol{\beta}) = \sum_{g\in[G]} \|\boldsymbol{\beta}_g\|_\infty$: We first consider the projection operator that projects onto an L1-ball with radius $\mu > 0$

$$\tilde{\mathcal{S}}_{\frac{1}{\mu}\|\cdot\|_1}(\boldsymbol{\eta}) := \arg\min_{\boldsymbol{\beta}} \ \frac{1}{2}\|\boldsymbol{\beta}-\boldsymbol{\eta}\|_2^2 \quad \text{s.t.} \quad \frac{1}{\mu}\|\boldsymbol{\beta}\|_1 \leq 1. \tag{53}$$

From standard results pertaining to the Moreau decomposition (Moreau, 1962; Bach et al., 2011) we have:

$$\mathcal{S}_{\mu\|.\|_\infty}(\boldsymbol{\eta}) + \tilde{\mathcal{S}}_{\frac{1}{\mu}\|\cdot\|_1}(\boldsymbol{\eta}) = \boldsymbol{\eta} \tag{54}$$

for any $\boldsymbol{\eta}$. Note that $\tilde{\mathcal{S}}_{\frac{1}{\mu}\|\cdot\|_1}(\boldsymbol{\eta})$ can be computed via a simple sorting operation (Van Den Berg and Friedlander, 2007, 2008; Condat, 2016), leading to a solution for $\mathcal{S}_{\mu\|.\|_\infty}(\boldsymbol{\eta})$. This observation can be used to solve Problem (52) with the L1/L_∞ Group regularizer by noticing that the problem separates across the G groups.

Thresholding operator when $\Omega(\boldsymbol{\beta}) = \sum_{i\in[p]} \lambda_i|\beta_{(i)}|$: For the Slope regularizer, Problem (52) reduces to the following optimization problem:

$$\min_{\boldsymbol{\beta}} \ \frac{1}{2}\|\boldsymbol{\beta}-\boldsymbol{\eta}\|_2^2 + \mu\sum_i \lambda_i|\beta_{(i)}|. \tag{55}$$

As noted by Bogdan et al. (2015), at an optimal solution to Problem (55), the signs of β_j and η_j are the same. In addition, since λ_i's are decreasing, a solution to Problem (55) can be found by solving the following close relative to the isotonic regression problem (Robertson, 1988)

$$\min_{\mathbf{u}} \ \frac{1}{2}\|\mathbf{u}-\tilde{\boldsymbol{\eta}}\|_2^2 + \sum_{j=1}^{p}\mu\lambda_j u_j \qquad \text{s.t.} \quad u_1 \geq \ldots \geq u_p \geq 0, \tag{56}$$

where, $\tilde{\boldsymbol{\eta}}$ is a decreasing re-arrangement of the absolute values of $\boldsymbol{\eta}$, with $\tilde{\eta}_i \geq \tilde{\eta}_{i+1}$ for all i. If $\hat{\mathbf{u}}$ is a solution to Problem (56)—then its ith coordinate $\hat{u}_i$ corresponds to $|\hat{\beta}_{(i)}|$ where, $\hat{\boldsymbol{\beta}}$ is an optimal solution of Problem (55).

4.3 Deterministic First Order Algorithms

(Accelerated) Proximal gradient descent: Let us denote the mapping (51) $\boldsymbol{\alpha} \mapsto \hat{\boldsymbol{\gamma}}$ by the operator: $\hat{\boldsymbol{\gamma}} := \Theta(\boldsymbol{\alpha})$. The basic version of the proximal gradient descent algorithm performs the updates: $\boldsymbol{\alpha}_{T+1} = \Theta(\boldsymbol{\alpha}_T)$, for $T \geq 1$, after starting with $\boldsymbol{\alpha}_1 = (\boldsymbol{\beta}_1, \beta_1^0)$. The accelerated gradient descent algorithm (Beck and Teboulle, 2009), which enjoys a faster convergence rate performs updates with a minor modification. It starts with $\boldsymbol{\alpha}_1 = \tilde{\boldsymbol{\alpha}}_0$, $q_1 = 1$ and then performs the updates: $\tilde{\boldsymbol{\alpha}}_{T+1} = \Theta(\boldsymbol{\alpha}_T)$ where, $\boldsymbol{\alpha}_{T+1} = \tilde{\boldsymbol{\alpha}}_T + \frac{q_T-1}{q_{T+1}}(\tilde{\boldsymbol{\alpha}}_T - \tilde{\boldsymbol{\alpha}}_{T-1})$

and $q_{T+1} = (1 + \sqrt{1 + 4q_T^2})/2$. This algorithm requires $O(1/\epsilon)$ iterations to reach an ϵ-optimal solution for the original problem with the hinge-loss. We perform these updates till some tolerance criterion is satisfied, for example, $\|\boldsymbol{\alpha}_{T+1} - \boldsymbol{\alpha}_T\| \leq \eta$ for some tolerance level $\eta > 0$. In most of our examples (cf Section 5), we set a generous tolerance of $\eta = 10^{-3}$ or run the algorithm with a limit on the total number of iterations, usually a few hundred in our experiments[6].

Block Coordinate Descent (CD) for the Group-SVM problem: We describe a cyclical proximal block coordinate (CD) descent algorithm (Wright, 2015) for the smooth hinge-loss function with the group regularizer. For the group-SVM experiments considered in Section 5, the block CD approach was found to exhibit superior numerical performance compared to a full gradient-based procedure. We note that Qin et al. (2013) explore block CD like algorithms for a family of group Lasso type problems[7] but our approaches differ.

We perform a proximal gradient step on the gth group of coefficients with all other blocks and β_0 held fixed. This is given by the following update:

$$\boldsymbol{\beta}_g^{t+1} \in \underset{\boldsymbol{\beta}_g}{\arg\min}\, \frac{1}{2}\left\|\boldsymbol{\beta}_g - \boldsymbol{\beta}_g^t - \frac{1}{C_g^\tau}\left\{\nabla F^\tau(\boldsymbol{\beta}_1^{t+1}, \ldots \boldsymbol{\beta}_{g-1}^{t+1}, \boldsymbol{\beta}_g^t, \ldots \boldsymbol{\beta}_G^t, \beta_0^t)\right\}_{\mathcal{I}_g}\right\|_2^2 + \frac{\lambda}{C_g^\tau}\|\boldsymbol{\beta}_g\|_\infty, \tag{57}$$

where $\{\nabla F^\tau(\cdot)\}_{\mathcal{I}_g}$ denotes the gradient restricted to the coordinates $\mathcal{I}_g$ and C_g^τ is its associated Lipschitz constant: $C_g^\tau = \sigma_{\max}(\mathbf{X}_{\mathcal{I}_g}^T \mathbf{X}_{\mathcal{I}_g})/4\tau$. We cyclically update the coefficients across each group $g \in [G]$ and then update β^0. This continues till some convergence criterion is met.

Computational savings are possible for this block CD algorithm by a careful accounting of floating point operations (flops). As one moves from one group to the next, the whole gradient can be updated easily. To this end, note that the gradient $\nabla F^\tau(\boldsymbol{\beta}, \beta_0)$ restricted to block g is given by:

$$\{\nabla F^\tau(\boldsymbol{\beta}, \beta_0)\}_{\mathcal{I}_g} = -\frac{1}{2}\mathbf{X}_{\mathcal{I}_g}^T \left\{\mathbf{y} \circ (1 + \mathbf{w}^\tau)\right\},$$

where '$\circ$' denotes element-wise multiplication. If $\mathbf{w}^\tau$ is known, the above computation requires $n|\mathcal{I}_g|$ flops. Recall that $\mathbf{w}^\tau$ depends upon $\boldsymbol{\beta}$ via: $w_i^\tau = \min\left(1, \frac{1}{2\tau}|z_i|\right)\mathrm{sign}(z_i)$ where $z_i = 1 - y_i(\mathbf{x}_i^T\boldsymbol{\beta} + \beta_0)$, $\forall i$. If $\boldsymbol{\beta}$ changes from $\boldsymbol{\beta}^{\text{old}}$ to $\boldsymbol{\beta}^{\text{new}}$, then $\mathbf{w}^\tau$ changes via an update in $\mathbf{X}\boldsymbol{\beta}$—this change can be efficiently computed by noting that: $\mathbf{X}\boldsymbol{\beta}^{\text{new}} = \sum_{g\in[G]} \mathbf{X}_{\mathcal{I}_g}\boldsymbol{\beta}_g^{\text{new}} = \mathbf{X}\boldsymbol{\beta}^{\text{old}} + \mathbf{X}_{\mathcal{I}_g}\Delta\boldsymbol{\beta}_g$ where, $\Delta\boldsymbol{\beta}_g = \boldsymbol{\beta}_g^{\text{new}} - \boldsymbol{\beta}_g^{\text{old}}$ is a change that is only restricted to block g. Hence updating $\mathbf{w}^\tau$ also requires $n|\mathcal{I}_g|$ operations. The above suggests that one sweep of block CD across all the coordinates has a cost similar to that of computing a full gradient. In addition, techniques like active set updates and warm-start continuation (Friedman et al., 2010b) can lead to improved computational performance for CD, in practice.

6. This choice is user-dependent—there is a tradeoff between computation time and the quality of solution. We recommend using a low accuracy solution as its purpose is to serve as an initialization for the column/constraint generation method
7. Qin et al. (2013) consider a different class of problems than those studied here. They consider a squared error loss function and use exact minimization for every block.

4.4 Scalability Heuristics for Large Problem instances

When n and/or p becomes large, the first order algorithms discussed above become expensive. Recall that the goal of the first order methods is to get a low-accuracy solution for the SVM problem and in particular, an estimate of the initial columns and/or constraints for the column/constraint generation algorithms. Hence, for scalability purposes, we use principled heuristics as a wrapper around the first order methods, discussed above.

4.4.1 Correlation Screening when p is Large and n is Small

When $p \gg n$, we use a feature screening method inspired by correlation screening (Tibshirani et al., 2012), to restrict the number of features. We apply the first order methods on this reduced set of features. Usually, for L1-SVM and Slope-SVM, we select for example, the top $10n$ columns with highest absolute inner product[8] with the output. For the Group-SVM problem: for each group, we compute the inner products between every feature within this group and the response, and take their L1-norm. We then sort these numbers and take the top n groups.

4.4.2 A Subsampling Heuristic when n is Large and p is Small

The methods described in Section 4.3 become expensive due to gradient computations when n becomes large. When n is large but p is small, we use a subsampling method inspired by Lee et al. (2017). To get an approximate solution to Problem (2) we apply the algorithm in Section 4.3 on a subsample $(y_i, \mathbf{x}_i), i \in \mathcal{A}$ with sample-indices $\mathcal{A} \subset [n]$. To adjust the dependence of λ on the sample size, we set $\lambda \leftarrow \frac{|\mathcal{A}|}{n}\lambda$ and approximately solve Problem (2) by using the algorithms in Section 4.3. Let the solution obtained be given by $\hat{\boldsymbol{\beta}}(\mathcal{A})$. We obtain $\hat{\boldsymbol{\beta}}(\mathcal{A}_j)$ for different subsamples $\mathcal{A}_j$, $j \in [Q]$ and average the estimators[9] to get: $\bar{\boldsymbol{\beta}}_Q = \frac{1}{Q}\sum_{j\in[Q]} \hat{\boldsymbol{\beta}}(\mathcal{A}_j)$. We maintain a counter for Q, and stop as soon as the average stabilizes[10], i.e., $\|\bar{\boldsymbol{\beta}}_Q - \bar{\boldsymbol{\beta}}_{Q-1}\| \leq \mu_{\text{Tol}}$ for some tolerance threshold μ_{Tol}. The estimate $\bar{\boldsymbol{\beta}}_Q$ is used to obtain the violated constraints for the SVM problem and serves to initialize the constraint generation method.

4.4.3 A Subsampling Heuristic when Both n and p are Large

In problems where both n and p are large, we use a combination of the ideas described above in Sections 4.4.1 and 4.4.2. More specifically, we choose a subsample $\mathcal{A}_j$ and for this subsample, we use correlation screening to reduce the number of features and obtain an estimator $\hat{\boldsymbol{\beta}}(\mathcal{A}_j)$. We then average these estimators across $\mathcal{A}_j$s, to obtain $\bar{\boldsymbol{\beta}}_Q$. If the support of $\bar{\boldsymbol{\beta}}_Q$ is too large, we sort the absolute values of the coefficients and retain the top few hundred coefficients in absolute value to initialize the column generation method. The estimator $\bar{\boldsymbol{\beta}}_Q$ is used to identify the samples for which the hinge-loss is nonzero—these indices are used to initialize the constraint generation method.

8. Note that the features are standardized to have unit L2-norm
9. We note that the estimates $\hat{\boldsymbol{\beta}}(\mathcal{A}_j)$ can all be computed in parallel.
10. We note that when n is large (but p is small), the estimator $\hat{\boldsymbol{\beta}}(\mathcal{A}_j)$ will serve as a proxy of a minimizer of Problem (2) (van de Geer, 2000)—we average the estimators $\hat{\boldsymbol{\beta}}(\mathcal{A}_j)$'s to reduce variance.

5. Experiments

We demonstrate the performance of our different methods on synthetic and real data sets. All computations are performed in Python 3.6 on the MIT Engaging Cluster with 1 CPU and 16GB of RAM. We use Gurobi 9.0.2 (Gurobi Optimization, 2021) in our experiments involving Gurobi's LP solver. Sections 5.1, 5.2 and 5.3 present results for the L1-SVM, Group-SVM and Slope-SVM problems, respectively.

5.1 Computational Results for L1-SVM

We present herein our computational experience with regard to the L1-SVM problem.

Data Generation: We consider n samples from a multivariate Gaussian distribution with covariance matrix $\mathbf{\Sigma} = ((\sigma_{ij}))$ with $\sigma_{ij} = \rho$ if $i \neq j$ and $\sigma_{ij} = 1$ otherwise. Half of the samples are from the $+1$ class and have mean $\boldsymbol{\mu}_+ = (\mathbf{1}_{k_0},\ \mathbf{0}_{p-k_0})$. The other half are from the -1 class and have mean $\boldsymbol{\mu}_- = -\boldsymbol{\mu}_+$. We standardize the columns of $\mathbf{X}$ to have unit L2-norm. In the following results with synthetic data sets, unless otherwise mentioned, we take $\rho = 0.1$, and $k_0 = 10$.

Accuracy Metric: For an algorithm 'Alg', a regularization parameter λ, we let $f^{\text{Alg}}_{\lambda,i}$ be the objective value obtained by 'Alg' for the unconstrained problem (2), in the i-th replication of the experiment. We let $f^*_{\lambda,i}$ be an estimate of the optimal objective value, set to be the best objective value among all methods. The Averaged Relative Error (ARE) in terms of objective value of a method 'Alg' is given by:

$$\text{ARE} = \frac{1}{R}\sum_{i=1}^{R}(f^{\text{Alg}}_{\lambda,i} - f^*_{\lambda,i})/f^*_{\lambda,i},$$

where R is the total number of replications. Note that ARE depends on λ, 'Alg' and R; but we drop the dependence for notational convenience.

We note that there are various other measures that can be used to assess the performance of an algorithm: candidates include test misclassification error, false positives/negatives, sparsity of the solution, etc. As the focus of our work is on optimization performance i.e, how well an algorithm performs in optimizing the LP (2), we consider the metric 'ARE' to compare the performance of different methods.

5.1.1 Synthetic Data sets for Large p and Small n

Different initializations for column generation: We first study the role of different initialization schemes in column generation (or, CLG in short) for solving the L1-SVM problem. We consider a *fixed* value of the regularization parameter, set to $\lambda = 0.01\lambda_{\max}$; and compare the following three schemes:

(**a**) `RP-CLG`: We compute a solution to Problem (2) at the desired value of λ, using a regularization path (RP) or continuation approach. We compute solutions on a grid of 7 regularization parameter values in the range $[\frac{1}{2}\lambda_{\max}, \lambda]$ using column generation (CLG) for every value of the regularization parameter.

(**b**) `FO-CLG`: This is the column generation method initialized with a first order (FO) method (cf Section 4.3) with smoothing parameter $\tau = 0.2$. We use a termination

criterion of $\eta = 10^{-3}$ or a maximum number of $T_{\max} = 200$ iterations for the first order method. We use correlation screening to retain the top $10n$ features before applying the first order method. The time displayed includes the time taken to run the first order method. For reference, we report the time taken to run column generation excluding the time of the first order method: "`CLG wo FO`".

(**c**) `Cor. screening`: This initializes the column generation method by using correlation screening to retain the top 50 features.

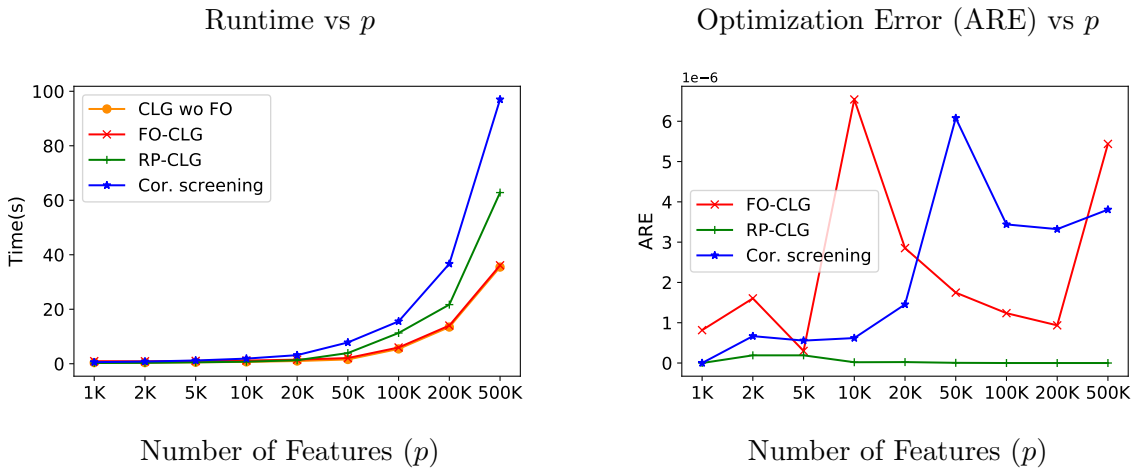

Figure 1: Comparison of different initialization schemes for the column generation method for the L1-SVM LP. Left panel shows runtime (s) versus p. Right panel shows the corresponding optimization error i.e., ARE, versus p. Here we consider $n = 100$ and vary p in the range $1,000$ to $500,000$.

The comparative timings between `FO-CLG` and `Cor. screening` show the effectiveness of using a first order method to initialize the column generation method. Method (a) computes a regularization path using column generation to arrive at a solution for the desired value of λ—it does not use any first order method like Method (b). Thus any timing difference between methods (a) and (b) can be attributed to the first order methods for warm-starting.

Figure 1 shows the results for synthetic data sets with $n = 100$, $k_0 = 10$, $\rho = 0.1$ and different values of p in the range 1000 to $500,000$. The results are averaged across $R = 10$ replications. In this figure, for the column generation methods, we consider a reduced cost threshold of $\epsilon = 0.001$, and set the maximum number of columns to be added in each iteration as 1000. The left panel in Figure 1 presents the run times and the right figure presents the ARE of different methods. As p increases, the run time for column generation when initialized with correlation screening, increases. Column generation is found to benefit the most when initialized with the first order method—recall that this algorithm is denoted by `FO-CLG`. The runtime of the first order method is negligible compared to the time taken by column generation, as seen from the nearly overlapping profiles of `FO-CLG` and `CLG wo FO`. The accuracies of the different procedures (a)—(c) are all quite high with ARE$\sim 10^{-6}$.

Comparison with benchmarks: We compare the performance of two column-generation methods `RP-CLG` and `FO-CLG` with the following benchmarks:

(d) `PSM`: This is a state-of-the-art algorithm (Pang et al., 2017) which is a parametric simplex based solver. We use the software made available by Pang et al. (2017) with default parameter-settings.

(e) `FOM`: This is our first order method based on accelerated gradient descent, denoted by FOM. We use $\tau = 0.02$ for a maximum of $T = 100,000$ iterations. We terminate the algorithm if the maximum iteration limit is reached or the L2-norm of the difference in $\boldsymbol{\beta}$ across the past two iterations is less than 10^{-3}.

(f) `SGD`: This runs a stochastic sub-gradient algorithm on the L1-SVM Problem (2). We use Python's `scikit-learn` package implementation `SGDClassifier` with fixed number of $10,000$ iterations. The learning rate is set to the "optimal" parameter.

(g) `SCS`: This is the Splitting Conic Solver (O'Donoghue et al., 2019) which is a variant of the ADMM method (Boyd et al., 2011). We use version 2.1.2 with default parameter settings. The solver is called through `CVXPY` (Diamond and Boyd, 2016) interface.

(h) `Gurobi`: This is the LP solver of Gurobi (Gurobi Optimization, 2021) with default setting for solving the full L1-SVM LP. The solver is called through `CVXPY`.

Note that all the benchmarks above optimize for the L1-SVM objective function. The only exception is `FOM` which considers a smooth approximation of the hinge-loss.

Table 1 presents the results for different methods: the top panel shows the case with $\lambda = 0.05\lambda_{\max}$, and the bottom panel shows results with $\lambda = 0.2\lambda_{\max}$. Here we run `RP-CLG` and `FO-CLG` with reduced cost tolerance $\epsilon = 0.01$. For each combination $(n, p) = (100, 10K)$, $(n, p) = (300, 10K)$ and $(n, p) = (100, 50K)$, and for each method considered, we show the runtime and associated ARE. Here, results are averaged over 5 replications, and numbers within parenthesis denote standard errors. For the instance in Table 1 with $n = 100$ and $p = 50K$, we run FOM for 2000 iterations—in this instance, using a tolerance threshold of 10^{-3} does not lead to a solution with high optimization accuracy.

In addition, to give an idea of the sparsity level of the solution across the chosen λ-values, we present the support size of an optimal solution $\hat{\boldsymbol{\beta}}$, as computed by `Gurobi`. We also report the number of columns $|\mathcal{J}|$ in the restricted problem, upon termination of `FO-CLG`. From Table 1, it can be seen that our proposed methods: `RP-CLG` and `FO-CLG` outperform all the benchmark methods in runtime by a factor of $30X \sim 500X$. In terms of solution accuracy, our column generation methods reach an ARE $\sim 10^{-5}$ or smaller, and appear to be comparable to that of `Gurobi`, `PSM`. The operator splitting method `SCS` leads to solutions of low-accuracy: the ARE $\sim 10^{-3}$ for $(n, p) = (100, 10K)$ and $(300, 10K)$, and the ARE is slightly larger $\sim 10^{-1}$—10^{-2} for $(n, p) = (100, 50K)$. `SGD` and `FOM` also lead to low-accuracy solutions; with `FOM` leading to somewhat better performance compared to `SCS` and `SGD`. We note that the poor performance of `SGD` in Table 2 should not come as a surprise, as stochastic subgradient methods are perhaps not designed for small n and large p settings. In addition, given our earlier discussion, deterministic subgradient methods for nonsmooth

$\lambda = 0.05\lambda_{\max}$

	$n = 100, p = 10K$		$n = 300, p = 10K$		$n = 100, p = 50K$	
Method	Time (s)	ARE	Time (s)	ARE	Time (s)	ARE
PSM	15.3(1.23)	6.9e-12(1.43e-12)	193.8(24.94)	1.8e-11(5.46e-12)	178.1(10.05)	2.0e-11(8.45e-12)
SGD	34.4(0.36)	9.5e-02(9.98e-03)	108.8(0.45)	6.1e-02(5.22e-03)	172.0(0.87)	1.6e-01(3.01e-02)
SCS	76.3(3.68)	1.4e-03(4.47e-04)	186.9(2.88)	1.2e-03(1.02e-04)	456.5(7.19)	7.3e-01(6.40e-01)
FOM	12.4(0.90)	2.0e-02(4.31e-04)	21.4(1.20)	9.1e-03(2.53e-04)	92.0(4.65)	3.0e-02(3.77e-04)
Gurobi	34.5(1.68)	0.0e+00(0.00e+00)	93.1(8.66)	0.0e+00(0.00e+00)	250.0(8.89)	0.0e+00(0.00e+00)
RP-CLG	0.4(0.03)	5.5e-06(3.81e-06)	2.0(0.14)	1.0e-06(6.43e-07)	3.3(0.56)	8.5e-06(5.79e-06)
FO-CLG	0.6(0.07)	6.7e-06(3.21e-06)	2.0(0.14)	7.6e-06(4.84e-06)	1.4(0.12)	2.0e-05(8.16e-06)
$\|\hat{\boldsymbol{\beta}}\|_0$ & $\|\mathcal{J}\|$	55.0 & 265.8		92.8 & 218.2		65.6 & 257.8	

$\lambda = 0.2\lambda_{\max}$

	$n = 100, p = 10K$		$n = 300, p = 10K$		$n = 100, p = 50K$	
Method	Time (s)	ARE	Time (s)	ARE	Time (s)	ARE
PSM	9.5(0.61)	3.9e-13(1.33e-13)	87.9(13.15)	8.9e-13(3.69e-13)	153.0(10.03)	2.9e-12(1.41e-12)
SGD	51.9(0.47)	1.1e-02(1.89e-03)	162.6(0.44)	5.3e-03(6.88e-04)	260.4(1.64)	1.1e-02(2.29e-03)
SCS	84.4(2.63)	3.0e-03(7.26e-04)	201.6(22.27)	2.3e-03(9.70e-04)	420.7(10.80)	4.2e-02(6.05e-03)
FOM	8.2(0.35)	3.7e-03(3.01e-04)	15.5(0.79)	1.0e-03(7.59e-05)	125.6(0.13)	6.4e-03(5.38e-04)
Gurobi	26.2(0.89)	0.0e+00(0.00e+00)	53.3(8.16)	0.0e+00(0.00e+00)	264.9(15.96)	0.0e+00(0.00e+00)
RP-CLG	0.5(0.10)	1.4e-07(9.93e-08)	0.4(0.08)	1.8e-08(1.65e-08)	4.8(0.20)	9.1e-07(8.18e-07)
FO-CLG	0.3(0.00)	2.2e-08(1.92e-08)	0.9(0.03)	0.0e+00(0.00e+00)	0.9(0.04)	6.5e-07(4.57e-07)
$\|\hat{\boldsymbol{\beta}}\|_0$ & $\|\mathcal{J}\|$	36.8 & 73.6		30.8 & 42.8		53.0 & 144.8	

Table 1: *L1-SVM, Synthetic data set ($p \gg n$)*: Training time (s) for L1-SVM: our proposed column generation method versus various benchmarks on synthetic data sets. We show two different values of λ: we select $\lambda = 0.05\lambda_{\max}$ for the top table, and $\lambda = 0.2\lambda_{\max}$ for the bottom table. Each table presents results for different values of (n, p) with $p \gg n$. For each table, the last row presents the number of nonzeros in $\hat{\boldsymbol{\beta}}$, which is an optimal solution to the L1-SVM problem. We also present the number of columns, $|\mathcal{J}|$, in the restricted problem, upon termination of FO-CLG. Results are averaged over 5 replications; and the numbers with parenthesis denote standard errors. For a cleaner display, all positive values below 10^{-15} are displayed as zero.

problems have a slower convergence compared to Nesterov's smoothing technique—the FOM presented here is an instance of Nesterov's smoothing technique (see Section 4).

Performance of CLG for difference tolerance levels ϵ: Table 1 above presents the results for FO-CLG for $\epsilon = 0.01$. In Figure 2 we study sensitivity to the choice of ϵ—we present the runtime and ARE of FO-CLG under different tolerance values $\epsilon \in \{0.01, 0.03, 0.1, 0.3, 1\}$ for $\lambda = 0.05\lambda_{\max}$. It can be seen that the ARE of FO-CLG changes with ϵ, and a value of $\epsilon = 0.01$ generally leads to a solution of high-accuracy. As ϵ decreases, the runtime slightly increases.

Computing a path of solutions: The results above discuss obtaining a solution to the L1-SVM problem for a fixed value of λ. Here we present results for computing L1-SVM solutions for a grid of λ values—we focus on comparing the performances of our column generation methods: RP-CLG and FO-CLG. We fix $n = 1000, p = 100K, k_0 = 10, \rho = 0.1$, and consider a sequence of 50 values of λ:

$$\lambda = \kappa\lambda_{\max}, \quad \kappa \in \{0.01, 0.02, \ldots, 0.49, 0.50\}. \tag{58}$$

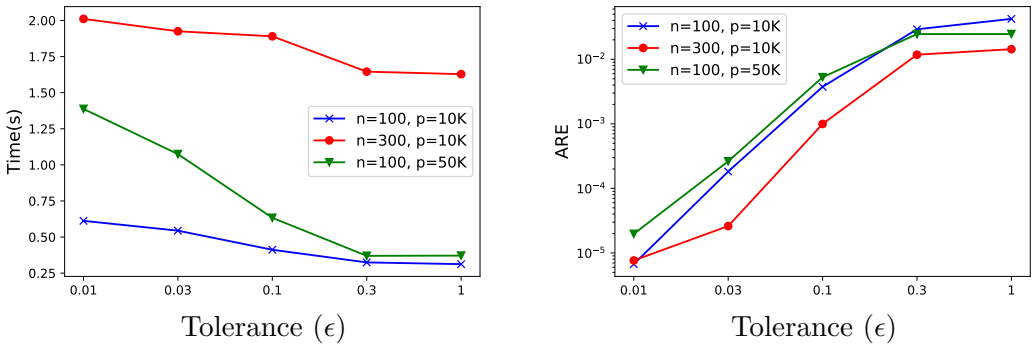

Figure 2: Runtime (s) and ARE of `FO-CLG` under different tolerance levels ϵ for the L1-SVM LP. We consider three different problem-sizes as indicated in the figure legends.

For convenience, let us denote the different values of λ by $\lambda_1 > \lambda_2 > \cdots > \lambda_m$—here, we take $m = 50$. For `RP-CLG`, we solve the problems in the order $\lambda_1, \lambda_2, \ldots, \lambda_{m-1}, \lambda_m$, and use the solution at λ_i as a warm start to compute the solution at λ_{i+1} using column generation. For `FO-CLG`, we solve the problems for different λ_i's *independently*—we use a solution from our first order method to initialize the initial set of columns for column generation. In the left panel of Figure 3, we present the runtime for both methods at each λ_i. We also present `CLG wo FO` which is the runtime of column generation excluding the time taken by the first order method. In the right panel of Figure 3, we present the support size of the solution $\hat{\boldsymbol{\beta}}$, which is denoted by "beta supp" in the figure. As shown in Figure 3, when λ decreases, the support size of the solution increases, and the runtimes of both `RP-CLG` and `FO-CLG` increase. The performance of `RP-CLG`, which performs warm-starts along the sequence of λ-values, appears to be superior to `FO-CLG`. As `FO-CLG` does not use regularization-path continuation, one can compute solutions for the different λ_i-values in parallel, which is not possible with the sequential approach `RP-CLG`. Note however, in our experiments, the L1-SVM solutions for the different λ_i values are computed sequentially and not in parallel. Based on this experiment, we recommend using `RP-CLG` when one wishes to compute a path of solutions to the L1-SVM problem, in a sequential fashion.

5.1.2 Synthetic Data sets for Large n and Small p

When $n \gg p$, we consider the numerical performance of the constraint generation procedure:

(i) `FO-CNG`: This is our constraint generation (CNG) method when initialized with a subsampling based first order (FO) heuristic as discussed in Section 4.4.

For `FO-CNG`, we use a reduced cost convergence threshold of $\epsilon = 0.01$. We limit the number of constraints to be added to at most 400. We note that the number 400 is somewhat ad hoc, and can be generally fine-tuned for improved performance.

We compare `FO-CNG` with several benchmarks: `SGD`, `SCS`, `FOM`, `Gurobi` as discussed in Section 5.1.1. We do not present the results for `PSM` as this was found to be much slower on these instances with large n (In some cases, `PSM` encountered numerical issues).

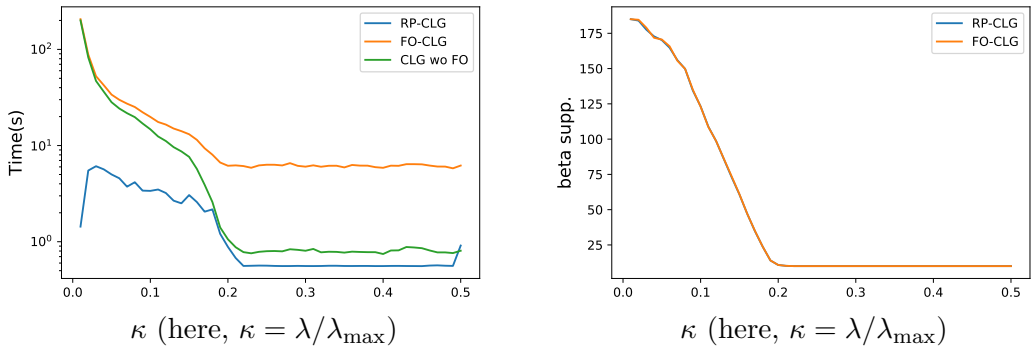

Figure 3: L1-SVM solutions on a regularization path. Here $n = 1K$ and $p = 100K$. The left figure presents runtime (s) versus κ, defined in (58). The right figure presents support size of $\boldsymbol{\beta}$ denoted by "beta supp.", i.e., number of nonzero SVM coefficients $\boldsymbol{\beta}$ for different values of κ. The two profiles for "beta supp", as available from algorithms `FO-CLG` and `RP-CLG`, are almost identical.

Table 2 presents the results for $\lambda = 0.001\lambda_{\max}$ and $\lambda = 0.01\lambda_{\max}$. For each choice of λ, we consider $(n, p) = (10K, 100)$, $(10K, 300)$ and $(50K, 100)$, and we present results averaged over 5 replications. As constraint generation leverages sparsity in $\boldsymbol{\xi}$, to get an idea of the sparsity of the problem we are dealing with, we present (i) the support size of the solution $\hat{\boldsymbol{\xi}}$, computed by `Gurobi`; and (ii) the number of constraints $|\mathcal{I}|$ in the restricted problem, upon termination of `FO-CNG`. From Table 2, it appears that `FO-CNG` outperforms other methods by a factor $4X \sim 30X$. In particular, `FO-CNG` has better performance when λ is small. Note that as we consider the setting where $n \gg p$, a small value of λ imparts less shrinkage on the SVM coefficients $\boldsymbol{\beta}$. Hence, the support size of $\hat{\boldsymbol{\xi}}$ is small—in other words, the number of mis-classified samples is small. As a result, the constraint generation method speeds up overall runtime making `FO-CNG` computationally attractive. Both `FO-CNG` and `Gurobi` reach high accuracy solutions. The accuracy of solutions obtained by `FO-CNG` is considerably higher compared to `SGD`, `SCS` and `FOM`. In this example we have $n \gg p$, and we observe that `SGD` works well compared to the examples in Section 5.1.1 where $p \gg n$.

Finally, we note that in the examples considered in Table 2 when λ is very large and the support size of $\hat{\boldsymbol{\xi}}$ is close to n, the runtime of `FO-CNG` is likely going to increase.

5.1.3 Synthetic Data sets for $n \approx p$

We study the performance of the algorithms when both n and p are comparable and moderately large. We consider the following method with both column and constraint generation:

(**j**) `FO-CLCNG`: This is the combined column-and-constraint generation method, denoted by the shorthand CLCNG (i.e., Algorithm 4), initialized with the first order method discussed[11] in Section 4.4.3. The column/constraint generation reduced cost thresholds are set to be equal $\epsilon := \epsilon_1 = \epsilon_2$.

11. For the subsampling heuristic, once the average estimate was obtained, we took the top 200 coefficients with largest magnitude, to initialize the set of columns for column generation.

$\lambda = 0.001\lambda_{\max}$

	$n = 10K, p = 100$		$n = 10K, p = 300$		$n = 50K, p = 100$	
Method	Time (s)	ARE	Time (s)	ARE	Time (s)	ARE
SGD	54.2(4.05)	1.8e-02(8.78e-04)	117.4(2.89)	4.5e-02(2.19e-03)	313.1(3.19)	7.4e-03(3.77e-04)
SCS	51.4(4.03)	1.7e-04(4.31e-05)	117.6(1.74)	5.2e-04(1.90e-04)	241.9(11.19)	3.9e-05(7.35e-06)
FOM	170.1(3.40)	2.8e-03(1.39e-04)	207.0(9.03)	5.9e-03(1.12e-04)	1147.4(35.85)	5.9e-04(1.73e-05)
Gurobi	66.3(3.52)	0.0e+00(0.00e+00)	133.8(5.48)	0.0e+00(0.00e+00)	626.6(42.49)	0.0e+00(0.00e+00)
FO-CNG	3.0(0.06)	1.3e-05(1.12e-05)	6.3(0.24)	2.1e-15(0.00e+00)	20.8(0.22)	0.0e+00(0.00e+00)
$\|\hat{\boldsymbol{\xi}}\|_0$ & $\|\mathcal{I}\|$	87.8 & 443.8		88.8 & 362.2		538.6 & 2277.8	

$\lambda = 0.01\lambda_{\max}$

	$n = 10K, p = 100$		$n = 10K, p = 300$		$n = 50K, p = 100$	
Method	Time (s)	ARE	Time (s)	ARE	Time (s)	ARE
SGD	55.6(2.88)	4.0e-03(3.51e-04)	130.1(4.50)	8.2e-03(2.38e-04)	326.7(5.36)	3.6e-03(1.28e-04)
SCS	28.1(3.99)	2.3e-05(3.62e-06)	79.6(7.68)	3.2e-05(6.57e-06)	112.9(7.21)	1.2e-05(3.27e-06)
FOM	47.8(0.90)	5.8e-04(1.05e-05)	68.8(4.00)	1.0e-03(3.21e-05)	297.4(0.82)	1.3e-04(5.87e-06)
Gurobi	49.3(1.84)	0.0e+00(0.00e+00)	104.1(5.43)	0.0e+00(0.00e+00)	377.8(15.05)	0.0e+00(0.00e+00)
FO-CNG	9.0(0.26)	3.5e-06(3.16e-06)	11.4(0.32)	8.3e-06(3.14e-06)	32.4(0.83)	1.3e-06(4.80e-07)
$\|\hat{\boldsymbol{\xi}}\|_0$ & $\|\mathcal{I}\|$	501.4 & 1090.2		472.2 & 813.8		2559.4 & 3472.8	

Table 2: *L1-SVM, Synthetic data set ($n \gg p$)*: Training time for L1-SVM versus different benchmarks on synthetic data sets. We show two different values of λ: $\lambda = 0.001\lambda_{\max}$ for the top table and $\lambda = 0.01\lambda_{\max}$ for the bottom table. Each table presents results for different values of (n, p) with $n \gg p$. For each table, the last row presents the number of nonzeros in $\boldsymbol{\xi}$ in an optimal solution to the problem and the number of active constraints $|\mathcal{I}|$ upon termination of FO-CNG. For a cleaner display, all positive values below 10^{-15} are displayed as zero.

For FO-CLCNG, we use $\epsilon = 0.01$ and limit the maximum number of columns/constraints that are added at an iteration to 400. We compare FO-CLCNG with benchmarks including SGD, SCS, FOM, Gurobi under the same setting as Section 5.1.1. Once again, PSM is found to be significantly slow as n is large, hence we do not include it in our results. Table 3 presents the results for $\lambda = 0.01\lambda_{\max}$ and $\lambda = 0.1\lambda_{\max}$, with $(n, p) = (3K, 3K)$, $(2K, 5K)$ and $(5K, 2K)$. Note that in these examples, $\mathbf{X}$ is dense so we do not consider larger problem-sizes—larger n, p values with a sparse $\mathbf{X}$ are considered in Section 5.1.4.

Table 3 presents runtimes (s) and ARE values across 5 independent experiments. In addition, to get an idea of the sparsity level of the problem, we also present the support sizes of the solutions $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\xi}}$, as computed by Gurobi. We also list the number of columns and constraints i.e., $|\mathcal{J}|$ and $|\mathcal{I}|$, in the restricted problem, upon termination of FO-CLCNG. As shown in Table 3, for $\lambda = 0.01\lambda_{\max}$, FO-CLCNG has a $7X \sim 30X$ speedup over other methods. For $\lambda = 0.1\lambda_{\max}$, FO-CLCNG has a $4X \sim 50X$ speedup over other methods. At the same time, it is important to note that the ARE of FO-CLCNG is around $10^{-5} \sim 10^{-6}$—notably better than that of SCS, FOM and SGD.

Performance under different tolerance levels ϵ: Table 3 presents results of FO-CLCNG for a fixed value of $\epsilon = 0.01$. In Figure 4, to understand sensitivity to the choice of ϵ, we present the runtime and ARE of FO-CLCNG under different choices of $\epsilon \in \{0.01, 0.03, 0.1, 0.3, 1\}$ for $\lambda = 0.01\lambda_{\max}$. The presented results are the mean of 5 independent replications. Similar to Figure 2, it can be seen in Figure 4, that as ϵ decreases from 1 to 0.01, the ARE of

$\lambda = 0.01\lambda_{\max}$

	$n = 3K, p = 3K$		$n = 2K, p = 5K$		$n = 5K, p = 2K$	
Method	Time (s)	ARE	Time (s)	ARE	Time (s)	ARE
`SGD`	393.3(32.06)	1.4e-01(7.79e-03)	387.3(25.89)	4.2e-01(1.20e-02)	486.9(46.32)	4.5e-02(1.28e-03)
`SCS`	281.8(34.91)	1.7e-04(1.73e-05)	261.7(37.13)	3.0e-04(7.05e-05)	426.3(37.83)	1.3e-04(2.44e-05)
`FOM`	100.6(9.74)	5.2e-03(1.27e-04)	93.4(5.33)	7.9e-03(1.83e-04)	142.3(12.46)	3.0e-03(5.39e-05)
`Gurobi`	103.9(27.61)	0.0e+00(0.00e+00)	109.8(31.84)	0.0e+00(0.00e+00)	585.9(57.41)	0.0e+00(0.00e+00)
`FO-CLCNG`	14.1(0.40)	3.5e-07(2.59e-07)	9.9(0.11)	8.2e-06(2.11e-06)	20.2(0.44)	3.6e-06(2.82e-06)
$\|\hat{\boldsymbol{\beta}}\|_0$ & $\|\mathcal{J}\|$	170.2 & 646.4		162.0 & 692.0		188.0 & 634.2	
$\|\hat{\boldsymbol{\xi}}\|_0$ & $\|\mathcal{I}\|$	132.8 & 565.2		87.6 & 533.4		221.4 & 668.8	

$\lambda = 0.1\lambda_{\max}$

	$n = 3K, p = 3K$		$n = 2K, p = 5K$		$n = 5K, p = 2K$	
Method	Time (s)	ARE	Time (s)	ARE	Time (s)	ARE
`SGD`	521.6(40.17)	3.5e-03(1.80e-04)	527.9(26.79)	5.8e-03(6.02e-04)	539.5(24.21)	2.4e-03(2.41e-04)
`SCS`	150.5(22.98)	3.4e-04(4.09e-05)	178.7(9.48)	3.8e-04(1.12e-04)	259.2(26.38)	8.2e-05(1.40e-05)
`FOM`	62.2(4.98)	2.6e-04(1.36e-05)	60.5(2.72)	4.4e-04(4.86e-05)	71.0(5.52)	1.3e-04(1.47e-05)
`Gurobi`	47.6(6.74)	0.0e+00(0.00e+00)	44.4(5.67)	0.0e+00(0.00e+00)	354.9(116.66)	0.0e+00(0.00e+00)
`FO-CLCNG`	11.7(0.14)	3.1e-06(1.65e-06)	8.3(0.13)	8.2e-07(6.71e-07)	18.7(0.16)	1.2e-06(6.69e-07)
$\|\hat{\boldsymbol{\beta}}\|_0$ & $\|\mathcal{J}\|$	56.8 & 255.8		68.4 & 322.2		48.0 & 235.8	
$\|\hat{\boldsymbol{\xi}}\|_0$ & $\|\mathcal{I}\|$	750.8 & 1053.2		504.8 & 751.2		1287.0 & 1657.4	

Table 3: **L1-SVM, Synthetic data set ($n \approx p$)**: Training time for L1-SVM versus state-of-art methods on synthetic data sets. We show two different values of λ: $\lambda = 0.01\lambda_{\max}$ for the top table and $\lambda = 0.1\lambda_{\max}$ for the bottom table. Each table presents results for different values of (n, p) with $n \approx p$. For each table, the last row presents the number of nonzeros in $\boldsymbol{\beta}$ and $\boldsymbol{\xi}$ in an optimal solution to the problem and the number of active variables and constraints, upon termination of `FO-CLCNG`. For a cleaner display, all positive values below 10^{-15} are displayed as zero.

`FO-CLCNG` decreases from 10^{-1} to $10^{-5} \sim 10^{-6}$, while the runtime only increases slightly. Therefore, a tolerance of $\epsilon = 0.01$ appears to be sufficiently small and leads to a fairly high accuracy for the numerical experiments considered.

5.1.4 Real Data Sets with Large n and p

Finally, we assess the quality of our hybrid column-and-constraint generation method: `FO-CLCNG` on large real data sets. For a fair comparison, we compare different methods in terms of their ability to optimize the same L1-SVM optimization problem. We consider three popular open-source data sets `rcv1`, `news20` and `real-sim` that can be found at `https://www.csie.ntu.edu.tw/~cjlin/liblinear/`. We also consider a larger data set derived from `rcv1`: We augment the original features of `rcv1` with noisy features obtained by randomly selecting a collection of features from the original data set, and randomly permuting the rows of the selected features. We denote this augmented data set by `rcv1-aug`. Similarly, we form an augmented version of the data set `real-sim` that we denote by `real-sim-aug`. Note that all these data sets are sparse—the number of nonzero entries in $\mathbf{X}$, denoted by $\text{nnz}(\mathbf{X})$, is quite small compared to np; and we use sparse matrices (scipy implementation) to deal with sparse matrix/vector multiplications.

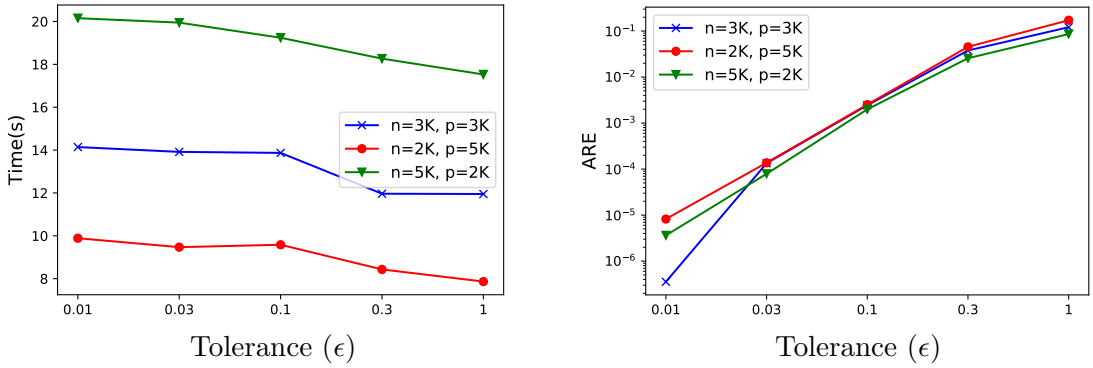

Figure 4: Runtime (s) and optimization error (ARE) of `FO-CLCNG` under different choices of the tolerance threshold ϵ. We consider 3 different (n, p)-values. This figure mirrors Figure 2 that shows results for column generation alone: `FO-CLG`.

We compare `FO-CLCNG` with benchmark methods `Gurobi`, `SGD` and `SCS`. All methods are run under the settings explained in Section 5.1.1, except that here we run `SGD` for $20,000$ epochs to arrive at an ARE$\sim 10^{-2}$—10^{-3}.

Table 4 presents the sizes of the data sets considered, the runtime (in seconds) and ARE for different algorithms. We consider a sequence of 11 values of $\lambda = \kappa\lambda_{\max}$ where κ lies in the range $\kappa \in [0.003, 0.3]$. To make the comparisons fair, all algorithms are run independently for different λ-values; and we present the average runtime and ARE for a value of λ. In the last row of each sub-table, we present the minimum, mean and maximum of the value $\|\hat{\boldsymbol{\beta}}\|_0 + \|\hat{\boldsymbol{\xi}}\|_0$ over the path of λ-values. The numbers are presented in the form of the triplet "(min, mean, max)"; and $(\hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\xi}})$ is the solution obtained from `FO-CLCNG`.

On the data sets `rcv1` and `rcv1-aug`, our proposed method `FO-CLCNG` outperforms `SCS` and `Gurobi` in runtime by a factor of $3X \sim 8X$ and delivers solutions of higher accuracy i.e., lower ARE. For other two data sets `news20` and `real-sim-aug`, algorithms `Gurobi` and `SCS` would run out of memory (we set a 16GB memory limit) for some values of λ. As our column-and-constraint generation procedure operates on a smaller reduced problem, it consumes less memory. The solutions of `FO-CLCNG` have high accuracy with ARE around 10^{-6}. By comparing `FO-CLCNG` and `SGD`, we see that the runtime of 20,000 epochs of `SGD` is comparable to that of `FO-CLCNG` on the `rcv1` data set, while it is slower than `FO-CLCNG` on other data sets. Note however that the optimization accuracy of the SGD solutions is significantly worse compared to `FO-CLCNG`.

5.2 Computational Results for Group-SVM

We now study the performance of the column generation algorithm presented in Section 2.4 for the Group-SVM Problem (3).

Data Generation: Here, the covariates are drawn from a multivariate Gaussian with covariance $\boldsymbol{\Sigma}$. The p covariates are divided into G groups each of the same size p_G. Within each group, covariates have pairwise correlation of ρ, and covariates are uncorrelated across

Data set: rcv1 ($n = 16194, p = 47237$, nnz=1.20e+06)		
method	Time (s)	ARE
`Gurobi`	442.0	0.0e+00
`SCS`	1185.9	2.2e-04
`SGD`	256.3	1.2e-02
`FO-CLCNG`	142.9	1.0e-06
$\|\hat{\boldsymbol{\xi}}\|_0 + \|\hat{\boldsymbol{\beta}}\|_0$	(3212, 6538, 10580)	

Data set: rcv1-aug ($n = 16194, p = 236185$, nnz=6.00e+06)		
method	Time (s)	ARE
`Gurobi`	1590.5	0.0e+00
`SCS`	3318.8	9.2e-04
`SGD`	1419.3	1.4e-02
`FO-CLCNG`	442.2	1.1e-06
$\|\hat{\boldsymbol{\xi}}\|_0 + \|\hat{\boldsymbol{\beta}}\|_0$	(3997, 6765, 10512)	

Data set: news20 ($n = 15997, p = 1355191$, nnz=7.31e+06)		
method	Time (s)	ARE
`Gurobi`	-	-
`SCS`	-	-
`SGD`	2014.4	4.3e-03
`FO-CLCNG`	112.0	0.0e+00
$\|\hat{\boldsymbol{\xi}}\|_0 + \|\hat{\boldsymbol{\beta}}\|_0$	(7106, 10010, 13853)	

Data set: real-sim-aug ($n = 57847, p = 104795$, nnz=1.47e+07)		
method	Time (s)	ARE
`Gurobi`	-	-
`SCS`	-	-
`SGD`	2908.9	2.5e-03
`FO-CLCNG`	955.3	0.0e+00
$\|\hat{\boldsymbol{\xi}}\|_0 + \|\hat{\boldsymbol{\beta}}\|_0$	(11975, 18931, 26632)	

Table 4: L1-SVM on real data sets where both n, p are large. We compare our method `FO-CLCNG` versus other benchmarks in terms of runtime and ARE on a range of λ-values, as discussed in the text. A "-" means that the method would not run due to memory limitations and/or numerical problems. The last row of every sub-table provides the (minimum, average, maximum)-tuple of the support-size $\|\hat{\boldsymbol{\xi}}\|_0 + \|\hat{\boldsymbol{\beta}}\|_0$ where the minimum, average and maximum values are taken across the sequence of λ-values considered.

groups. All variances are equal. Half of the samples are from the $+1$ class with population mean $\boldsymbol{\mu}_+ = (\mathbf{1}_{p_G}, \ldots, \mathbf{1}_{p_G}, \mathbf{0}_{p_G}, \ldots, \mathbf{0}_{p_G})$ where we have k_0 many sub-vectors $\mathbf{1}_{p_G}$. The remaining half of the samples from class -1 have population mean $\boldsymbol{\mu}_- = -\boldsymbol{\mu}_+$. In the following example, we take $\rho = 0.1$, $p_G = 10$ and $k_0 = 10$.

Group-SVM ($p \gg n$)

	$n = 100, p = 10K$		$n = 300, p = 10K$		$n = 100, p = 30K$	
Method	Time (s)	ARE	Time (s)	ARE	Time (s)	ARE
`SCS`	78.1(16.05)	8.7e-04(1.10e-04)	82.6(14.65)	7.0e-04(2.89e-04)	287.4(21.38)	1.1e-02(5.55e-03)
`Gurobi`	120.8(5.16)	0.0e+00(0.00e+00)	321.7(14.19)	0.0e+00(0.00e+00)	503.8(26.53)	0.0e+00(0.00e+00)
`FO-CLG`	2.1(0.17)	0.0e+00(0.00e+00)	3.8(0.13)	0.0e+00(0.00e+00)	2.3(0.14)	0.0e+00(0.00e+00)

Table 5: Training time (s) and ARE for Group-SVM versus various benchmarks on synthetic data sets, $\lambda = 0.1\lambda_{\max}$. For a cleaner display, all positive values below 10^{-15} are displayed as zero.

Comparison with benchmarks: We compare our column generation method `FO-CLG` with `SCS` and `Gurobi`. Both `SCS` and `Gurobi` are called through CVXPY under its default settings. Our method `FO-CLG` (cf Section 2.4) applies column generation after initialization

with the block coordinate descent procedure (cf Section 4.3). We use a smoothing parameter $\tau = 0.2$ to smooth the hinge-loss, and use a CD method restricted to the top n groups obtained via correlation screening (cf Section 4.4.1). We use a reduced cost tolerance of $\epsilon = 0.01$ in the column generation method. Table 5 shows the results on synthetic data sets with $(n, p) = (100, 10K)$, $(300, 10K)$ and $(100, 30K)$ and set $\lambda = 0.1\lambda_{\max}$. The reported values are based on 5 independent replications. We can see that on these examples, `FO-CLG` outperforms the other two methods by a factor of at least $30X$ in runtime, and also delivers a solution with high accuracy, as seen from the ARE values.

5.3 Computational Results for Slope-SVM

We present the computational performance of the column-and-constraint generation methods presented in Section 3.3 for the Slope-SVM Problem (4). The synthetic data is simulated as in Section 5.1.1—we set $n = 100,\ k_0 = 10,\ \rho = 0.1$.

Comparison when λ_is are not all distinct: We are not aware of any publicly available specialized implementation for the Slope SVM problem. We use the `CVXPY` modeling framework to model (See Section A.1) the Slope SVM problem and solve it using state-of-the art general purpose solvers like Ecos and Gurobi. We first consider a special instance of the Slope penalty (28) that corresponds to the coefficients $\lambda_i = 2\tilde{\lambda}$ for $i \leq k_0$ and $\lambda_i = \tilde{\lambda}$ for $i > k_0$; where $\tilde{\lambda} = 0.01\lambda_{\max}$. We solve the resulting problem with both the Ecos and Gurobi solvers, denoted by "`CVXPY Ecos`" and "`CVXPY Gurobi`", respectively. We compare them with our proposed column-and-constraint generation algorithm, referred to as "`FO-CLCNG`". For our method, we first run the first order algorithm presented in Section 4.3 with $\tau = 0.2$. We restrict the optimization of the first order method to the $10n$ columns with highest absolute correlations with the response. The column-and-constraint generation algorithm (cf Section 3.3) uses a tolerance level of $\epsilon = 0.001$. We limit the number of columns added at each iteration to 10. For reference, we also report the run time of our algorithm by excluding the time taken by the initialization step—this is denoted by "`CLCNG wo FO`". The results, averaged over 5 replications, are presented in Table 6. Results in Table 6 indicate that our proposed method `FO-CLCNG` exhibits a 50X—110X improvement compared to competing solvers. In some cases, when $p \geq 50K$, `CVXPY Ecos` encounters numerical problems and hence would not run. On the other hand, we note that `FO-CLCNG` has the best ARE for $p \geq 20K$ even when compared with `CVXPY Gurobi`. This may be because the model size is large and `CVXPY Gurobi` appears to use a low-accuracy termination condition.

Comparison when λ_is are distinct: Here we consider a different sequence of λ-values. Following Bellec et al. (2018), we set $\lambda_j = \sqrt{\log(2p/j)}\tilde{\lambda}$ with $\tilde{\lambda} = 0.01\lambda_{\max}$. We observed that `CVXPY` could not handle even small instances of this problem—in particular, the Ecos solver crashed for $n = 100, p = 200$. We compare our proposed method `FO-CLCNG` with the first order method (cf. Section 4.3) applied to the full smoothed Slope-SVM problem with $\tau = 0.2$. Due to the high per iteration cost of the first order method `FOM`, we terminate the method after a few iterations—the associated ARE is reported within parenthesis. Table 7 compares our methods—we use the same synthetic data set as in the previous Slope-SVM example. We average the results over 5 replications though the first order method was run for one replication due to its long run time.

Slope-SVM ($p \gg n$)

	FO-CLCNG		CLCNG wo FO	CVXPY Ecos		CVXPY Gurobi	
p	Time (s)	ARE	Time (s)	Time (s)	ARE	Time (s)	ARE
10k	1.7(0.07)	1.3e-06	1.2(0.05)	64.3(6.81)	7.0e-12	84.0(2.86)	0.0e+00
20k	3.0(0.01)	0.0e+00	2.5(0.00)	130.5(3.17)	1.7e-05	221.9(5.64)	1.7e-05
50k	7.1(0.35)	0.0e+00	6.6(0.34)	-	-	842.3(20.06)	9.1e-06
100k	16.5(0.02)	0.0e+00	15.9(0.02)	-	-	1837.3(70.44)	4.1e-06

Table 6: Training times and ARE of our column-and-constraint generation method for Slope-SVM versus CVXPY—we took $n = 100$, $\lambda_i/\lambda_j = 2$ for all $i \in [k_0]$ and $j > k_0$, as described in the text. When the number of features are in the order of tens of thousands, our proposed method FO-CLCNG enjoys nearly a 100X speedup in run time. A '-' symbol denotes that the corresponding algorithm encountered numerical problems.

Slope-SVM ($p \gg n$)

	FO-CLCNG		CLCNG wo FO	FOM	
p	Time (s)	ARE	Time (s)	Time (s)	ARE
10k	1.4(0.11)	0.0e+00(0.00e+00)	0.5(0.05)	164.6	3.0e-01
20k	1.6(0.09)	0.0e+00(0.00e+00)	0.7(0.04)	427.3	3.2e-01
50k	3.3(0.20)	0.0e+00(0.00e+00)	2.4(0.14)	1633.8	3.2e-01

Table 7: Training times and ARE of our column-and-constraint generation procedure for Slope-SVM with coefficients $\lambda_j = \sqrt{\log(2p/j)}\tilde{\lambda}$, as discussed in the text. We consider synthetic data sets with $n = 100$ and a large number of features. Our proposed approach FO-CLCNG appears to outperform the first order methods in runtimes by at least 100-times. In addition, FO-CLCNG delivers solutions of higher accuracy compared to FOM.

Acknowledgments

The authors would like to thank the Action Editor and three anonymous reviewers for their helpful comments and constructive feedback that helped improve the paper. This research was supported in part, by grants from the Office of Naval Research: ONR-N000141812298 (YIP), the National Science Foundation: NSF-IIS-1718258 and IBM.

Appendix A. Appendix

A.1 Another Formulation for the Slope Norm

Without loss of generality, we consider a vector $\boldsymbol{\alpha} \geq \mathbf{0}$. Now note that for every $m \in [p]$, we can represent $\alpha_{(1)} + \ldots + \alpha_{(m)} \leq s_m$ as

$$\alpha_{(1)} + \ldots + \alpha_{(m)} \leq s_m, \quad \boldsymbol{\alpha} \geq \mathbf{0} \iff \begin{cases} \mathbf{0} \leq \boldsymbol{\alpha} \leq \theta_m \mathbf{1} + \mathbf{v}_m \\ m\theta_m + \mathbf{1}^T \mathbf{v}_m \leq s_m, \end{cases} \tag{59}$$

with variables $\boldsymbol{\alpha}, \mathbf{v}_m \in \mathbb{R}^p, \theta_m \in \mathbb{R}$ and $\mathbf{1} \in \mathbb{R}^p$ being a vector of all ones. Note that the rhs formulation in (59) has $O(p)$ variables and $O(p)$-constraints. Note that we can write:

$$\sum_{j=1}^{p} \lambda_j \alpha_{(j)} = \sum_{m=1}^{p} \tilde{\lambda}_m (\alpha_{(1)} + \ldots + \alpha_{(m)}),$$

where, $\tilde{\lambda}_m = \lambda_m - \lambda_{m-1}$ for all $m \in \{1, \ldots, p\}$. Therefore, representing $\sum_{j=1}^{p} \lambda_j \alpha_{(j)} \leq \eta$ will require a representation (59) for $m = 1, \ldots, p$—this will lead to a formulation with $O(p^2)$ variables and $O(p^2)$ constraints, which can be quite large as soon as p becomes a few hundred.

References

Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Convex optimization with sparsity-inducing norms. *Optimization for Machine Learning*, 5:19–53, 2011.

P. Balamurugan, Anusha Posinasetty, and Shirish Shevade. ADMM for training sparse structural SVMs with augmented l1 regularizers. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 684–692. SIAM, 2016.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

Stephen R. Becker, Emmanuel J. Candès, and Michael C. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical programming computation*, 3(3):165, 2011.

Pierre C. Bellec, Guillaume Lecué, Alexandre B. Tsybakov, et al. Slope meets Lasso: improved oracle bounds and optimality. *The Annals of Statistics*, 46(6B):3603–3642, 2018.

Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

Malgorzata Bogdan, Ewout van den Berg, Weijie Su, and Emmanuel Candes. Statistical estimation and testing via the sorted l1 norm. *arXiv preprint arXiv:1310.1969*, 2013.

Małgorzata Bogdan, Ewout van den Berg, Chiara Sabatti, Weijie Su, and Emmanuel J. Candès. Slope adaptive variable selection via convex optimization. *The annals of applied statistics*, 9(3):1103, 2015.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

Paul S. Bradley and Olvi L. Mangasarian. Feature selection via concave minimization and support vector machines. In *ICML*, volume 98, pages 82–90, 1998.

Laurent Condat. Fast projection onto the simplex and the ℓ_1 ball. *Mathematical Programming*, 158(1):575–585, 2016.

George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

Jacques Desrosiers and Marco E. Lübbecke. A primer in column generation. In *Column generation*, pages 1–32. Springer, 2005.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Lester Randolph Ford Jr. and Delbert R. Fulkerson. A suggested computation for maximal multi-commodity network flows. *Management Science*, 5(1):97–101, 1958.

Vojtěch Franc and Soeren Sonnenburg. Optimized cutting plane algorithm for support vector machines. In *Proceedings of the 25th international conference on Machine learning*, pages 320–327. ACM, 2008.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010a.

J.H. Friedman, T. Hastie, and R. Tibshirani. Regularized paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1), 2010b.

LLC Gurobi Optimization. Gurobi optimizer reference manual, 2021. URL `http://www.gurobi.com`.

Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, 5(Oct): 1391–1415, 2004.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer New York, 2 edition, 2009.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear SVM. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

Junzhou Huang and Tong Zhang. The benefit of group sparsity. *The Annals of Statistics*, 38(4):1978–2004, 2010.

Thorsten Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

Jason D. Lee, Qiang Liu, Yuekai Sun, and Jonathan E. Taylor. Communication-efficient sparse regression. *Journal of Machine Learning Research*, 18(5):1–30, 2017.

Xingguo Li, Tuo Zhao, Xiaoming Yuan, and Han Liu. The flare package for high dimensional linear regression and precision matrix estimation in r. *J. Mach. Learn. Res.*, 16:553–557, 2015.

Olvi L. Mangasarian. Exact 1-norm support vector machines via unconstrained convex differentiable minimization. *Journal of Machine Learning Research*, 7(Jul):1517–1530, 2006.

Jean-Jacques Moreau. Dual convex functions and proximal points in a Hilbert space. *CR Acad. Sci. Paris Ser. At Math.*, 255:2897–2899, 1962.

Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Norwell, 2004.

Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

Yurii Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

Brendan O'Donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. SCS: Splitting conic solver, version 2.1.2. `https://github.com/cvxgrp/scs`, November 2019.

Haotian Pang, Han Liu, Robert J. Vanderbei, and Tuo Zhao. Parametric simplex method for sparse learning. In *Advances in Neural Information Processing Systems*, pages 188–197, 2017.

Zhiwei Qin, Katya Scheinberg, and Donald Goldfarb. Efficient block-coordinate descent algorithms for the group Lasso. *Mathematical Programming Computation*, 5(2):143–169, 2013.

Tim Robertson. *Order restricted statistical inference*. Wiley, New York., 1988.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM, 2007.

Robert Tibshirani, Jacob Bien, Jerome Friedman, Trevor Hastie, Noah Simon, Jonathan Taylor, and Ryan J. Tibshirani. Strong rules for discarding predictors in Lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74 (2):245–266, 2012.

Sara A van de Geer. *Empirical Processes in M-estimation*, volume 6. Cambridge University Press, 2000.

Ewout Van Den Berg and Michael P. Friedlander. SPGL1: A solver for large-scale sparse reconstruction, June 2007. http://www.cs.ubc.ca/labs/scl/spgl1.

Ewout Van Den Berg and Michael P. Friedlander. Probing the Pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008. doi: 10.1137/080714488. URL `http://link.aip.org/link/?SCE/31/890`.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

Stephen J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1): 3–34, 2015.

Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *The Journal of Machine Learning Research*, 11:3183–3234, 2010.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Xiangrong Zeng and Mário A. T. Figueiredo. The ordered weighted ℓ_1 norm: Atomic formulation, dual norm, and projections. *CoRR*, abs/1409.4271, 2014. URL `http://arxiv.org/abs/1409.4271`.