The published version can be found here:
"Sparse Convex Regression." Bertsimas, Dimitris, Nishanth Mundru. INFORMS Journal of Computing Vol. 33, No. 1(2021):262-279.
DOI: 10.1287/ijoc.2020.0954

Sparse convex regression

Dimitris Bertsimas
Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA; dbertsim@mit.edu

Nishanth Mundru
Operations Research Center, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, USA; nmundru@mit.edu

Abstract

We consider the problem of best $k-$subset convex regression using n observations in d variables. For the case without sparsity, we develop a scalable algorithm for obtaining high quality solutions in practical times that compare favorably with other state of the art methods. We show that by using a cutting plane method, the least squares convex regression problem can be solved for sizes $(n, d) = (10^4, 10)$ in minutes and $(n, d) = (10^5, 10^2)$ in hours. Our algorithm can be adapted to solve variants such as finding the best convex or concave functions with coordinate-wise monotonicity, norm bounded subgradients, and minimize the ℓ_1 loss - all with similar scalability to the least squares convex regression problem. Under sparsity, we propose algorithms which iteratively solve for the best subset of features based on first order and cutting plane methods. We show that our methods scale for sizes $(n, d, k) = (10^4, 10^2, 10)$ in minutes and $(n, d, k) = (10^5, 10^2, 10)$ in hours. We demonstrate that these methods control for the false discovery rate effectively.

1. Introduction

Given data $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, we consider the problem of finding a convex function on the $\mathbf{x} \in \mathbb{R}^d$ variables (features) that best fits the dependent variables $y \in \mathbb{R}$. Formally, we wish to estimate a function $f : \mathbb{R}^d \to \mathbb{R}$ where

$$y = f(\mathbf{x}) + \epsilon \tag{1}$$

with the requirement that f be a convex function. Here the random noise ϵ is assumed to have zero mean. Note that one can equivalently perform concave regression, as the requirement that f is convex is identical to restricting $-f$ to be concave. As we discuss next, such convexity/concavity constraints arise naturally in several settings. Such problems fall in the general class of shape constrained function estimation.

Shape constrained regression problems have many applications in various fields such as, but not limited to, operations research, econometrics, geometric programming [Magnani and Boyd, 2009], image analysis [Goldenshluger and Zeevi, 2006], and target reconstruction [Lele et al., 1992]. In operations research, these problems arise in reinforcement learning [Shapiro et al., 2009], [Hannah et al., 2014], in resource allocation [Topaloglu and Powell, 2003], and while analyzing performance measures of queueing networks [Chen and Yao, 2001]. In economics, such problems are encountered when demand [Varian, 1982], utility functions [Varian, 1984], and production functions [Allon et al., 2007] are assumed to be concave. For a more detailed list of applications, see Lim and Glynn [2012] and Hannah and Dunson [2013].

The convex least squares estimator is the solution of the following generalized regression problem:

$$\min_{f \in \mathcal{C}} \quad \frac{1}{2} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2, \tag{2}$$

where $\mathcal{C}$ represents the space of convex functions on $\mathbb{R}^d$. Note that Problem (2) is an optimization problem over functions. Surprisingly, this can be written equivalently as a finite dimensional convex quadratic optimization problem where the variables are the function values and subgradients at each of the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ [Boyd and Vandenberghe, 2004]. As part of the constraints, we enforce the convexity condition, i.e., the graph of the convex function lies above each of its tangent planes. More precisely, this convexity condition implies that given any point $\mathbf{x}_i$, the value of f at $\mathbf{x}_i$ is greater or equal to the value of any tangent hyperplane of f evaluated at $\mathbf{x}_i$. Clearly any convex function has a nonempty subdifferential at every point, and the existence of such tangent planes is guaranteed. For this problem, it suffices to enforce this condition for all $n(n-1)$ pairs of points $\mathbf{x}_i, \mathbf{x}_j,\ 1 \leq i, j \leq n$.

The resulting quadratic optimization problem with variables $(\boldsymbol{\theta}, \{\boldsymbol{\xi}_i\}_{i=1}^n)$ is given as follows.

$$\begin{aligned} \min_{\boldsymbol{\theta}, \{\boldsymbol{\xi}_i\}_{i=1}^n} \quad & \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta_i)^2 \\ \text{subject to} \quad & \theta_i + \boldsymbol{\xi}_i^T (\mathbf{x}_j - \mathbf{x}_i) \leq \theta_j \quad \forall i, j, \\ & \boldsymbol{\theta} \in \mathbb{R}^n, \\ & \boldsymbol{\xi}_i \in \mathbb{R}^d \ \forall i. \end{aligned} \tag{3}$$

The variables θ_i represent the values of $f(\mathbf{x}_i)$, and $\boldsymbol{\xi}_i$ belongs to the subdifferential set of the convex function f at each $\mathbf{x}_i$. The solution to this problem $\boldsymbol{\theta}^*$ is referred to as the convex least squares estimator (CLSE). Note that we recover the usual least squares linear regression problem by setting $\boldsymbol{\xi}_i = \boldsymbol{\xi}\ \forall i$ and $\theta_i = \boldsymbol{\xi}^T \mathbf{x}_i\ \forall i$.

We note that the feasible set of Problem 3 can be unbounded, that may lead to potential instability. There can be multiple values of the subgradients leading to the same objective value. Hence, we propose solving the following regularized optimization problem, for a given $\lambda > 0$,

$$\begin{aligned} \min_{\boldsymbol{\theta}, \{\boldsymbol{\xi}_i\}_{i=1}^n} \quad & \frac{1}{2} \sum_{i=1}^{n} (y_i - \theta_i)^2 + \frac{\lambda}{2} \sum_{i=1}^{n} \|\boldsymbol{\xi}_i\|^2 \\ \text{subject to} \quad & \theta_i + \boldsymbol{\xi}_i^T (\mathbf{x}_j - \mathbf{x}_i) \leq \theta_j \quad \forall i, j, \\ & \boldsymbol{\theta} \in \mathbb{R}^n, \\ & \boldsymbol{\xi}_i \in \mathbb{R}^d \ \forall i. \end{aligned} \tag{4}$$

By adding a regularization term on the subgradients, which leads to a strongly convex objective, the subgradients $\xi_{i,j}$ cannot take any value for a given objective value and feasibility.

1.1. Related literature

In this section, we review the relevant literature. Recently, there has been considerable interest in shape constrained regression among the statistics community. Seijo and Sen [2011] and Lim and Glynn [2012] characterize and show consistency of the CLSE. Seijo and Sen [2011] use off-the-shelf interior point solvers (like MOSEK, cvx) for solving the problem. But these solvers do not scale well for $n \geq 300$ due to the presence of $O(n^2)$ constraints. This motivated the recent work by Mazumder et al. [2015] which presents a first order method based on the

Alternating Direction Method of Multipliers (ADMM) to compute the optimal solutions for the least squares convex regression problem. They demonstrate the flexibility of their approach in the presence of monotonicity constraints, and bounded subgradients. Their method solves instances of sizes of $n \approx 1000$ to an accuracy of 10^{-3} in a few seconds, and moderate accuracy solutions for $n \approx 5000$ in a few minutes. However, their method cannot be easily extended for least absolute deviation convex regression (where the loss function is the ℓ_1 norm rather than the ℓ_2 least squares loss), or any joint constraints over the subgradients. Hannah and Dunson [2013] consider an approximation of the convex regression problem which is based on iteratively partitioning the set of observations, and report results for n of the order of $10,000$ in a few minutes. Balázs et al. [2015] propose an aggregate cutting plane based method for solving the full convex regression problem along with an approximate version, and they demonstrate via numerical experiments that their algorithm solves instances with sizes of $n \approx 500$ in a few minutes. However, they do not perform large scale computations and show how their method scales. Regarding statistical results, Han and Wellner [2016] sharply characterize the rate of statistical convergence for the minimax risk.

In the context of linear regression, the problem of sparse regression refers to finding the optimal vector of coefficients $\boldsymbol{\beta} \in \mathbb{R}^d$ which minimizes the sum of squares of the residuals, with the additional restriction that $\boldsymbol{\beta}$ only have at most k (for some positive integer $k < d$) elements different from zero. In high dimensional settings where $d >> n$ such an assumption is valuable for conducting statistical inference, and for settings where $d < n$ sparsity improves interpretability of the model. We explore the notion of sparsity in this setting - we impose the restriction that the union of supports of the subgradients is a set with cardinality at most k. We refer to this problem as the sparse convex regression problem. Sparsity and variable selection for nonparametric regression models is a new and relatively unexplored area. Recently, Xu et al. [2014] develop a method for high dimensional sparse convex regression which solves an approximate problem, with the additional restriction that the target convex function f itself be a sum of univariate convex functions. Additionally, they show that under certain conditions on the samples, this approximation is accurate for the purpose of variable selection.

However, such a cardinality constraint makes the sparse linear regression problem NP-hard [Natarajan, 1995], and has led to this problem being considered as intractable. However, there have been tremendous advances in computing power over the last thirty years - both in hardware and optimization software (see Bixby [2012], Nemhauser [2013] for more details), which can computationally benefit such problems in statistics. Recently, there has been some work that propose using modern Mixed Integer Optimization (MIO) methods along with tools from first order methods in convex optimization for solving classical statistical problems such as best subset selection [Bertsimas et al., 2016] and least quantiles regression [Bertsimas and Mazumder, 2014]. More recently, Bertsimas and Van Parys [2016] propose a reformulation of the sparse regression problem where, they develop a cutting plane algorithm using a duality perspective that solves problems with sizes of n, d in the order of $100,000$s in a few seconds. We explore the use of such techniques while solving the sparse convex regression problem, where we select the best subset of features whose cardinality is bounded by k, and find the optimal convex function on this subset.

1.2. Contributions

In this section, we outline the main contributions of our work.

1. In this paper, we consider the problem of convex regression, and develop a scalable algorithm for obtaining high quality solutions in practical times that compare favorably with other state of the art methods. We show that by using a cutting plane method, the least

squares convex regression problem can be solved for sizes $(n, d) = (10^4, 10)$ in minutes and $(n, d) = (10^5, 10^2)$ in hours. We emphasize that this approach can be also used for ℓ_1 convex regression (where we minimize the ℓ_1 norm of the residuals vector $\mathbf{y} - \boldsymbol{\theta}$) as well with similar scalability results.

2. We propose algorithms which iteratively solve for the best subset of features based on first order and cutting plane methods. To the best of our knowledge, these are the first algorithms for sparse convex regression. We consider two variants of this problem, and develop algorithms for each of them. In one variant, we consider the sparse problem with bounded subgradients, and develop iterative mixed integer optimization based algorithms for solving it. In the second variant, we consider the sparse problem with ridge regularization, and develop a binary cutting plane method for this problem. With the help of computational experiments, we show that our methods are scalable and obtain near exact subset recovery for sizes $(n, d, k) = (10^4, 10^2, 10)$ in minutes, and $(n, d, k) = (10^5, 10^2, 10)$ in hours.

1.3. Structure of the Paper

The structure of the paper is as follows. In Section 2, we present the cutting plane algorithm for solving the least squares convex regression problem and other variants. In Section 3, we define the sparse convex regression problem, and present our solution approach. We illustrate the effectiveness of our approach with computational results and discuss the results in Section 4.

1.4. Notation

For any positive integer n, we use $[n]$ to denote the set of the first n positive integers, that is, $[n] = \{1, \ldots, n\}$. The response vector is an n-dimensional vector of observations, and the covariates are each d-dimensional vectors, i.e., $y \in \mathbb{R}^n, \mathbf{x}_i \in \mathbb{R}^d \; \forall i \in [n]$, where $d \geq 1$. Also, $\|.\|_0$ denotes the ℓ_0 norm, given by the number of nonzero elements in a vector. Finally, $Supp(x)$ denotes the set of indices of the vector x whose corresponding values are non zero.

2. Optimization Algorithm for Convex Regression

In this section, we propose an algorithm to solve the convex regression problem. Additionally, we show that our algorithm can easily accommodate the case with an ℓ_1 objective, as well as other constraints on the subgradients.

2.1. Algorithm

We present a cutting plane based algorithm for solving Problem (4). We now explain the various steps in the algorithm in the following subsections.

Cutting plane algorithms

Cutting plane algorithms are an effective tool for solving large-scale optimization problems where the number of constraints is very high. Before we proceed, we define some terminology that is commonly used in the large-scale optimization literature. In this context, master problem refers to the full formulation (4) with $n(n-1)$ constraints, while the reduced master problem refers to a problem with the same objective and variables, but with only a subset of

the constraints. The main idea behind these methods is to start solving the problem with a few constraints initially - the initial reduced master problem. We then find the violated constraints, and iteratively add them in a delayed manner - at each iteration we solve a reduced master problem (but with progressively more constraints than the initial reduced master problem). Consequently, such methods are also referred to as delayed constraint generation in the large-scale optimization literature [Bertsimas and Tsitsiklis, 1997]. The success of this method depends greatly on the efficiency of finding the violated constraints.

Initial reduced master problem

We start with a fraction of the $n(n-1)$ constraints - an initial reduced master problem. Typically only a small fraction of the $n(n-1)$ constraints will be active at the optimal solution to the full problem, and solving the problem with only these active constraints is clearly equivalent to solving the full problem. However, these active constraints are not known beforehand. A key advantage of starting with a constraint set that is "close" to the active constraint set is that it could substantially reduce the number of cuts added at later iterations, and reduce the net computational burden.

We motivate our algorithm from the solution to the convex regression problem for $d = 1$, where the convexity condition is applied to only the immediate neighboring points. Recall, when $d = 1$, Problem (4) can be computed by solving with only $n-1$ constraints, i.e., by sorting x_i's and considering the adjacent index pairs.

For $d > 1$, given $\mathbf{x}_1, \ldots, \mathbf{x}_n$, we form a spanning path (SP) based on the Euclidean distances between these points. The algorithm works as follows - starting from $\mathbf{x}_{i_1}$ (say $i_1 = 1$), we find the closest point (based on the usual Euclidean distance metric) $\mathbf{x}_{i_2}$ to it, and add it as the next point. We then find the closest point $\mathbf{x}_{i_3}$ to $\mathbf{x}_{i_2}$ over all the points excluding $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$, then we find the closest point $\mathbf{x}_{i_4}$ to $\mathbf{x}_{i_3}$ over all the points excluding $\mathbf{x}_{i_1}$, $\mathbf{x}_{i_2}$ and $\mathbf{x}_{i_3}$, and so on. We utilize the $n-1$ edges in the spanning path among $\mathbf{x}_1, \ldots, \mathbf{x}_n$ as initial constraints. These $n-1$ constraints initially form the reduced master problem:

$$\begin{aligned} \min_{\boldsymbol{\theta}, \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n} \quad & \frac{1}{2}\|\mathbf{y} - \boldsymbol{\theta}\|^2 + \frac{\lambda}{2}\sum_{i=1}^{n} \|\boldsymbol{\xi}_i\|^2 \\ \text{subject to} \quad & \theta_{i_1} + \boldsymbol{\xi}'_{i_1}(\mathbf{x}_{i_2} - \mathbf{x}_{i_1}) \leq \theta_{i_2}, \\ & \theta_{i_2} + \boldsymbol{\xi}'_{i_2}(\mathbf{x}_{i_3} - \mathbf{x}_{i_2}) \leq \theta_{i_3}, \\ & \vdots \\ & \theta_{i_{n-1}} + \boldsymbol{\xi}'_{i_{n-1}}(\mathbf{x}_{i_n} - \mathbf{x}_{i_{n-1}}) \leq \theta_{i_n}, \\ & \|\boldsymbol{\xi}_j\|_\infty \leq M^\star \; \forall 1 \leq j \leq n, \end{aligned} \tag{5}$$

with the solution as $\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\xi}}_1, \ldots, \hat{\boldsymbol{\xi}}_n$. The last constraint bounds the feasible space, with $M^\star$ obtained via solving Problems 14 and 15.

Alternatively, we have also computed the minimum spanning tree (MST) among $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and used the $n-1$ edges of the MST as initial constraints. We have also used randomly chosen pairs of points (Method (R)) as the initial reduced master problem and also selected the closest point for each point $\mathbf{x}_i$ (Method (C)). For $d = 1$, we note that the MST and SP methods coincide. In Section 4, we compare Methods SP, MST, R and C.

Delayed constraint generation

For any given solution to the reduced master problem (5) given by $\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\xi}}_1, \ldots, \hat{\boldsymbol{\xi}}_n$, we need to check if this is a feasible solution for the full problem. If it is indeed feasible, clearly it is also optimal for the full problem. On the other hand, if it is not feasible, we need to find a violated constraint efficiently. This problem of finding a violated constraint is also referred to as the separation problem, as this amounts to finding a hyperplane that separates $\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\xi}}_1, \ldots, \hat{\boldsymbol{\xi}}_n$ from the feasible set (Bertsimas and Tsitsiklis [1997]). Thus, for each i, the i^{th} separation problem is to find the maximal index $j(i)$, where

$$j(i) = \arg \max_{1 \le k \le n} \left\{ \hat{\theta}_i - \hat{\theta}_k + \hat{\boldsymbol{\xi}}_i'(\mathbf{x}_k - \mathbf{x}_i) \right\}, \tag{6}$$

and check if the corresponding largest value is greater than 0.

In practice, we only consider a constraint to be violated if it exceeds a given tolerance Tol. In the case of such a violation, we add the constraint

$$\theta_i + \boldsymbol{\xi}_i'(\mathbf{x}_{j(i)} - \mathbf{x}_i) \le \theta_{j(i)} \tag{7}$$

to the reduced master problem for each i, and re-solve it. Let us denote the index pairs of the violated constraints we add at the k^{th} iteration be given by T_k. Thus, at the k^{th} iteration, the problem we solve is given by

$$\begin{aligned} \min_{\boldsymbol{\theta}, \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n} \quad & \frac{1}{2}\|\mathbf{y} - \boldsymbol{\theta}\|^2 + \frac{\lambda}{2} \sum_{i=1}^{n} \|\boldsymbol{\xi}_i\|^2 \\ \text{subject to} \quad & \theta_i + \boldsymbol{\xi}_i^T(\mathbf{x}_j - \mathbf{x}_i) \le \theta_j, \forall (i,j) \in T_0, \\ & \theta_i + \boldsymbol{\xi}_i^T(\mathbf{x}_j - \mathbf{x}_i) \le \theta_j, \forall (i,j) \in T_1, \\ & \vdots \\ & \theta_i + \boldsymbol{\xi}_i^T(\mathbf{x}_j - \mathbf{x}_i) \le \theta_j, \forall (i,j) \in T_k. \end{aligned} \tag{8}$$

If $\max_{1 \le k \le n} \left\{ \hat{\theta}_i - \hat{\theta}_k + \hat{\boldsymbol{\xi}}_i'(\mathbf{x}_k - \mathbf{x}_i) \right\} \le Tol \ \forall i \in [n]$, then the current solution is in fact optimal for the full problem (4) with $n(n-1)$ constraints, and the method terminates. The complete algorithm is as follows:

Algorithm 1 Cutting plane algorithm for Problem (4)

Input: Data $(y_i, \mathbf{x}_i)$, $i = 1, \ldots, n$, tolerance $Tol > 0$.
Output: An optimal solution $(\boldsymbol{\theta}^*, \boldsymbol{\xi}_1^*, \ldots, \boldsymbol{\xi}_n^*)$ to Problem (4).

1: Solve the reduced master problem, i.e., Problem (8) with $k = 0$.
2: Set $success = 0$.
3: **while** $success == 0$ **do**
4: **for** $1 \le i \le n$ **do**
5: For this i, solve the separation problem (6) to find a $j(i)$.
6: Add the corresponding violated constraint (Eq. (7)), to the reduced master problem.
7: **end for**
8: If there is no violated constraint within the tolerance Tol, set $success \leftarrow 1$.
9: Else, re-solve Problem (8) with new constraint set T_{k+1}, consisting of additional constraint(s) added from Steps $4 - 7$.
10: $k \leftarrow k + 1$
11: **end while**

We also note that this cutting plane algorithm, by successively adding violated constraints to the reduced master problem, is guaranteed to converge to an optimal solution in a finite number of steps [Kelley, 1960].

Theorem 1. *The cutting plane Algorithm 1 converges to an optimal solution of Problem (4) in a finite number of iterations.*

2.2. ℓ_1 convex regression

Consider the problem of ℓ_1 convex regression, given by,

$$\min_{f \in \mathcal{C}} \quad \sum_{i=1}^{n} |y_i - f(x_i)| \tag{9}$$

where, as before, $\mathcal{C}$ is the space of convex functions on $\mathbb{R}^d$. Such a variant is along the lines of linear regression with an ℓ_1 loss, rather than the usual least squares loss. Problem (9) can be written as an equivalent finite dimensional linear optimization problem (10), using additional auxiliary variables $\mathbf{z} \in \mathbb{R}^n_+$ as follows.

$$\begin{aligned} \min_{\boldsymbol{\theta}, \{\boldsymbol{\xi}_i\}_{i=1}^n, \mathbf{z}} \quad & \sum_{i=1}^{n} z_i \\ \text{subject to} \quad & z_i \geq y_i - \theta_i \quad \forall i, \\ & z_i \geq -(y_i - \theta_i) \quad \forall i, \\ & \theta_i + \boldsymbol{\xi}_i^T (\mathbf{x}_j - \mathbf{x}_i) \leq \theta_j \quad \forall i, j, \\ & \boldsymbol{\theta}, \mathbf{z} \in \mathbb{R}^n, \\ & \boldsymbol{\xi}_i \in \mathbb{R}^d \; \forall i \in [n]. \end{aligned} \tag{10}$$

We utilize the dual simplex algorithm, as when we introduce a new cut the optimality conditions are satisfied, while the previous solution may be infeasible. As we illustrate in Section 4, this method is fast in practice and scales well.

2.3. Extensions

Algorithm 1 can be extended to accommodate the following additional requirements on $f(\cdot)$:

a) The function f is coordinate-wise monotone, i.e., its subgradients $\boldsymbol{\xi}_i$ are either $\boldsymbol{\xi}_i \geq \mathbf{0}$ or $\boldsymbol{\xi}_i \leq \mathbf{0}$ (non-decreasing or non-increasing respectively) for all i.

b) The subgradients $\boldsymbol{\xi}_i$ are bounded, i.e., $\|\boldsymbol{\xi}_i\|_p \leq L \; \forall i$ for some L and ℓ_p norm $\|.\|_p$. The usual cases of $p \in \{1, 2, \infty\}$ result in conic optimization problems and can be handled by this approach. Such constraints could be added as a part of the reduced master problem all at once, or in a delayed manner as and when they are violated.

3. Sparse Convex Regression

In this section, we consider the problem of sparse convex regression, in which the union of supports of the subgradients of f in each point x is a set whose cardinality is bounded by k.

We formulate this as the following optimization problem over sets,

$$\begin{aligned} \min_{\boldsymbol{\theta},\{\boldsymbol{\xi}_i\}_{i=1}^n,S} \quad & \frac{1}{2}\sum_{i=1}^{n}(y_i-\theta_i)^2+\frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{\xi}_i\|_2^2 \\ \text{subject to} \quad & \theta_i+\boldsymbol{\xi}_i^T(\mathbf{x}_j-\mathbf{x}_i)\leq\theta_j\ \forall i,j, \\ & Supp(\boldsymbol{\xi}_i)\subseteq S\ \forall i, \\ & \boldsymbol{\theta}\in\mathbb{R}^n, \\ & \boldsymbol{\xi}_i\in\mathbb{R}^d\ \forall i, \\ & |S|\leq k, S\subseteq\{1,\ldots,d\}\,. \end{aligned} \tag{11}$$

3.1. Primal approach

In this section, we present a primal-based approach of solving for the optimal subset of features for the convex regression problem. Consider the following mixed integer (binary) quadratic optimization (MIQO) problem

$$\begin{aligned} \min_{\boldsymbol{\theta},\mathbf{z},\{\boldsymbol{\xi}_i\}_{i=1}^n} \quad & \frac{1}{2}\sum_{i=1}^{n}(y_i-\theta_i)^2+\frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{\xi}_i\|_2^2 \\ \text{subject to} \quad & \theta_i+\boldsymbol{\xi}_i^T(\mathbf{x}_j-\mathbf{x}_i)\leq\theta_j\ \forall 1\leq i,j\leq n, \\ & |(\xi_i)_j|\leq Mz_j\ \forall i\in[n], j\in[d], \\ & \sum_{j=1}^{n}z_j\leq k, \\ & \mathbf{z}\in\{0,1\}^d\,, \\ & \boldsymbol{\theta}\in\mathbb{R}^n, \\ & \boldsymbol{\xi}_i\in\mathbb{R}^d\ \forall i\in[n]. \end{aligned} \tag{12}$$

for some positive constant M.

To solve this problem, we first develop heuristics based on convex optimization which generate solutions and are fast in practice. We solve a reduced MIQO problem (using a commercial mixed integer optimization solver (Gurobi)) to generate lower bounds, which provide a guarantee on the quality of this solution. Bertsimas et al. [2016] used commercial state of the art MIO solvers to solve the sparse linear regression problem with considerable success. We present the details on this algorithm in the following section.

Algorithmic approach

In this section, we present an algorithm to solve Problem (12). To summarize, our solution approach involves generating lower bounds by solving the reduced MIQO problem, and improving this bound at each successive iteration. We use heuristics in order to find feasible solutions fast, and generate lower bounds in order to determine the quality of the proposed solution, or potentially improve it further. We elaborate in more detail on the heuristics in Section 3.3. In order to determine the quality of our solution (in terms of optimality gap), we generate lower bounds. For this, we solve the full sparse problem as an MIQO problem, but with only the initial reduced set of constraints to start. Whenever possible, we warm-start this problem with a feasible solution obtained via heuristics, which we briefly discuss in Section 3.3. We then iteratively add the violated constraints to Problem (12) to tighten the bounds, similar in spirit

to the cutting plane approach. For the upper bound, we solve the full convex regression problem on this restricted support. To be precise, this problem is given by

$$\begin{aligned} \min_{\boldsymbol{\theta},\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_n} \quad & \frac{1}{2}\sum_{i=1}^{n}(y_i-\theta_i)^2+\frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{\xi}_i\|_2^2 \\ \text{subject to} \quad & \theta_i+\boldsymbol{\xi}_i'\big((x_j)_S-(x_i)_S\big)\le\theta_j\ \forall i,j, \\ & \|\boldsymbol{\xi}_i\|_\infty\le M\ \forall i, \\ & \boldsymbol{\xi}\in\mathbb{R}^k\ \forall i,\boldsymbol{\theta}\in\mathbb{R}^n. \end{aligned} \tag{13}$$

where S is the support set obtained from the MIO solution, and $\mathbf{v}_S$ is the vector $\mathbf{v}$ restricted to the set S. The overall primal algorithm is as follows:

Algorithm 2 Primal approach.

Input: Initial constraints ($\mathcal{C}^{(0)}$, a subset of the $n(n-1)$ constraints), tolerance $\epsilon>0$, a positive integer T.

Output: A sparse optimal solution to problem (12).

1: Initialize Problem (12) with the initial constraints $\mathcal{C}^0$.
2: Use the initialization heuristic (Section 3.3) to generate an initial solution $S^{(0)}$.
3: Set $t\leftarrow 1$.
4: **while** $t\le T$ AND gap $>\epsilon$ **do**
5: Solve problem (12), with reduced constraint set $\mathcal{C}^{(t)}$, to obtain support set $S^{(t)}$, possibly utilizing $S^{(t-1)}$ as a warm-start.
6: Set LB (Lower bound) to the optimal objective of problem (12).
7: With the output support, solve Problem (13) on the support $S^{(t)}$.
8: Update UB (Upper bound) to be the optimal objective.
9: Update gap $\leftarrow \frac{UB-LB}{LB}$.
10: Add (at most) n violated constraints (one for each $1\le i\le n$), which forms $\mathcal{C}^{(t+1)}$ by this solution to the lower bound MIQO problem (12) to obtain the support $S^{(t+1)}$.
11: Warm start it with the solution to the same lower bound problem constraints on this restricted support set $S^{(t+1)}$ by solving Problem (13).
12: $t\leftarrow t+1$
13: **end while**

Computing the bound M

In this section, we describe how we compute bounds on the subgradient values. For some initial feasible solution θ^0 and $\boldsymbol{\xi}_1^0,\ldots,\boldsymbol{\xi}_n^0$ for Problem (11), we solve the following problems, for each $1\le t\le n, 1\le u\le d$:

$$\begin{aligned} \min_{\boldsymbol{\theta},\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_n} \quad & \boldsymbol{\xi}_{t,u} \\ \text{subject to} \quad & \frac{1}{2}\|\mathbf{y}-\boldsymbol{\theta}\|^2+\frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{\xi}_i\|_2^2\le\frac{1}{2}\|\mathbf{y}-\boldsymbol{\theta}^0\|^2+\frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{\xi}_i^0\|_2^2, \\ & \theta_i+\boldsymbol{\xi}_i'(\mathbf{x}_j-\mathbf{x}_i)\le\theta_j\ \forall i,j, \end{aligned} \tag{14}$$

and

$$\begin{aligned} \max_{\boldsymbol{\theta},\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_n} \quad & \boldsymbol{\xi}_{t,u} \\ \text{subject to} \quad & \frac{1}{2}\|\mathbf{y}-\boldsymbol{\theta}\|^2 + \frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{\xi}_i\|_2^2 \leq \frac{1}{2}\|\mathbf{y}-\boldsymbol{\theta}^0\|^2 + \frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{\xi}_i^0\|_2^2, \\ & \theta_i + \boldsymbol{\xi}_i'(\mathbf{x}_j - \mathbf{x}_i) \leq \theta_j \ \forall i,j. \end{aligned} \tag{15}$$

We note that the feasible regions of Problems (14) and (15) are bounded, and hence the optimal objective values for both these problems is guaranteed to be finite.

Let M^* be the maximum of the optimal solution absolute values of (14) and (15) over all $1 \leq t \leq n$ and $1 \leq u \leq d$. An optimal solution of (11) is clearly feasible to both (14) and (15). Therefore, using M^* in the formulation (12) does not exclude optimal solutions to (11) and therefore the optimal solution values of Problems (11) and (12) are equal.

3.2. Dual approach

In this section, we adapt the approach proposed by Bertsimas and Van Parys [2016] for sparse linear regression to this convex regression setting. We solve the following regularized problem, for a given $\lambda > 0$,

$$\begin{aligned} \min_{\boldsymbol{\theta},\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_n} \quad & \frac{1}{2}\|\mathbf{y}-\boldsymbol{\theta}\|^2 + \frac{\lambda}{2}\sum_{i=1}^{n}\|\boldsymbol{\xi}_i\|^2 \\ \text{subject to} \quad & \theta_i + \boldsymbol{\xi}_i'(\mathbf{x}_j - \mathbf{x}_i) \leq \theta_j \ \forall i,j, \\ & Supp(\boldsymbol{\xi}_i) \subseteq S \ \forall i, \\ & \boldsymbol{\theta} \in \mathbb{R}^n, \\ & \boldsymbol{\xi}_i \in \mathbb{R}^d \ \forall i, \\ & |S| \leq k, S \subseteq \{1,\ldots,d\}. \end{aligned} \tag{16}$$

Before we proceed, we introduce some notation. S_k^d denotes the set of d dimensional binary vectors with at most k non-zero components, i.e.,

$$S_k^d = \{z \in \{0,1\}^d : \sum_{i=1}^{d} z_i \leq k\}.$$

We next present the following result that transforms this problem to a binary optimization problem with a convex objective function.

Theorem 2. *Problem (16) is equivalent to solving the following binary optimization problem with convex objective, given by*

$$\min_{\mathbf{z}\in S_k^d} \ g(\mathbf{z}), \tag{17}$$

where

$$g(\mathbf{z}) = \max_{\mu \geq 0} \quad -\frac{1}{2}\sum_{i=1}^{n}\left(y_i + \sum_{j=1}^{n}\mu_{ji} - \sum_{j=1}^{n}\mu_{ij}\right)^2 - \frac{1}{2\lambda}\sum_{i=1}^{n}\sum_{p=1}^{d} z_p \left(\sum_{j=1}^{n}\mu_{ij}(\mathbf{x}_j - \mathbf{x}_i)\right)_p^2, \tag{18}$$

and a subgradient of g is given by the vector with p^{th} element given by

$$(\partial g(\mathbf{z}))_p = -\frac{1}{2\lambda}\sum_{i=1}^{n}\left(\sum_{j=1}^{n}\hat{\mu}_{ij}(x_{ip} - x_{jp})\right)^2, \tag{19}$$

where $\hat{\mu}$ is an optimal solution to the concave maximization problem given in Eq. (18).

Proof. Using binary variables $\mathbf{z} \in \{0,1\}^d$ to denote the support set $\big(z_j = 0 \iff (\boldsymbol{\xi}_i)_j = 0 \; \forall i \in [n]\big)$, we write Problem (16) as

$$\begin{aligned} \min_{\mathbf{z}\in S_k^d, \mathbf{Z}=diag(\mathbf{z})} \quad & \min_{\boldsymbol{\theta},\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_n} && \frac{1}{2}\|\mathbf{y}-\boldsymbol{\theta}\|^2 + \frac{\lambda}{2}\sum_{i=1}^n \|\boldsymbol{\xi}_i\|^2 \\ & && \text{subject to} \quad \theta_i + \boldsymbol{\xi}_i'\mathbf{Z}(\mathbf{x}_j - \mathbf{x}_i) \leq \theta_j \; \forall i,j. \end{aligned} \tag{20}$$

We take the dual of the inner convex optimization problem, which is given by

$$\max_{\mu\geq 0} \quad -\frac{1}{2}\sum_{i=1}^n \left(y_i + \sum_j \mu_{ji} - \sum_j \mu_{ij}\right)^2 - \frac{1}{2\lambda}\sum_{i=1}^n \left\| \sum_j \mu_{ij}\mathbf{Z}(\mathbf{x}_i - \mathbf{x}_j)\right\|^2.$$

For brevity, let $\mathbf{v}_i = \sum_j \mu_{ij}(\mathbf{x}_i - \mathbf{x}_j)$. Note that $\mathbf{Z}'\mathbf{Z} = \mathbf{Z}^2 = \mathbf{Z}$, and thus we get

$$\max_{\mu\geq 0} \quad -\frac{1}{2}\sum_{i=1}^n \left(y_i + \sum_j \mu_{ji} - \sum_j \mu_{ij}\right)^2 - \frac{1}{2\lambda}\sum_{i=1}^n\sum_{p=1}^d z_p \left(\sum_{j=1}^n \mu_{ij}(\mathbf{x}_j - \mathbf{x}_i)\right)_p^2, \tag{21}$$

and thus, the result follows. □

From Theorem 2, as g is convex in $\mathbf{z}$, we use $\hat{\mu}$ to compute the subgradient of g which we use to solve the outer binary minimization problem using cutting planes. This is equivalent to approximating the convex function g by a piecewise linear function of its lower tangents, while improving the outer approximation by adding a new tangent at each iteration. To be precise, we solve the outer problem as

$$\begin{aligned} \min_{\mathbf{z}\in\{0,1\}^d} \quad & \max_{i=1,\ldots,m} \left\{ g(\mathbf{z}^i) + \partial g(\mathbf{z}^{(i)})'(\mathbf{z} - \mathbf{z}^i)\right\} \\ \text{subject to} \quad & \sum_{i=1}^d z_i \leq k, \end{aligned} \tag{22}$$

or equivalently in epigraph form,

$$\begin{aligned} \min_{\mathbf{z}\in\{0,1\}^d,\gamma} \quad & \gamma \\ \text{subject to} \quad & g(\mathbf{z}^i) + \partial g(\mathbf{z}^{(i)})'(\mathbf{z} - \mathbf{z}^i) \leq \gamma \; \forall 1 \leq i \leq m \\ \text{subject to} \quad & \sum_{i=1}^d z_i \leq k, \end{aligned} \tag{23}$$

where m is the number of cuts added.

While solving Problem (22) we use dynamic constraint generation, or lazy callbacks, which enables the solver to avoid building multiple branch and bound trees each time a new constraint is added to the problem. This leads to only one branch and bound tree being built. Typically, lazy constraints are used when the full set of constraints is too large to enumerate explicitly. Under this scheme, cuts are added to the model whenever a binary feasible solution is found.

As mentioned in Bertsimas and Van Parys [2016] for the sparse linear regression case, the linear relaxation of problem (17) provides strong warm starts to problem (16). This motivates the following corollary.

Corollary 2.1. *The linear relaxation of problem (17) is given by the following convex optimization problem with semi-infinite constraints:*

$$\begin{aligned} \min_{\mu \geq 0, \gamma} \quad & \frac{1}{2} \sum_{i=1}^{n} \left(y_i + \sum_j \mu_{ji} - \sum_j \mu_{ij} \right)^2 + \gamma \\ \text{subject to} \quad & \gamma \geq \frac{1}{2\lambda} \sum_{p=1}^{d} z_p \left\{ \sum_{i=1}^{n} \left(\sum_j \mu_{ij} (x_{ip} - x_{jp}) \right)^2 \right\} \quad \forall z \in \Delta_{k,d} \end{aligned} \tag{24}$$

where $\Delta_{k,d} = \left\{ z \in \mathbb{R}^d : 0 \leq z \leq 1, \sum_{i=1}^{d} z_i \leq k \right\}$.

We solve the relaxation to generate warm-starts for the original binary optimization problem (22). Once again, we use cutting planes to solve this problem. At the optimal solution, the support set would be the corresponding indices of the k largest values of the vector $\mathbf{v}$, with p^{th} element given by

$$v_p = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \mu_{ij} (x_{ip} - x_{jp}) \right)^2. \tag{25}$$

In practice, we have observed that this method does provide good quality warm-starts.

Before we elaborate further, we introduce some terminology. Here, the outer problem refers to the binary minimization problem (22). As we have noted in the statement of the theorem 2, evaluating the function g requires us to solve an optimization problem (18), which we shall henceforth refer to as the inner problem.

Column generation methods for the inner problem

An issue with the above approach is that for the inner problem, the number of variables μ is too large, i.e., $O(n^2)$, and is thus not practical for larger n. Hence, we propose a column generation approach for solving the inner problem (18). We start with a subset of all the $n(n-1)$ variables μ (with the rest set to zero), and add corresponding variables as we go along. From the KKT conditions, for a given dual optimal solution $\hat{\mu}$ we recover the primal variables as:

$$\begin{aligned} \theta_i &= y_i + \sum_{j=1}^{n} \hat{\mu}_{ji} - \sum_{j=1}^{n} \hat{\mu}_{ij} \ \forall i, \\ \boldsymbol{\xi}_i &= \frac{1}{\lambda} \sum_{j=1}^{n} \hat{\mu}_{ij} \mathbf{Z} (\mathbf{x}_i - \mathbf{x}_j) \ \forall i. \end{aligned} \tag{26}$$

We add the violated constraint if

$$\theta_i + \boldsymbol{\xi}_i' (\mathbf{x}_j - \mathbf{x}_i) - \theta_j \leq 0. \tag{27}$$

If not, then for any i, we find the j^* such that

$$j^* = \arg \max_{1 \leq j \leq n} \left\{ \theta_i - \theta_j + \boldsymbol{\xi}_i' (\mathbf{x}_j - \mathbf{x}_i) \right\}, \tag{28}$$

and add the variable μ_{ij^*} to the set of active variables, and re-solve problem (18).

In practice, problem (18), while having relatively simple nonnegative constraints, has a dense quadratic objective, which often results in larger solve times. Instead, we solve its dual, which is

the inner minimization problem in Eq. (20). We use Algorithm 1 to solve the inner minimization problem in (20) and calculate the variables $\boldsymbol{\theta}$ and $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n$, as well as the dual variables μ_{ij} corrsponding to the constraints in Eq. (20). Given these values, and the expression in Eq. (19), we compute the subgradient of g at this value of z, and add the corresponding constraint to the outer binary optimization problem in the case of a violation. This dual approach differs from the method in Bertsimas et al. [2016], which is a primal method. In Section 4, we observe that this dual approach has a significant edge over the primal one.

We now present the complete algorithm for the dual approach:

Algorithm 3 Cutting plane based algorithm for the dual approach

Input: $\lambda > 0$, tolerance $\epsilon > 0$.
Output: Optimal support z^*.

Start with $\gamma^0 = 0$ and some feasible z^0.
$t \leftarrow 0$.
while $\gamma^t < g(z^t) + \epsilon$ **do**
    Compute a subgradient value of g at z^t, using Theorem 2.
    Add the constraint $g(z^t) + \partial g(z^t)'(z - z^t) \leq \gamma$.
    Re-solve the outer problem (23), with solution given by (z^{t+1}, γ^{t+1}).
    $t \leftarrow t + 1$.
end while

3.3. Initialization heuristics

In this section, we briefly describe a thresholding based heuristic for the sparse convex regression problem. This method provides an alternative approach of generating warm starts to solving the relaxation problem (24). For the sake of brevity, we do not include the ridge regularization term, but these methods can be easily adapted to include it as well.

$$\begin{aligned} \min_{\boldsymbol{\theta}, \boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{y} - \boldsymbol{\theta}\|^2 \\ \text{subject to} \quad & \bar{A}\boldsymbol{\theta} + \sum_{i=1}^{n} \bar{B}_i \boldsymbol{\xi}_i \leq \mathbf{0}, \\ & Supp(\boldsymbol{\xi}_i) \subseteq S \; \forall i, \\ & \boldsymbol{\theta} \in \mathbb{R}^n, \\ & \boldsymbol{\xi}_i \in \mathbb{R}^d \; \forall i, \\ & |S| \leq k, S \subseteq \{1, \ldots, d\}. \end{aligned} \tag{29}$$

where $\bar{A}, \bar{B}_i$ are the full matrices representing the total $n(n-1)$ constraints. Typically at any feasible solution, only a few of the constraints will be active. Let the indices of the binding constraints be described in T, and the sub matrices be given by $A_T, B_{T,i}$. Thus, the problem

can be written as

$$\begin{aligned} \min_{\boldsymbol{\theta},\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{y}-\boldsymbol{\theta}\|^2 \\ \text{subject to} \quad & \bar{A}_T\boldsymbol{\theta} + \sum_{i=1}^{n} \bar{B}_{T,i}\boldsymbol{\xi}_i \leq \mathbf{0}, \\ & Supp(\boldsymbol{\xi}_i) \subseteq S \; \forall i, \\ & \boldsymbol{\theta} \in \mathbb{R}^n, \\ & \boldsymbol{\xi}_i \in \mathbb{R}^d \; \forall i, \\ & |S| \leq k, S \subseteq \{1,\ldots,d\}. \end{aligned} \tag{30}$$

Dualizing the linear inequality constraints, the objective is given by

$$f(\boldsymbol{\theta},\boldsymbol{\xi}) = \max_{\lambda \geq \mathbf{0}} \quad \frac{1}{2}\|\mathbf{y}-\boldsymbol{\theta}\|^2 + \lambda'(\bar{A}_T\boldsymbol{\theta} + \sum_{i=1}^{n} \bar{B}_{T,i}\boldsymbol{\xi}_i). \tag{31}$$

We smoothen the objective function by subtracting a strongly convex term ($\frac{\tau}{2}\|\lambda\|^2$) for some fixed scalar $\tau > 0$. Note that we need to efficiently compute this function f for different values of $\theta, \boldsymbol{\xi}$. The smooth convex objective is now

$$f_\tau(\boldsymbol{\theta},\boldsymbol{\xi}) = \max_{\lambda \geq \mathbf{0}} \quad \frac{1}{2}\|\mathbf{y}-\boldsymbol{\theta}\|^2 + \lambda'(\bar{A}_T\boldsymbol{\theta} + \sum_{i=1}^{n} \bar{B}_{T,i}\boldsymbol{\xi}_i) - \frac{\tau}{2}\|\lambda\|_2^2. \tag{32}$$

This function f_τ is Lipschitz continuous with parameter ℓ, where $\ell = \frac{\lambda_{\text{MAX}}(\mathcal{M}'\mathcal{M})}{\tau}$ [Nesterov, 2005]. The matrix $\mathcal{M} \in \mathbb{R}^{(m)\times(n+nd)}$, where m is the number of rows of $\bar{A}_T$ (the number of active equality constraints), and is given by

$$\mathcal{M} = \begin{bmatrix} \bar{A}_T & \bar{B}_{T,1} & \ldots & \bar{B}_{T,n}. \end{bmatrix}$$

Now, the optimal λ_τ^* can be computed by

$$\lambda_\tau^* = \frac{1}{\tau}(\bar{A}_T\boldsymbol{\theta} + \sum_{i=1}^{n} \bar{B}_{T,i}\boldsymbol{\xi}_i)_+. \tag{33}$$

We then apply an upper quadratic approximation to the above function, followed by an iterative thresholding procedure to the above function $f_\tau(\boldsymbol{\theta}, \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n)$, while sequentially reducing the value of τ. The complete details of this algorithm can be found in the Appendix.

4. Computational Experiments

Our objective in this section is

1. To understand the scalability and run times of Algorithm 1 for convex regression for synthetic and real data.
2. To compare the performance of Algorithm 1 to other state of the art methods.
3. To understand the scalability and run times of Algorithms 2 and 3 for sparse convex regression. Given that there are no competing approaches for this problem to the best of our knowledge, we do not include any comparisons.

The structure of this section is as follows. In Section 4.1, we discuss the data generation mechanism, and compare various initialization schemes for Algorithm 1 in Section 4.2. We then examine its run times on synthetic data in Section 4.3, and infeasibility of the solution at each iteration of Algorithm 1 in Section 4.4. Next, we compare it with other approaches in Section 4.5, and discuss the run times of Algorithm 1 applied to the convex regression problem with an ℓ_1 loss in Section 4.6. We then present the run times and infeasibility of Algorithm 1 on real data in Section 4.7. Next, we consider the sparse convex regression problem in Section 4.8, where we present the run times of Algorithm 2 (primal approach) and Algorithm 3 (dual approach) for various sizes. Additionally, we present the accuracy and run times of Algorithm 3 as a function of various parameters such as k, d, ρ, SNR, and present the false positive rates of both the algorithms as well in this section. We conclude by discussing our findings from these experiments in Section 4.9.

In all the experiments that follow, we use Gurobi 6.5.2 Gurobi as the optimization solver, within the Julia programming language [Bezanson et al., 2014] using the JuMP modeling framework [Dunning et al., 2015]. All computations were performed on nodes of the Engaging cluster, which is a collaboration between the Massachusetts Green High Performance Computing Center (MGHPCC) and several of Boston's leading universities. Each compute node of the cluster had two 8-core, 2GHz Intel Xeon E2650 processors, 64GB of memory and 3.5TB of local disk. We provide the code and test instances data at `https://github.com/nmundru/scr`. In the online supplement, we briefly describe the code for generating the random data, and implementing the algorithms in this paper.

4.1. Synthetic Data

In this section, we generate $\mathbf{X}$ data from a standard Gaussian distribution, and use the convex function $\Phi(\mathbf{x}) = \|\mathbf{x}\|_2^2$, where $y_i = \Phi(\mathbf{x}_i) + \epsilon_i,\ 1 \leq i \leq n$. The errors ϵ_i are assumed to be independent and identically distributed Gaussian, i.e., $N(0, \sigma^2)$, for $i = 1, \ldots, n$. We scale the data appropriately so that the Signal to Noise ratio (SNR) is 3, i.e., $\frac{\mathrm{Var}(\mu)}{\mathrm{Var}(\epsilon)} = 3$. Finally, before feeding the data into the algorithm, we mean-center and normalize the features and response vectors to have unit ℓ_2 norm.

We report the number of blocks of cuts (iterations) added till the end, along with another metric called primal infeasibility [Mazumder et al., 2015],

$$\text{Primal infeasibility} = \frac{1}{n}\|\mathbf{V}\|_F \tag{34}$$

where the matrix $\mathbf{V}$ has entries given by $\mathbf{V}_{i,j} = (\hat{\theta}_i + \hat{\boldsymbol{\xi}}_i'(\mathbf{x}_j - \mathbf{x}_i) - \hat{\theta}_j)_+,\ \forall 1 \leq i, j \leq n$, where $z_+ = \max\{z, 0\}$. $\mathbf{V}_{i,j}$ indicates the magnitude of violation of that constraint, and a value of 0 indicates no violation. Note that $\|\cdot\|_F$ denotes the usual Frobenius norm, where

$$\|V\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} V_{i,j}^2. \tag{35}$$

Finally, *Tol* is the threshold above which we report the constraint to be violated.

4.2. Comparison of initialization methods for the reduced master problem

In this section, we apply Algorithm 1 to the least squares convex regression problem (4). We compare the run times of different ways of forming the reduced master problem for Problem (4): We used methods MST, SP, C and R. MST refers to the Euclidean minimum spanning tree formed on the set of points $\mathbf{x}_1, \ldots, \mathbf{x}_n$. SP refers to the spanning path approach described

in Section 2.1. C refers to finding the point closest to each point, and adding that pair in that order. For example, if $\mathbf{x}_j$ is closest to $\mathbf{x}_i$, we add the constraint

$$\theta_i + \boldsymbol{\xi}_i'(\mathbf{x}_j - \mathbf{x}_i) \leq \theta_j.$$

Finally, R refers to finding a point randomly sampled from the remaining $n-1$ for each $\mathbf{x}_i$, and adding the resulting n constraints. The last four methods 2-MST, 2-SP, 2-C, and 2-R denote two sided constraints, i.e., for each pair $(\mathbf{x}_i, \mathbf{x}_j)$, we add both the constraints

$$\theta_i + \boldsymbol{\xi}_i'(\mathbf{x}_j - \mathbf{x}_i) \leq \theta_j$$

and

$$\theta_j + \boldsymbol{\xi}_j'(\mathbf{x}_i - \mathbf{x}_j) \leq \theta_i.$$

The term Tol in Table 1 refers to the tolerance to which each of the $n(n-1)$ constraints is satisfied while terminating Algorithm 1. The sizes for all the instances are set to $(n, d) = (10^4, 10)$, and we use the least squares objective function $\|\mathbf{x}\|_2^2$ without the ridge regularization term of the subgradients. All entries in the table are averaged over the same twenty instances. The numbers in parenthesis indicate the standard deviation.

Method	Tol	Cuts added (Blocks)	Primal Infeasibility	Run time (seconds)
MST	0.10	26 (2)	0.0104 (0.0001)	94.44 (20.826)
SP	0.10	9 (6)	0.0112 (0.0002)	21.36 (12.820)
C	0.10	25 (2)	0.0106 (0.0002)	55.93 (4.539)
R	0.10	6 (3)	0.0106 (0.0002)	15.39 (6.125)
2-MST	0.10	26 (2)	0.0098 (0.0002)	343.53 (62.363)
2-SP	0.10	15 (5)	0.0091 (0.0001)	35.66 (11.276)
2-C	0.10	26 (2)	0.0104 (0.0002)	131.09 (18.933)
2-R	0.10	21 (2)	0.0088 (0.0001)	46.47 (5.273)
MST	0.05	29 (2)	0.0073 (0.0002)	221.21 (40.035)
SP	0.05	25 (2)	0.0078 (0.0002)	56.75 (7.027)
C	0.05	30 (2)	0.0061 (0.0001)	117.07 (19.565)
R	0.05	26 (3)	0.0074 (0.0001)	57.95 (8.107)
2-MST	0.05	31 (3)	0.0055 (0.0001)	1448.46 (313.576)
2-SP	0.05	25 (2)	0.0068 (0.0001)	58.30 (7.041)
2-C	0.05	31 (2)	0.0059 (0.0001)	567.26 (121.269)
2-R	0.05	26 (2)	0.0064 (0.0001)	61.51 (6.861)

Table 1: The effect of the initialization method for $(n, d) = (10^4, 10)$ in the ℓ_2 convex regression for tolerances $Tol = 0.1$ and 0.05.

The results of Table 1 suggest that starting from a "good" initial reduced master problem can substantially impact the total run time of Algorithm 1. Both the Spanning path (SP) and Random (R) methods outperform the other methods. SP and R perform similarly, with the one-sided being marginally better than the two-sided constraints. We chose R in all of our further experiments.

4.3. Run times of ℓ_2 convex regression

In this section, we report how Algorithm 1 scales as n, d increase for Problem (4), different tolerances and least square objective. Table 2 presents the results obtained for a tolerance of 0.1, while Table 3 shows the results for a tolerance of 0.05.

n	d	Cuts (Blocks)	Infeasibility	Run time
10^3	10^1	24 (2)	0.0147 (0.0016)	2.4s (1.5s)
10^4	10^1	8 (5)	0.0106 (0.0002)	16.5s (8.7s)
10^4	10^2	14 (3)	0.0107 (0.0003)	169.2s (35.5s)
10^4	10^3	22 (6)	0.0107 (0.0002)	1.5h (0.4h)
10^5	10^1	5 (4)	0.0054 (0.0001)	1156.9s (859.4s)
10^5	10^2	5 (1)	0.0056 (0.0001)	3.8h (0.4h)
10^5	5×10^2	6 (1)	0.0056 (0.0001)	19.1h (3.0h)
5×10^5	10^1	5 (4)	0.0034 (0.0000)	20.2h (7.2h)

Table 2: Run times for $Tol = 0.1$ and ℓ_2 convex regression.

n	d	Cuts (Blocks)	Infeasibility	Run time
10^3	10^1	36 (4)	0.0026 (0.0004)	58.0s (25.6s)
10^4	10^1	25 (3)	0.0074 (0.0001)	57.0s (8.4s)
10^4	10^2	110 (3)	0.0065 (0.0003)	1369.3s (91.7s)
10^5	10^1	11 (6)	0.0039 (0.0001)	1.0h (0.4h)
10^5	10^2	11 (1)	0.0040 (0.0000)	6.8h (0.7h)

Table 3: Run times for $Tol = 0.05$ and ℓ_2 convex regression.

We make the following observations:

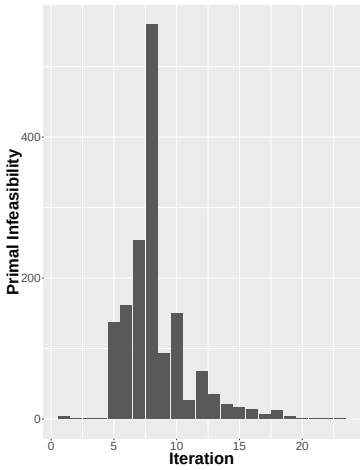
- As the number of dimensions increases, the problem becomes harder to solve as each added constraint becomes more dense. This is reflected in both Tables 2 and 3.
- The largest instances of $(10^5, 500)$ and $(5 \times 10^5, 10)$ took almost a day on average to solve to the required tolerance. While we tried solving them with $Tol = 0.05$, the run time took more than 24 hours, after which we terminated them. For such problems, the interior point solvers, even if they solve the initial reduced master problem, stall at subsequent iterations when the quadratic problem has close to a million constraints.
- When tolerance is reduced to 0.05, the run times of $(10^4, 10^2)$ increases from a 2.5 minutes to 23 minutes with the average number of iterations increasing by a factor of eight.
- To further aid in interpreting the results, we performed a linear regression of the run times versus n, d and Tol. Our results indicate that a linear relationship between these variables has an R^2 of 0.96, which indicates a good fit. Regressing the logarithms of times versus the logarithm of n and d yields that the run time depends on $n^{1.25}$ and $d^{1.05}$, which also resulted in an R^2 value of 0.96.

4.4. Infeasibility as a function of iterations

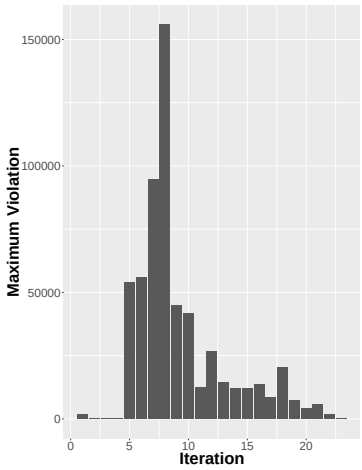
In this section, we aim to understand how the primal infeasibility changes as a function of the iterations for different values of tolerance. In addition to primal infeasibility defined in (34), we report the maximum violation defined as

$$\max_{i \in [n], j \in [n]} \left\{ \theta_i - \theta_j + \boldsymbol{\xi}_i'(\mathbf{x}_j - \mathbf{x}_i) \right\}, \tag{36}$$

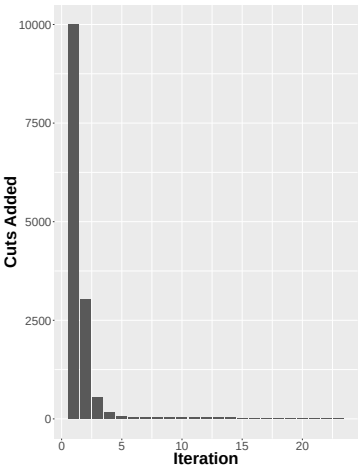
as well as the constraints added at each iteration. We present two instances with $(n, d) = (10^4, 10)$ - with tolerance set to 0.1 and 0.05 respectively, and illustrate the progress of the algorithm - constraints added at each iteration, primal infeasibility and the maximum violation at the end of each iteration.



(a) Primal infeasibility defined in (34) as a function of the number of iterations.

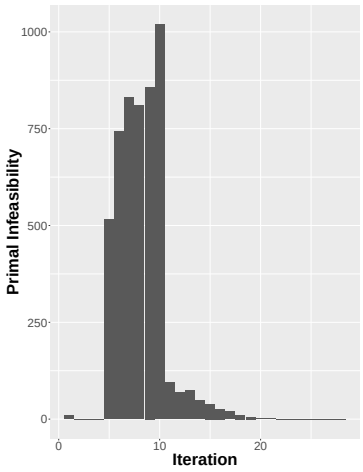


(b) Maximum violation defined in (36) as a function of the number of iterations.

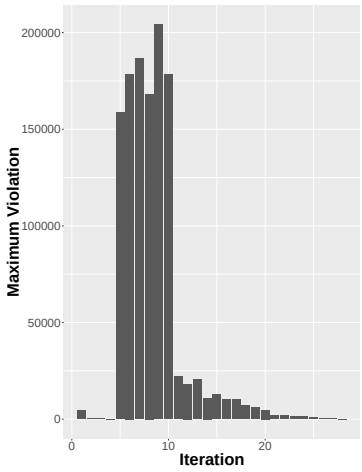


(c) Number of constraints added at each iteration.

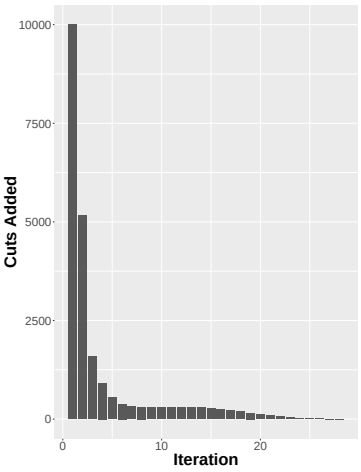
Figure 1: Progress of Algorithm 1 for $(n, d) = (10^4, 10)$, $Tol = 0.1$.



(a) Primal infeasibility defined in (34) as a function of the number of iterations.



(b) Maximum violation defined in (36) as a function of the number of iterations.



(c) Number of constraints added at each iteration.

Figure 2: Progress of Algorithm 1 for $(n, d) = (10^4, 10)$, $Tol = 0.05$.

Figures 1-2 suggest that Algorithm 1 makes rapid progress to decrease infeasibility. It takes twenty to twenty five iterations to decrease infeasibility (and violation) to near zero. Moreover, the number of cuts added decreases substantially as Algorithm 1 progresses. At the final few iterations when Algorithm 1 is close to convergence, the algorithm only adds typically less than 5 constraints at each iteration. Even for the larger sizes, we observe this trend of fewer constraints per iteration at later stages of the algorithm.

4.5. Comparison with other state of the art methods

In this section, we compare Algorithm 1 with two other recent methods proposed in the literature for the least squares convex regression Problem (4):

1. The cutting plane based method proposed in Balázs et al. [2015] and referred as aggregated cutting planes (ACP). The main difference with Algorithm 1 is that Balázs et al. [2015] use aggregated constraints in the reduced master problem.

2. The method in Mazumder et al. [2015], where the authors use an Alternating Direction Method of Multipliers (ADMM) framework to solve the least squares convex regression problem.

The ACP algorithm solves a variant of Problem (4), with bounds on both the function values and the subgradients, which we both set to ∞ in Algorithm 1. Both the ACP algorithm and Algorithm 1 were run with an upper bound of 1000 on the iteration limit and $Tol = 0.1$. Each of the rows with $n < 10^5$ were averaged over twenty random independently generated samples of that given size, while the larger ones ($n \geq 10^5$) were averaged over ten independently generated samples.

In Table 4, we record the final values of primal infeasibility and total running times for Algorithm 1 and ACP respectively As far as the quality of solution is considered, the final infeasibility indicates that the solutions obtained from both these methods are quite similar. However, Algorithm 1 is approximately twenty times faster than Algorithm ACP as (n, d) increase. For $(n, d) = (10^5, 10^2)$, while Algorithm 1 obtained solutions in a few hours, the ACP algorithm did not complete even after 24 hours, after which it was terminated. We remark that most of the time Algorithm ACP takes is to form the initial aggregation constraints. The results followed a similar pattern for $Tol = 0.05$, and thus we omit them for the sake of brevity.

n	d	(Alg. 1) Inf.	(Alg. 1) Run time	(ACP) Inf.	(ACP) Run time
10^3	10^1	0.0143 (0.0012)	1.9 (0.6)	0.0168 (0.0011)	7.3 (0.8)
10^4	10^1	0.0106 (0.0002)	25.2 (10.8)	0.0099 (0.0002)	411.7 (26.6)
10^4	10^2	0.0107 (0.0002)	153.5 (20.3)	0.0097 (0.0003)	4785.7(363.7)
10^5	10^1	0.0054 (0.0001)	1841.8 (230.9)	0.0050 (0.0001)	36842.7 (1391.03)

Table 4: Comparison for ℓ_2 convex regression between Algorithm 1 and ACP for $Tol = 0.1$.

In Table 5, we present a comparison between ADMM and Algorithm 1 for instances with $n = 10^3$ and $d = 10$. For larger sizes of $n = 10^4$, the ADMM method ran into memory issues and hence we do not report the performance for those cases. We set both the primal error and gradient error tolerance to be 0.1 in the ADMM algorithm. We observe that the ADMM algorithm has an edge on Algorithm 1 in terms of infeasibility, where as Algorithm 1 has the edge in terms of maximum violation. Algorithm 1 improves when the tolerance is reduced to 0.05 with similar primal infeasibility to the ADMM solution. However, the maximum violation is guaranteed to be at most 0.05 for Algorithm 1, while it is not satisfied by the ADMM method. The ADMM solution can be improved by reducing the primal and gradient error tolerances, but the point we emphasize is that Algorithm 1 gives a direct control on the maximum constraint violation.

n	d	Tol	(Alg. 1) Inf.	(Alg. 1) time	ADMM Inf.	ADMM time	ADMM Max viol.
10^3	10	0.1	0.0150	8.3	0.0059	47.8	0.0840
10^3	10	0.05	0.0029	142.7	0.0059	46.8	0.0885

Table 5: Comparison for ℓ_2 convex regression with ADMM.

4.6. Run times for ℓ_1 convex regression

In this section, we solve Problem (10), where we minimize the ℓ_1 loss rather than the usual least squares loss, and demonstrate how the algorithm scales in this context. Table 6 shows the run times and cuts added for a few instance sizes with tolerance set as 0.1.

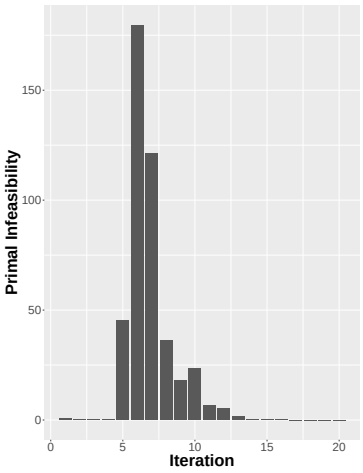
n	d	Cuts (Blocks)	Infeasibility	Run time (seconds)
10^3	10^1	24 (3)	0.0158 (0.0012)	2.9 (3.7)
10^4	10^1	10 (1)	0.0118 (0.0001)	25.3 (2.4)
10^4	10^2	168 (10)	0.0119 (0.0001)	2437.7 (470.3)
10^5	10^1	9 (1)	0.0056 (0.0001)	2501.9 (416.3)

Table 6: ℓ_1 convex regression - Run times for $Tol = 0.1$.

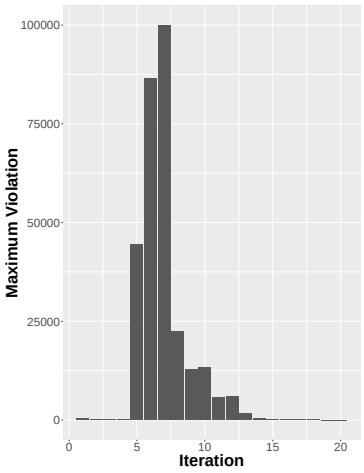
We observe that for the same 0.1 tolerance, the run times are higher than the ones obtained for ℓ_2 regression (Table 2). Also, as d increases for a given n, the run times increase as compared to the ℓ_2 case.

4.7. Experiments on real data

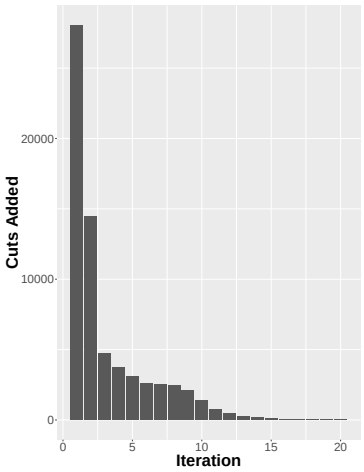
In this section, we apply some of our methods on a real world data set. This data set, which was considered in Mekaroonreung and Johnson [2012], was downloaded from `https://ampd.epa.gov/ampd/`. The data consists of the amount of heat input (in MMBtu) and the following four covariates – the NO_x emission rate, and emissions of SO_2, CO_2 and NO_x in tons. We consider nine years worth of data of electric utility units from 2000-2008, and after removing some rows with missing entries, we obtain a dataset with $n = 28,063$ and $d = 4$. We took a logarithmic transformation of the covariates, centered and scaled them so that they had mean zero and standard deviation of one. We ran the cutting plane algorithm for solving the least squares convex regression problem on this dataset, and present the results below.



(a) Primal infeasibility as a function of the number of iterations.

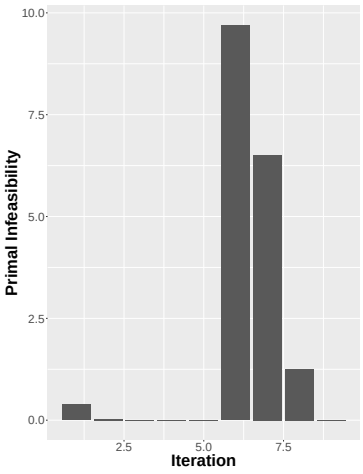


(b) Maximum violation as a function of the number of iterations.

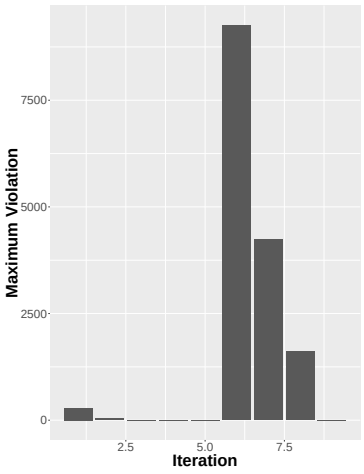


(c) Number of constraints added at each iteration.

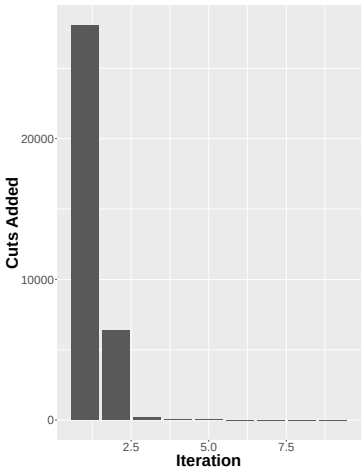
Figure 3: Progress of Algorithm 1 for $Tol = 0.01$.



(a) Primal infeasibility as a function of the number of iterations.



(b) Maximum violation as a function of the number of iterations.



(c) Number of constraints added at each iteration.

Figure 4: Progress of Algorithm 1 for $Tol = 0.05$.

We make the following observations.

- Figures 3-4 suggest that Algorithm 1 makes rapid progress to decrease infeasibility. For tolerance value of 0.05, it reaches optimality fairly quickly in around ten iterations, while it takes around twenty iterations for a smaller value of tolerance 0.01.
- Similar to the experiments on synthetic data, the number of cuts added decreases substantially as Algorithm 1 progresses. The final few iterations involve adding a very small number of cuts at each iteration.
- Finally, we include a note on the running times of the algorithm for this data. We observe a run time of $20 - 30$ minutes for a tolerance of 0.05, which is along the lines of what we

observe in Table 3 for synthetic data. On reducing the tolerance to 0.01, the run time increases to $60-70$ minutes, which is expected as the number of iterations doubles in this case.

4.8. Sparse convex regression

In this section, we present the computational results for Algorithms 2 and 3 applied to the problem of optimal subset selection in this setting. As for the continuous case, we generate $\mathbf{X}$ from a Gaussian distribution, and randomly sample the support set of size k from $\{1,\ldots,d\}$. We generate n d-dimensional vectors $\mathbf{x}_i$, each of which is generated from a Gaussian distribution with zero mean and correlation matrix Σ, where $\Sigma_{ij}=\rho^{|i-j|},\ 1\leq i,j\leq d$ for some correlation $0\leq\rho\leq 1$. Note that when $\rho=0$, the features are i.i.d, and higher ρ indicates that the correlation among the features is larger.

We use the convex function $\Phi(\mathbf{x})=\sum_{i\in S^*}x_i^2$, and the response data $y_i=\Phi(\mathbf{x}_i)+\epsilon_i,\ 1\leq i\leq n$. The errors ϵ_i are i.i.d. $N(0,\sigma^2)$, for all $i=1,\ldots,n$. We scale the data appropriately so that the Signal to Noise ratio (SNR) is 3. Again, we mean-center and normalize the features and response vectors to have unit ℓ_2 norm before providing the data as an input into Algorithms 2 and 3.

First, we demonstrate the value of using an MIO solver, by iteratively adding constraints to the primal problem according to Algorithm 2, and show the computational results in Tables 7 and 8. Next, we present the results for Algorithm 3 in Table 9, where we reformulate the sparse problem as minimizing a convex piecewise linear function over pure binary variables. If $\hat{S}$ is the optimal set obtained by our algorithms, we define accuracy as

$$\text{Accuracy}=\frac{|S^*\cap\hat{S}|}{k}, \tag{37}$$

where S^* is the true support. Next, we define the false positive rate, which is the fraction of features from the recovered support that are outside the true support S^*, i.e.,

$$\text{False Positive Rate}=\frac{|\hat{S}\setminus S^*|}{|\hat{S}|}. \tag{38}$$

A. Primal approach (Algorithm 2)

We present the results for the primal approach, as defined in Algorithm 2, in Tables 7 and 8 for $n=$ 50k and 100k respectively. First, we discuss how the value of M is selected in Problem (12) in the execution of Algorithm 2. In this primal approach, we solve the sparse convex regression problem with ℓ_∞ norm bounds on the subgradients. Mazumder et al. [2015] argue that the subgradients of the points near the boundary of $Conv(\mathbf{x}_1,\ldots,\mathbf{x}_n)$ grow large resulting in overfitting, and thus a bound on the subgradients is needed. Consequently, we vary the value of M, and select it via cross-validation. Using M^*, the maximum of the absolute of optimal solutions of Problem (14) and 15, we vary the value M as ηM^* by varying η, and calculate the validation error for each of these choices of M. For smaller values of M, the solution is too constrained, and for larger values, overfitting tends to occur.

We use the one standard error rule for cross validation [Hastie et al., 2009] while selecting the value of the parameter M. While performing cross validation to find the best hyperparameter, we typically select various values of the parameter $M_1,\ldots,M_s$, with corresponding mean errors and standard deviations of the mean error on the validation set given by $E_1,\ldots,E_s$ and $\sigma_1,\ldots,\sigma_s$ respectively. Typically, these values are obtained by $K-$fold cross validation. The

one standard error rule selects the parameter $\hat{M} = M_j$ where j is the smallest index in the set $\{i|E_i \leq E_{i^*} + \sigma_{i^*}\}$, and $i^* = \arg\min_{i=1,\ldots,s} E_i$.

For $n \leq 10,000$ the thresholding heuristic described in Section 3.3 was used to provide warm-starts to Algorithm 2. For $n > 10^4$, we ran the same thresholding heuristic on a sample of the points, and used the resulting support as a warm start. For $n \leq 10^4$, we solve Problems (14) and 15 to find the value M^* and set the value of $M = \eta M^*$. We choose η via cross validation from the set $\left\{10^{-3}, 10^{-2}, 10^{-1}, 0.5\right\}$. For $n > 10^4$, we avoid solving Problems (14) and (15) due to high solve times and select M from the set $\left\{10^{-3}, 10^{-2}, 10^{-1}, 0.5\right\}$ via cross validation. In this case, we set the ridge regularization parameter λ to be zero.

For every row in all the following tables, we report the median run times and mean accuracies over ten independently generated samples for the case when $n = 50\text{k}, d = 100$, and five samples for instances where $d = 500$ or $n = 100\text{k}$ with the standard deviation across these samples in the parentheses. The key finding is that by comparing Tables 2 and 8, we see that the sparse convex regression in fact solves faster than convex regression at least for $k = 10$. Moreover, the resulting accuracy is at least 95%. Furthermore, as n increases the accuracy of Algorithm 2 increases to near perfect value of 100%.

Tables 7 and 8 indicate that as n increases the accuracy increases and beyond a certain n the accuracy becomes 100%.

			$n = 50\text{k}$	
			Accuracy %	Run time
$\rho = 0.0$	$k = 10$	$d = 100$	100.0 (0.0)	1691.16 (284.8)
		$d = 500$	100.0 (0.0)	6522.37 (281.1)
	$k = 20$	$d = 100$	98.0(2.7)	2411.76(323.0)
		$d = 500$	92.0(2.3)	15276.47(5862.6)
$\rho = 0.1$	$k = 10$	$d = 100$	100.0 (0.0)	2778.83 (6369.8)
		$d = 500$	100.0 (0.0)	6326.79 (613.4)
	$k = 20$	$d = 100$	99.0 (2.2	2109.42 (292.8)
		$d = 500$	94.4(2.2)	11589.47(6883.6)
$\rho = 0.5$	$k = 10$	$d = 100$	100.0 (0.0)	2062.20 (508.0)
		$d = 500$	98.0 (4.5)	6083.46 (441.1)
	$k = 20$	$d = 100$	95.0 (3.5)	3158.10 (868.4)
		$d = 500$	93.3* (4.1)	25596.20 (3310.3)

Table 7: Accuracy% and Run times for Algorithm 2 for $n = 50\text{k}$.

		$n = 100\text{k}, d = 100$	
	k	Accuracy %	Run time
$\rho = 0.0$	10	95.0 (12.2)	7665.68 (38.23)
	20	100.0(0.0)	6605.0 (357.8)
$\rho = 0.1$	10	100.0 (0.0)	8939.83 (3575.0)
	20	100.0 (0.0)	11638.24 (1950.7)
$\rho = 0.5$	10	100.0 (0.0)	6823.41 (777.0)
	20	96.0 (2.2)	11937.81 (2120.6)

Table 8: Accuracy% and Run times for Algorithm 2 for $n = 100\text{k}, d = 100$.

We make the following observations:

- For $n = 50\text{k}$, increasing d from hundred to five hundred increases the run time by almost four times. The accuracy however, remains close to 100%. When we raised d to a thousand, the machines ran out of memory.
- The median run times and accuracies for $\rho = 0.1$ remain comparable in magnitude with the results for $\rho = 0$. For $(50\text{k}, 100, 10)$ the run times were highly skewed with values ranging from 2000 to a maximum of $15,000$ seconds. Excluding the three values with run times of over $10,000$ seconds, the median run time of the remaining seven instances was $2,509.64$ seconds with a standard deviation of 413.9 seconds.
- For $\rho = 0.5$, the solver could not solve one instance out of the five samples for $(50\text{k}, 500, 20)$. We report the median over the remaining four instances. The median run times, on average, increase with a corresponding increase in ρ.

B. Dual approach (Algorithm 3)

We present the results of the dual approach in Table 9 where we report the average accuracy (in %) and the average run times (in seconds) to provable optimality (MIO optimality gap of 1%). Each row for $d = 100$ is averaged over ten independently generated random instances of that size, with five such samples for problems where $d = 500$. As before, we use the one-standard error rule while using cross validation [Hastie et al., 2009] to select the final value of the parameter λ by varying it from 10^{-3} to 10^{-1}. The tolerance parameter ϵ in Algorithm 3 is set to 10^{-3} in all the experiments that follow.

			$n = 50$k	
			Accuracy %	Run time (seconds)
$\rho = 0.0$	$k = 10$	$d = 100$	97.0 (4.8)	2072.23 (236.5)
		$d = 500$	96.0 (5.5)	1785.43 (239.3)
	$k = 20$	$d = 100$	94.0 (3.9)	2356.16 (318.7)
		$d = 500$	91.1 (4.9)	1633.81 (523.8)
$\rho = 0.1$	$k = 10$	$d = 100$	97.0 (4.8)	2073.42 (434.3)
		$d = 500$	96.7 (5.8)	2178.06 (358.9)
	$k = 20$	$d = 100$	91.0 (4.6)	2055.16 (608.3)
		$d = 500$	90.0 (0.0)	1493.74 (169.9)
$\rho = 0.5$	$k = 10$	$d = 100$	92.0 (6.3)	1210.07 (171.4)
		$d = 500$	98.0 (4.5)	1858.62 (522.8)
	$k = 20$	$d = 100$	91.0 (4.2)	1122.74 (149.8)
		$d = 500$	89.0 (5.5)	1685.85 (340.2)

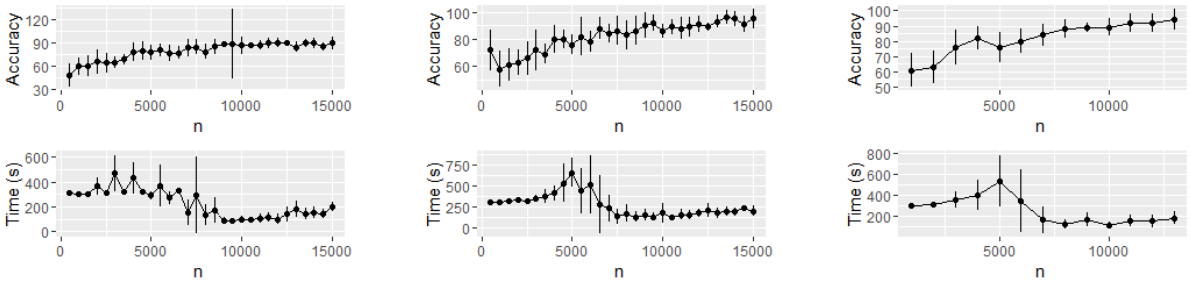
Table 9: Accuracy% and Run times for Algorithm 3 for $n = 50$k.

A key takeaway from Table 9 is that the dual algorithm 3 is able to solve instances with $50,000$ sample points in a few minutes for various values of correlation with high support recovery rates. For $n > 50,000$ the key bottleneck is computing the objective g and its subgradients. Recall that while evaluating g, Algorithm 3 requires the solution of a continuous ridge regularized convex regression problem on a restricted support set (Problem (18)) which has $O(nk)$ terms in its objective. The relaxation problem (24), which provides good quality warm starts, also becomes computationally expensive to solve due to the presence of dense semi-infinite constraints.

To summarize, both the primal and dual methods achieve exact or near-exact recovery on fairly noisy data (as evidenced by the fairly low Signal-to-Noise ratio of 3 of the data). While the primal approach seems to have an edge in terms of scalability over the dual approach, the dual approach is faster than the primal approach when $(n, d, k) = (5 \times 10^4, 500, 10)$.

C. Accuracy

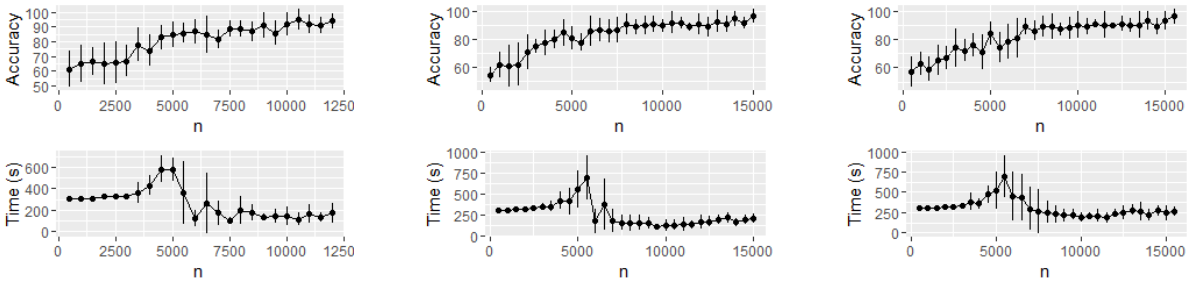
In this section, we report on the accuracy of the solutions obtained by Algorithm 3 as function of the parameters d, k, ρ, and SNR. We generate synthetic data for various values of each of these parameters and vary one of these parameters at a time while keeping the remaining constant. We present the mean accuracy and run times averaged over fifteen independently generated samples, along with their one standard deviation error bars.



(a) $\sqrt{\text{SNR}} = 3$ (b) $\sqrt{\text{SNR}} = 7$ (c) $\sqrt{\text{SNR}} = 20$

Figure 5: Accuracy and run times for varying SNR.

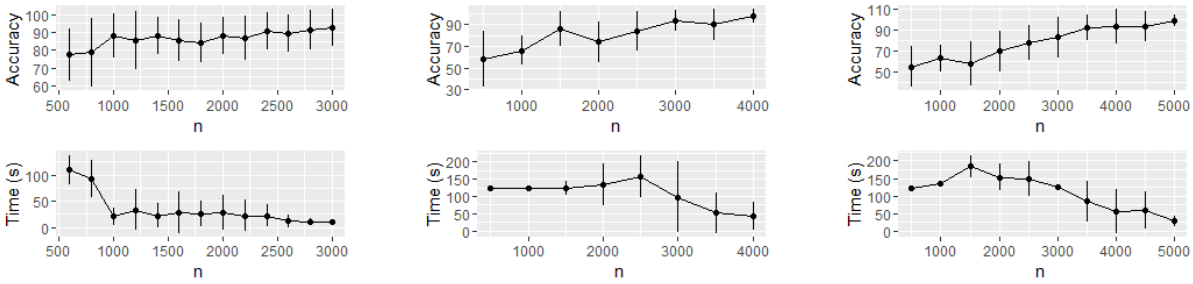
In Figures 5a–5c, we fix $(d, k, \rho) = (100, 10, 0.1)$, and vary $\sqrt{\text{SNR}} \in \{3, 7, 20\}$.



(a) $\rho = 0.0$ (b) $\rho = 0.1$ (c) $\rho = 0.5$

Figure 6: Accuracy and run times for varying correlation ρ.

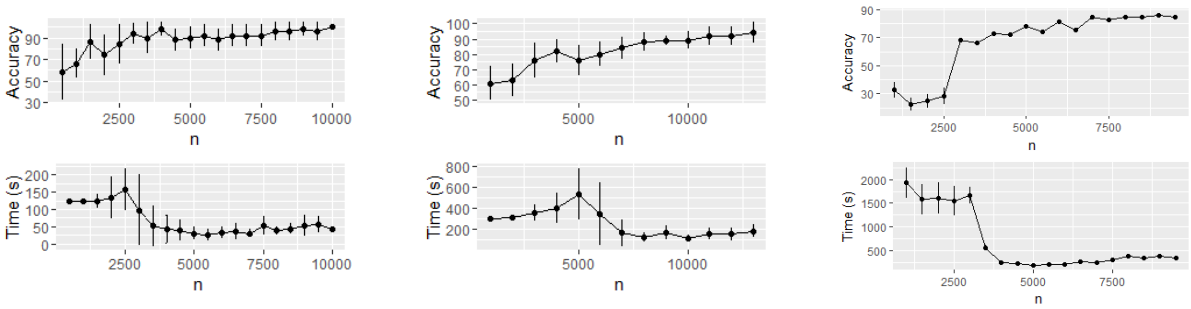
In Figures 6a–6c, we fix $(d, k, \sqrt{\text{SNR}}) = (100, 10, 20)$, and vary ρ in the set $\{0, 0.1, 0.5\}$.



(a) $d = 50$ (b) $d = 100$ (c) $d = 150$

Figure 7: Accuracy and run times for varying dimension d.

In Figures 7a–7c, we fix $(k, \rho, \sqrt{\text{SNR}}) = (5, 0.1, 20)$ and vary d in the set $\{50, 100, 150\}$.



(a) $k = 5$ (b) $k = 10$ (c) $k = 15$

Figure 8: Accuracy and run times for varying sparsity parameter k.

Finally, in Figures 8a–8c, we fix $(d, \rho, \sqrt{\text{SNR}}) = (100, 0.1, 20)$ and vary the sparsity level k in the set $\{5, 10, 15\}$. We solve the problems with a time cutoff of two, five, and ten minutes for $k = 5, 10, 15$ respectively, and take the best solution obtained until that time in case the incumbent solution has not been guaranteed to be optimal by that time.

We make the following observations:

(a) As n increases, the accuracy of Algorithm 3 increases and the running time decreases. These observations are consistent with the findings of Bertsimas and Van Parys [2016] in the context of sparse linear regression.

(b) As SNR increases, we reach higher accuracy for smaller values of n, that is the problem becomes easier (see Figures 5a–5c).

(c) To reach accuracy of 95% we need n equal to $10,000$, $12,000$ and $15,000$ for ρ being 0, 0.1 and 0.5, respectively (see Figures 6a–6c).

(d) To reach accuracy of 95% we need n equal to $3,000$, $4,000$ and $5,000$ for d being 50, 100 and 150, respectively (see Figures 7a–7c).

(e) To reach accuracy of 90% we need n equal to $2,500$, $8,000$ and $10,000$ for k being 5, 10 and 15, respectively (Figures 8a–8c).

D. False Positive rates

In this section, we investigate the false discovery rate for the estimator resulting from this algorithm. So far we have taken k, the sparsity parameter as a given in all of our experiments. In reality however, this value needs to be inferred from the data, and is usually done by cross validation. Thus, it is imperative that the algorithm not only choose the relevant features, but also that it picks no extra spurious ones and mark them as relevant.

To check this, we performed an experiment with simulated data for $(n, d) = (10000, 100)$ with five features chosen randomly. We vary k in the set $\{3, \ldots, 10\}$, and choose the best k by five fold cross validation. We then run our algorithms for that value of k, and report the median false positive rate over ten independently generated samples. We present our results for both the primal and dual algorithms in Tables 10 and 11 respectively.

For the dual approach, we impose a time limit of 120 seconds, and take the best solution obtained by that point of time and for the primal method, we do not impose any such time limit. We report the median false positive rate over ten independently chosen samples for different values of ρ and SNR. This suggests that our algorithms not only pick the relevant features, but are also able to control for spurious discoveries.

$\sqrt{\text{SNR}}$	$\rho = 0.0$	$\rho = 0.1$	$\rho = 0.5$
3	0%	0%	0%
7	0%	0%	0%
20	0%	0%	0%

Table 10: False Positive rate for Algorithm 3.

$\sqrt{\text{SNR}}$	$\rho = 0.0$	$\rho = 0.1$	$\rho = 0.5$
3	0%	0%	0%
7	0%	0%	0%
20	0%	0%	0%

Table 11: False Positive rate for Algorithm 2.

4.9. Discussion

(a) For the problem of convex regression, we see that Algorithm 1 has a significant edge over other state of the art methods in terms of run time and accuracy. Our approach allows us to solve problems of $n = 100,000$ and $d = 100$ in hours. Also, it is flexible enough to accommodate other constraints such as coordinate-wise monotonicity and norm bounded subgradients.

(b) For the sparse convex regression problem, the dual approach (Algorithm 3) has an edge over the primal method (Algorithm 2) in run times and scalability. Surprisingly, Algorithm 3 solves the sparse convex regression problem in times comparable to the continuous case, implying that the price of sparsity is small. Since we break new ground in this area, we are unable to include any comparisons to other methods.

(c) For the sparse convex regression problem, the primal approach scales to problems of the size $(n, d, k) = (10^5, 100, 10)$ in hours, while the dual approach scales to $(n, d, k) = (5 \times 10^4, 500, 10)$ in minutes. We perform various experiments by varying the degree of correlation among the covariates ρ, signal to noise ratio (SNR), number of features d, sparsity level k, and demonstrate that our algorithms achieve near perfect support recovery as n increases. Also, we note that both Algorithms 2 and 3 limit the false discovery rate.

References

G. Allon, M. Beenstock, S. Hackman, U. Passy, and A. Shapiro. Nonparametric estimation of concave production technologies by entropic methods. *J. Appl. Econometrics*, 22:795–816, 2007.

G. Balázs, A. György, and A. Szepesvári. Near-optimal max-affine estimators for convex regression. *In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 38:56–64, 2015.

D. Bertsimas and R. Mazumder. Least quantile regression via modern optimization. *Annals of Statistics*, 42(6):2494–2525, 2014.

D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*, volume 6. Athena Scientific, 1997.

D. Bertsimas and B. Van Parys. Sparse high dimensional regression: Exact scalable algorithms and phase transitions. Submitted, 2016.

D. Bertsimas, A. King, and R. Mazumder. Best subset selection via a modern optimization lens. *Annals of Statistics*, 44(2):813–852, 2016.

J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *arXiv preprint arXiv:1411.1607*, 2014.

R. E. Bixby. A brief history of linear and mixed-integer programming computation. *Documenta Mathematica, Extra Volume: Optim. Stories*, pages 107–121, 2012.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

H. Chen and D. D. Yao. *Fundamentals of Queueing Networks: Performance, Asymptotics, and Optimization*. Springer-Verlag, 2001.

I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *arXiv preprint arXiv:1508.01982*, 2015.

A. Goldenshluger and A. Zeevi. Recovering convex boundaries from blurred and noisy observations. *Annals of Statistics*, 34:1375–1394, 2006.

Gurobi. Gurobi Optimizer Reference Manual. `http://www.gurobi.com`, 2015.

Qiyang Han and Jon A Wellner. Multivariate convex regression: global risk bounds and adaptation. *arXiv preprint arXiv:1601.06844*, 2016.

L. A. Hannah and D. B. Dunson. Multivariate convex regression with adaptive partitioning. *J. Mach. Learn. Res.*, 14:3261–3294, 2013.

L. A. Hannah, W. B. Powell, and D. B. Dunson. Semiconvex regression for metamodeling based optimization. *SIAM Journal on Optimization*, 24(2):573–597, 2014.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics, Springer, New York, second edition, 2009.

James E Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

A. S. Lele, S. R. Kulkarni, and A. S. Willsky. Convex-polygon estimation from support-line measurements and applications to target reconstruction from laser-radar data. *Journal of the Optical Society of America, Series A*, 9:1693–1714, 1992.

E. Lim and P. W. Glynn. Consistency of multidimensional convex regression. *Operations Research*, 60(1):196–208, 2012.

A. Magnani and S. Boyd. Convex piecewise-linear fitting. *Optimization and Engineering*, 10 (1):1–17, 2009.

R. Mazumder, A. Choudhury, G. Iyengar, and B. Sen. A computational framework for multivariate convex regression and its variants. arXiv preprint, 2015. URL `http://arxiv.org/abs/1509.08165v1`.

Maethee Mekaroonreung and Andrew L Johnson. Estimating the shadow prices of so2 and nox for us coal power plants: a convex nonparametric least squares approach. *Energy Economics*, 34(3):723–732, 2012.

B. Natarajan. Sparse approximate solutions to linear systems. *SIAM Journal on Computing*, 24:227–234, 1995.

G. Nemhauser. Integer programming: The global impact. EURO, INFORMS, 2013. URL `https://smartech.gatech.edu/handle/1853/49829`.

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

E. Seijo and B. Sen. Nonparametric least squares estimation of a multivariate convex regression function. *Annals of Statistics*, 39:1633–1657, 2011.

A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming, Modeling and Theory*. Society for Industrial and Applied Mathematics and the Mathematical Programming Society, 2009.

H. Topaloglu and W. B. Powell. An algorithm for approximating piecewise linear concave functions from sample gradients. *Operations Research Letters*, 31:66–76, 2003.

H. R. Varian. The nonparametric approach to demand analysis. *Econometrica*, 50(4):945–973, 1982.

H. R. Varian. The nonparametric approach to production analysis. *Econometrica*, 52(3):579–597, 1984.

M. Xu, M. Chen, and J. Lafferty. Faithful variable screening for high-dimensional convex regression. 2014. URL `http://arxiv.org/abs/1411.1805`. arXiv preprint.