A Computational Framework for Multivariate Convex Regression and its Variants

Rahul Mazumder[*]

MIT Sloan School of Management and Operations Research Center,

Massachusetts Institute of Technology

Arkopal Choudhury[†]

Department of Biostatistics, University of North Carolina at Chapel Hill

Garud Iyengar[‡]

IEOR Department, Columbia University

Bodhisattva Sen[§]

Department of Statistics, Columbia University

September 29, 2015

Abstract

We study the nonparametric least squares estimator (LSE) of a multivariate convex regression function. The LSE, given as the solution to a quadratic program with $O(n^2)$ linear constraints (n being the sample size), is difficult to compute for large problems. Exploiting problem specific structure, we propose a scalable algorithmic framework based on the augmented Lagrangian method to compute the LSE. We develop a novel approach to obtain smooth convex approximations to the fitted (piecewise affine) convex LSE and provide formal bounds on the quality of approximation. When the number of samples is not too large compared to the dimension of the predictor, we propose a regularization scheme — Lipschitz convex regression — where we constrain the norm of the subgradients, and study the rates of convergence of the obtained LSE. Our algorithmic framework is simple and flexible and can be easily adapted to

1

handle variants: estimation of a non-decreasing/non-increasing convex/concave (with or without a Lipschitz bound) function. We perform numerical studies illustrating the scalability of the proposed algorithm.

Keywords: Augmented Lagrangian method, Lipschitz convex regression, nonparametric least squares estimator, scalable quadratic programming, smooth convex regression.

1 Introduction

Consider the task of fitting a *multivariate convex* function to observations $\{(\mathbf{X}_i, Y_i)\}_{i=1}^n$ where the covariates $\mathbf{X}_i \in \Re^d$, $d \geq 1$, and the response $Y_i \in \Re$. This problem has been recently considered by several authors; see e.g., Seijo and Sen (2011) and Lim and Glynn (2012) and the references therein. Convex (concave) regression problems are common in economics, operations research and reinforcement learning. In economics, production functions and utility function preferences are usually known to be concave (see Allon et al. (2007) and Varian (1984)) whereas consumer preferences (see Meyer and Pratt (1968)) are often assumed convex. In operations research and reinforcement learning, value functions for stochastic optimization problems can be convex (see Shapiro et al. (2009)).

Probably the most natural estimator of the convex regression function in this setup is the least squares estimator (LSE) $\hat{\phi}_n$, defined as a minimizer of the empirical L_2-norm, i.e.,

$$\hat{\phi}_n \in \arg\min \sum_{i=1}^n (Y_i - \psi(\mathbf{X}_i))^2, \tag{1}$$

where the minimum is taken over all convex functions $\psi : \Re^d \to \Re$. See Seijo and Sen (2011) and Lim and Glynn (2012) for the characterization, computation and consistency of the LSE. An appealing feature of the convex LSE over most other nonparametric function estimation methods is that no tuning parameters (e.g., smoothing bandwidths as in kernel regression) need to be specified in the estimation procedure.

The seemingly infinite dimensional optimization problem in (1) can indeed be reduced to a finite dimensional one as described below. Letting $\hat{\theta}_i = \hat{\phi}_n(\mathbf{X}_i)$, for $i = 1, \ldots, n$, we can compute $\hat{\boldsymbol{\theta}} = (\hat{\theta}_1, \ldots, \hat{\theta}_n)$ by solving the following Quadratic Program (QP):

$$\begin{aligned} \underset{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n; \boldsymbol{\theta}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{Y} - \boldsymbol{\theta}\|_2^2 \\ s.t. \quad & \theta_j + \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \rangle \leq \theta_i; \quad i \neq j \in \{1, \ldots, n\}, \end{aligned} \tag{2}$$

where $\boldsymbol{\Delta}_{ij} := \mathbf{X}_i - \mathbf{X}_j \in \Re^d$, $\mathbf{Y} = (Y_1, \ldots, Y_n)^\top$, $\boldsymbol{\xi}_i \in \Re^d$, $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)^\top \in \Re^n$, $\langle \cdot, \cdot \rangle$ denotes the usual inner product and $\| \cdot \|_2$ denotes the standard Euclidean norm. In words, Problem (2) estimates the function values at the points $\mathbf{X}_i$'s and also delivers estimates of subgradients $\boldsymbol{\xi}_i$'s of the function at the n points $\mathbf{X}_i$'s. A natural way to extend $\hat{\phi}_n$ to a convex function defined on the whole of $\Re^d$ is to use the following rule:

$$\hat{\phi}_n(\mathbf{x}) = \max_{j=1,\ldots,n} \left\{ \hat{\theta}_j + \langle \mathbf{x} - \mathbf{X}_j, \hat{\boldsymbol{\xi}}_j \rangle \right\}. \tag{3}$$

Seijo and Sen (2011) used off-the-shelf interior point solvers (e.g., `cvx`, `MOSEK`, etc.) to solve the optimization problem (2). However, off-the-shelf interior point solvers are not scalable and quickly become prohibitively expensive for sample size $n \geq 300$ due to the presence of $O(n^2)$ linear constraints. This motivates the present study where we propose scalable first order methods based on modern convex optimization techniques and investigate the statistical and computational issues of variants of convex regression. During the course of this work, we became aware of the conference paper by Aybat and Wang (2014) which studies algorithms for an approximation of Problem (2) by adding a ridge regularization on the subgradients $\boldsymbol{\xi}_j$'s – the authors report computational savings over interior point methods for problem sizes up to $n = 1600$. Our algorithmic approach in this paper, however, is different and we demonstrate scalability to problems of larger size.

We summarize our main contributions below.

Algorithmic Framework: We propose, in Section 2, an algorithmic framework based on the augmented Lagrangian method (Bertsekas, 1999; Boyd et al., 2011) and first order optimization techniques (Nesterov, 2004) to solve the Problem (2), which is a quadratic program (QP) with $O(n^2)$ constraints in $O(nd)$ variables. Our algorithmic framework is scalable, yet general enough to accommodate variants of the problem (see the discussion below). The convergence properties of the algorithm are subtle, and are discussed. An important aspect of our algorithm is that it heavily exploits problem structure to enhance its computational scalability. This enables us to compute the convex LSE for problem sizes much beyond the capacity of off-the-shelf interior point methods. Our approach delivers moderate accuracy solutions for $n \approx 5000$ within a few minutes; and scales quite easily for problems with $n \approx 10000$ or more. When $d \ll n$, the computational cost of our algorithm is dominated by $O(n^2)$, stemming from matrix-vector multiplications — the cost is indeed reasonable since Problem (2) has $O(n^2)$ linear constraints. Note that solving Problem (2) with off-the-shelf interior point methods has a complexity of $O(n^3 d^3)$; see e.g., Boyd and Vandenberghe (2004).

Smooth Estimators: A characteristic feature of the LSE $\hat{\phi}_n$, as defined in (3), is that it is piecewise affine and hence non-smooth. This may be perceived as a possible drawback of the LSE, since in various statistical applications, smooth estimators are desired; see e.g., Birke and Dette (2007), Aguilera et al. (2011), Du et al. (2013) and the references therein. Most of the existing work on function estimation under both smoothness and shape constraints are for one dimensional functions — estimating a smooth multivariate convex function is a challenging statistical problem. We present in Section 3, a new approach to this problem, using tools in convex optimization — we perform a smoothing operation on the convex LSE that retains convexity. Our approach yields an estimator that is (a) smooth, (b) convex and (c) is uniformly close (up to any desired precision) to the LSE. In fact, we can provide theoretical bounds on the quality of the smooth approximation. We emphasize that our approach is very different from the usual techniques (e.g., kernel smoothing) employed in nonparametric statistics to obtain smooth approximations of non-smooth estimators. In fact, this technique of smoothing is quite general and applies well beyond the specific instance of piecewise affine functions arising in the context of convex LSE.

Lipschitz Regularization: In the presence of a limited sample size the convex LSE $\hat{\phi}_n$ may lead to overfitting, especially towards the boundary of the convex hull of the covariate domain. This is due to the fact that the fitted subgradients can take very large values near the boundary. To ameliorate this problem we propose least squares estimation of the convex regression function with a (user specified) bounded Lipschitz norm and derive the statistical rates of convergence of this estimator in Section 4. The Lipschitz convex LSE thus obtained has lower prediction error and risk when compared to the usual convex LSE. We discuss data-dependent ways to estimate the tuning parameter (i.e., the Lipschitz norm) in practice.

Additional Structure: In several applications of interest, e.g., while estimating production functions and supply/demand functions, it is natural to impose the requirement that the function not only be convex/concave, but that it also be non-decreasing/non-increasing. Our algorithmic framework can be easily adapted to consider such variants of convex regression. We study such variants of our algorithm in Section 5.

Experiments: We perform a detailed simulation study in Section 6 demonstrating the superior scalability and performance of our framework over existing publicly available algorithms for this problem. We also consider a few real data examples and highlight the usefulness of convex regression.

Broader Outlook and Context: Our paper is intended to be viewed as a contribution to the larger literature on function estimation in the presence of "shape constraints". The earliest (and most extensively studied) such problem is that of isotonic regression (in which the regression function is presumed to be monotone); see e.g., Brunk (1955), and Ayer et al. (1955). Recently there has been an upsurge of interest in these shape constrained problems, especially in the multivariate setting; see e.g., Cule et al. (2010) and Seregin and Wellner (2010) for developments in the context of density estimation, and Seijo and Sen (2011) and Hannah and Dunson (2013) for developments in the regression context.

2 Algorithm for Multivariate Convex Regression

In this section we investigate a scalable algorithm for computing the LSE defined in (2). We first give a brief discussion as to why Problems (2) and (1) are equivalent. Observe that any solution $\boldsymbol{\xi}_i^*$'s and $\boldsymbol{\theta}^*$ of Problem (2) can be extended to a convex function by the interpolation rule (3). Note that $\hat{\phi}_n(\mathbf{x})$ thus defined is convex in $\Re^d$ and has the same loss function as the optimal objective value of Problem (2). Thus solving Problem (2) is equivalent to solving (1).

We will employ a prototypical version of the alternating direction method of multipliers (ADMM) algorithm; see Bertsekas (1999); Boyd et al. (2011). For this we consider the following equivalent representation for (2):

$$
\begin{aligned}
\underset{\boldsymbol{\xi}_1,\dots,\boldsymbol{\xi}_n;\boldsymbol{\theta};\boldsymbol{\eta}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{Y} - \boldsymbol{\theta}\|_2^2 \\
s.t. \quad & \eta_{ij} = \theta_j + \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \rangle - \theta_i; \qquad \eta_{ij} \le 0; \qquad i, j = 1, \dots, n,
\end{aligned}
\tag{4}
$$

where $\boldsymbol{\eta} = ((\eta_{ij})) \in \Re^{n \times n}$ is a matrix with (i, j)'th entry η_{ij}. Define the augmented Lagrangian corresponding to the above formulation as

$$
\begin{aligned}
\mathcal{L}_\rho((\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_n; \boldsymbol{\theta}; \boldsymbol{\eta}); \boldsymbol{\nu}) \quad := \quad & \frac{1}{2}\|\mathbf{Y} - \boldsymbol{\theta}\|_2^2 + \sum_{i,j} \nu_{ij} \left(\eta_{ij} - (\theta_j + \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \rangle - \theta_i) \right) \\
& + \frac{\rho}{2} \sum_{i,j} \left(\eta_{ij} - (\theta_j + \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \rangle - \theta_i) \right)^2
\end{aligned}
\tag{5}
$$

where $\boldsymbol{\nu} \in \Re^{n \times n}$ is the matrix of dual variables. We will employ a multiple-block version of ADMM for the above problem following Boyd et al. (2011). This requires the following sequential updates as described in Algorithm 1.

Algorithm 1 Multiple-block splitting ADMM for (4)

Initialize variables $(\boldsymbol{\xi}_1^{(1)}, \ldots, \boldsymbol{\xi}_n^{(1)})$, $\boldsymbol{\theta}^{(1)}$, $\boldsymbol{\eta}^{(1)}$ and $\boldsymbol{\nu}^{(1)}$.

Perform the following Steps 1—4 for $k \geq 1$ till convergence.

   1. Update the subgradients $(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n)$:

$$(\boldsymbol{\xi}_1^{(k+1)}, \ldots, \boldsymbol{\xi}_n^{(k+1)}) \in \underset{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n}{\arg\min} \mathcal{L}_\rho \left((\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n; \boldsymbol{\theta}^{(k)}; \boldsymbol{\eta}^{(k)}); \boldsymbol{\nu}^{(k)} \right). \qquad (6)$$

   2. Update the function values $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^{(k+1)} \in \underset{\boldsymbol{\theta}}{\arg\min} \mathcal{L}_\rho \left((\boldsymbol{\xi}_1^{(k+1)}, \ldots, \boldsymbol{\xi}_n^{(k+1)}; \boldsymbol{\theta}; \boldsymbol{\eta}^{(k)}); \boldsymbol{\nu}^{(k)} \right). \qquad (7)$$

   3. Update the residual matrix $\boldsymbol{\eta}$:

$$\boldsymbol{\eta}^{(k+1)} \in \underset{\boldsymbol{\eta}\,:\,\eta_{ij} \leq 0,\, \forall i,j}{\arg\min} \mathcal{L}_\rho \left((\boldsymbol{\xi}_1^{(k+1)}, \ldots, \boldsymbol{\xi}_n^{(k+1)}; \boldsymbol{\theta}^{(k+1)}; \boldsymbol{\eta}); \boldsymbol{\nu}^{(k)} \right). \qquad (8)$$

   4. Update the dual variable:

$$\nu_{ij}^{(k+1)} \leftarrow \nu_{ij}^{(k)} + \rho \left(\eta_{ij}^{(k+1)} - \left(\theta_j^{(k+1)} + \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j^{(k+1)} \rangle - \theta_i^{(k+1)} \right) \right); i, j = 1, \ldots, n. \quad (9)$$

2.1 ADMM with three blocks: Algorithm 1

We now describe the computations of the different steps appearing in Algorithm 1.

2.1.1 Performing the Updates of Algorithm 1

Updating the subgradients: solving Problem (6)

Problem (6) is an unconstrained QP in nd variables, where the variables are $\boldsymbol{\xi}_j \in \Re^d, j = 1, \ldots, n$. Solving such a problem naively has complexity $O(n^3 d^3)$. However, the problem is separable in $\boldsymbol{\xi}_j$'s — hence it suffices to consider the updates for each $\boldsymbol{\xi}_j$ asynchronously, over different j's. For every j, the subgradient vector $\boldsymbol{\xi}_j$ can be computed as follows:

$$\hat{\boldsymbol{\xi}}_j \in \underset{\boldsymbol{\xi}_j}{\arg\min} \sum_{i=1}^n \left(\bar{\eta}_{ij} - \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \rangle \right)^2 \implies \hat{\boldsymbol{\xi}}_j = \left(\sum_i \boldsymbol{\Delta}_{ij} \boldsymbol{\Delta}_{ij}^\top \right)^{-1} \left(\sum_i \boldsymbol{\Delta}_{ij} \bar{\eta}_{ij} \right) \quad (10)$$

where $\bar{\eta}_{ij} = \nu_{ij}/\rho + \eta_{ij} - (\theta_j - \theta_i)$, which is a least squares regression problem for every $j = 1, \ldots, n$. If $n > d$ and the design points come from a continuous distribution, $\hat{\boldsymbol{\xi}}_j$

is unique with probability one. We note that the matrices $\overline{\Delta}_j := \left(\sum_i \Delta_{ij}\Delta_{ij}^\top\right)^{-1}, j = 1, \ldots, n$, need not be computed at every iteration, they can be computed offline at the onset of the algorithm: computing $\sum_i \Delta_{ij}\Delta_{ij}^\top$ takes $O(nd^2)$ and the inversion $O(d^3)$ for every j. Thus, the total cost for all j values would be $O(n^2d^2+nd^3)$, a computation that can be done once in parallel at the onset of the algorithm. Once the inverses are computed, computing $\hat{\boldsymbol{\xi}}_j$ for all j requires an additional cost of $O(n^2d + nd^2)$ — this is due to the cost of computing the $\bar{\eta}_{ij}$'s, the matrices $\sum_i \Delta_{ij}\bar{\eta}_{ij}$ and the subsequent matrix multiplications leading to $\hat{\boldsymbol{\xi}}_j$. Since we typically have $d \ll n$, the cost per iteration is $O(n^2)$.

Updating the function values: solving Problem (7)

We introduce the following notations:

$$\{\mathbf{D}\boldsymbol{\theta}\}_{(i-1)n+j} = (\theta_j - \theta_i), \quad \{\text{vec}(\boldsymbol{\nu})\}_{(i-1)n+j} = \nu_{ij}, \quad \{\text{vec}(\widetilde{\boldsymbol{\eta}})\}_{(i-1)n+j} = (\eta_{ij} - \langle \Delta_{ij}, \boldsymbol{\xi}_j \rangle),$$

for $i, j = 1, \ldots, n$, where $\mathbf{D}$ is a sparse $n^2 \times n$ matrix. Using the above notation, the optimization problem (7) reduces to the minimization of the following function (with respect to $\boldsymbol{\theta}$):

$$\frac{1}{2}\|\mathbf{Y} - \boldsymbol{\theta}\|_2^2 - \langle \text{vec}(\boldsymbol{\nu}), \mathbf{D}\boldsymbol{\theta} \rangle + \frac{\rho}{2}\|\text{vec}(\tilde{\boldsymbol{\eta}}) - \mathbf{D}\boldsymbol{\theta}\|_2^2,$$

which is equivalent to solving the following system:

$$(\mathbf{I} + \rho\mathbf{D}^\top\mathbf{D})\hat{\boldsymbol{\theta}} = \underbrace{\mathbf{Y} + \mathbf{D}^\top\text{vec}(\boldsymbol{\nu}) + \rho\mathbf{D}^\top\text{vec}(\tilde{\boldsymbol{\eta}})}_{:=\mathbf{v}}. \tag{11}$$

Computing the vector $\mathbf{v}$ will cost $O(n^2)$ flops. The matrix appearing on the left-hand side of the above equation, $(\mathbf{I} + \rho\mathbf{D}^\top\mathbf{D})$, has dimension $n \times n$. A direct inversion of the matrix to solve for $\boldsymbol{\theta}$ will have a complexity of $O(n^3)$. We show below that the cost can be reduced substantially by exploiting the structure of the linear system (11). From the definition of $\mathbf{D}$ we have $\mathbf{D}^\top\mathbf{D} = 2n\mathbf{I}_{n\times n} - 2\,\mathbf{1}\mathbf{1}^\top$, where $\mathbf{1}$ denotes the $n \times 1$ vector of all ones and $\mathbf{I}_{n\times n}$ is the identity matrix of order n. Thus we have $(\mathbf{I} + \rho\mathbf{D}^\top\mathbf{D}) = (1 + 2n\rho)\mathbf{I} - 2\rho\mathbf{1}\mathbf{1}^\top$ and

$$(\mathbf{I} + \rho\mathbf{D}^\top\mathbf{D})^{-1} = \left(\frac{1}{1 + 2n\rho}\mathbf{I} + \frac{2\rho}{1 + 2n\rho}\mathbf{1}\mathbf{1}^\top\right), \tag{12}$$

using which the system (11) can be solved for $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{\theta}} = \left(\frac{1}{1+2n\rho}\mathbf{I} + \frac{2\rho}{1+2n\rho}\mathbf{1}\mathbf{1}^\top \right)\mathbf{v} \implies \hat{\theta}_i = \frac{1}{(1+2n\rho)}\left(v_i + 2\rho\sum_i v_i \right). \quad (13)$$

The above computation can be done quite efficiently in $O(n)$ flops, given $\mathbf{v}$. Thus the total cost per iteration in computing $\hat{\boldsymbol{\theta}}$ is $O(n^2)$.

Updating the residuals: solving Problem (8)

Updating the residuals, i.e., η_{ij}'s, are simple — due to the separability of the objective function the partial minimization splits into n^2 independent univariate optimization problems. It suffices to consider the following update rule for any (i,j):

$$\hat{\eta}_{ij} = \min\left\{ \theta_j + \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \rangle - \theta_i - \frac{1}{\rho}\nu_{ij}, 0 \right\}; \quad i,j = 1,\ldots,n. \quad (14)$$

Note that the inner products $\langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \rangle$ are already available as a by-product of the $\boldsymbol{\theta}$ update, while solving Problem (7); hence they need not be computed from scratch. Thus, the cost for performing this operation is $O(n^2)$, since there are n^2 many variables to be updated.

Computational Complexity: Note that updating the dual variable $\boldsymbol{\nu}$ using (9) requires $O(n^2)$ flops. Thus, gathering the discussion above, the cost per iteration of Algorithm 1 is $O(\max\{n^2 d, nd^3\})$, with an additional $O(n^2 d^2 + nd^3)$ for the offline computation of matrix inverses for Problem (6). If $d \ll n$, the total cost is $O(n^2)$, which seems reasonable since the optimization problem involves $O(n^2)$ constraints.

2.1.2 Optimality Conditions for Problem (2)

Let $\boldsymbol{\theta}^*, \eta_{ij}^*$ and $\boldsymbol{\xi}_j^*$, for $i,j \in \{1,\ldots,n\}$, denote optimal solutions to Problem (2), or equivalently, Problem (5), and let $\boldsymbol{\nu}^*$ denote an optimal dual variable. The optimality conditions for the problem (Boyd and Vandenberghe, 2004; Boyd et al., 2011) are

given by:

$$\eta_{ij}^* - \left(\theta_i^* + \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j^* \rangle - \theta_j^*\right) = 0, \qquad i, j = 1, \ldots, n, \tag{15a}$$

$$\sum_{i=1}^n \nu_{ij}^* \Delta_{ij} = 0, \qquad j = 1, \ldots, n, \tag{15b}$$

$$(\boldsymbol{\theta}^* - \mathbf{Y}) - \mathbf{D}^\top \mathrm{vec}(\boldsymbol{\nu}^*) = \mathbf{0}, \tag{15c}$$

$$\eta_{ij}^* - \min\left\{\eta_{ij}^* - \frac{1}{\rho}\nu_{ij}^*, 0\right\} = 0, \qquad i, j = 1, \ldots, n, \tag{15d}$$

where (15a) is the primal feasibility condition, (15b) is the optimality condition with respect to $\boldsymbol{\xi}_j$'s, (15c) is the optimality condition with respect to $\boldsymbol{\theta}$ and (15d) denotes the optimality condition with respect to $\boldsymbol{\eta}$.

2.2 Convergence properties

Algorithm 1 is a direct application of the schematic multiple-block (with three blocks) ADMM algorithm described in Boyd et al. (2011) — it is quite simple and easy to implement. There is, however, one caveat. As soon as the number of blocks becomes larger than two, the multiple block version of ADMM however, does not necessarily converge — see e.g., Chen et al. (2014) for a counter example. Multiple block versions of ADMM, however, can be shown to be convergent, under certain restrictive assumptions on the problem and by possibly modifying the algorithm; see e.g., Hong and Luo (2012). While a thorough convergence analysis of multiple block ADMM is not the main theme or focus of this paper, it is indeed quite simple to check if Algorithm 1 has converged or not by verifying if the conditions of optimality (15) are met. In all of our experiments, Algorithm 1 was indeed found to converge. Nevertheless, we discuss a simple variant of Algorithm 1 that has superior convergence guarantees. The main principle behind the modification(s) is quite simple: since a three block version of the ADMM is not guaranteed to converge, one can resort to a two-block version of the ADMM. One way to obtain a two-block version of ADMM is by combining the serial updating of $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$, i.e., Problems (7) and (8) into one that jointly optimizes over both $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$, i.e.,

$$(\boldsymbol{\theta}^{(k+1)}, \boldsymbol{\eta}^{(k+1)}) \in \underset{\boldsymbol{\theta}, \boldsymbol{\eta}:\, \eta_{ij} \leq 0, \forall i, j}{\arg\min} \quad \mathcal{L}_\rho\left((\boldsymbol{\xi}_1^{(k+1)}, \ldots, \boldsymbol{\xi}_n^{(k+1)}; \boldsymbol{\theta}; \boldsymbol{\eta}); \boldsymbol{\nu}^{(k)}\right). \tag{16}$$

The update in $\boldsymbol{\xi}_j$'s and the dual variable update remains the same. Convergence guarantees of the resultant two block variant of Algorithm 1 are described in Boyd

et al. (2011). Note that update (16) can be performed by using block coordinate descent (see e.g., Bertsekas (1999) and Friedman et al. (2007)) on the blocks $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$, which is essentially equivalent to performing updates (7) and (8) sequentially till convergence.

If we collapse all the variables $(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n, \boldsymbol{\theta}, \boldsymbol{\eta})$ into one block $\boldsymbol{\zeta}$ (say) then Problem (5) may be thought of as an augmented Lagrangian in the variable $\boldsymbol{\zeta}$, for which one may apply the augmented Lagrangian Method of Multipliers — a method for which convergence is relatively well understood (Bertsekas (1999)); also see the recent work of Aybat and Iyengar (2012). We briefly describe this method below.

Algorithm 2

1. Update: $\boldsymbol{\zeta}^{(k+1)} := \left(\boldsymbol{\xi}_1^{(k+1)}, \ldots, \boldsymbol{\xi}_n^{(k+1)}, \boldsymbol{\theta}^{(k+1)}, \boldsymbol{\eta}^{(k+1)} \right)$ as:

$$\boldsymbol{\zeta}^{(k+1)} \in \underset{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n, \boldsymbol{\theta}, \boldsymbol{\eta}}{\arg\min} \, \mathcal{L}_\rho(\boldsymbol{\zeta}; \boldsymbol{\nu}^{(k)}) \, \, s.t. \, \, \eta_{ij} \leq 0, \, \forall \, i, j = 1, \ldots, n. \tag{17}$$

2. Update: $\boldsymbol{\nu}^{k+1}$ as in (9).

We make some remarks about Algorithm 2 below.

• The *joint* optimization (17) with respect to $\boldsymbol{\zeta}$ can be done via block coordinate descent by updating $\boldsymbol{\xi}_i$'s, $\boldsymbol{\theta}$ and $\boldsymbol{\eta}$ sequentially as per the update rules (6), (7) and (8).

• The bottleneck in Algorithm 2 is the optimization Problem (17), since it needs to be performed for every iteration k. The problems, however, can be warm-started by using the estimates from the previous iteration for the current iteration. Since Algorithm 1 has excellent practical performance, i.e., the block updates are fairly cheap to carry out, we found it useful to initialize Algorithm 2 with a solution obtained from Algorithm 1. This hybrid version enjoys the convergence guarantees of Algorithm 2 as well as the good practical performance of Algorithm 1; see Section 6 where our experimental results are described.

• The step-size ρ can be updated *dynamically* and the optimization subproblems (17) can be solved *inexactly* such that the overall algorithm converges to an ϵ-optimal and ϵ-feasible solution in $O(\frac{1}{\epsilon})$ operations, as shown in Aybat and Iyengar (2012).

3   Smoothing Non-smooth Convex estimators

For simplicity of notation, in this section, we will write $\hat{\phi}_n$ as $\hat{\phi}$. Problem (2) describes a method to estimate the unknown convex regression function at the given covariate values $\mathbf{X}_i$'s. Although (3) describes a way of extending it to the whole of $\Re^d$, the obtained estimator is neither smooth nor unique (as $\hat{\boldsymbol{\xi}}_j$'s are not unique). While there are several ways in which the LSE can be defined beyond the data points, we briefly describe below a natural alternative that produces a unique estimator.

Let $\mathcal{X} := \{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$ and let $\text{Conv}(A)$ denote the convex hull of a set A. We can take the nonparametric LSE to be the function $\check{\phi} : \Re^d \to \Re$ defined by

$$\check{\phi}(\mathbf{x}) := \inf \left\{ \sum_{k=1}^{n} \alpha_k \hat{\theta}_k : \sum_{k=1}^{n} \alpha_k = 1, \ \sum_{k=1}^{n} \alpha_k \mathbf{X}_k = \mathbf{x}, \ \alpha_k \geq 0 \ \forall \ k = 1, \ldots, n \right\} \quad (18)$$

for any $\mathbf{x} \in \Re^d$. Here we take the convention that $\inf(\emptyset) = +\infty$. The function $\check{\phi}$ is well-defined and is finite on $\text{Conv}(\mathcal{X})$. In fact, $\check{\phi}$ is a polyhedral convex function (i.e., a convex function whose epigraph is a polyhedron; see Seijo and Sen (2011)). It is easy to see that $\check{\phi}$ is unique (as it only depends on the θ_k's); in fact, $\check{\phi}$ is the largest convex function satisfying the constraints $\check{\phi}(\mathbf{X}_k) = \theta_k$, for $k = 1, \ldots, n$; see Seijo and Sen (2011).

We describe below a novel approach to obtaining smooth approximations to the convex regression estimators $\hat{\phi}$ and $\check{\phi}$ with provable theoretical bounds on the quality of approximation. Our approach is different from the usual methods in nonparametric statistics to obtain smooth estimators under known shape restrictions; see e.g., Mukerjee (1988), Mammen (1991), Birke and Dette (2007), Aguilera et al. (2011), Du et al. (2013) and the references therein.

We propose a method of finding a smooth approximation ϕ^{Sm} to the piecewise affine LSE $\hat{\phi}$ (or $\check{\phi}$) with the following properties:

1. ϕ^{Sm} is differentiable and its gradient $\nabla \phi^{\text{Sm}}(\cdot)$ is Lipschitz continuous with parameter $\mathcal{L}$, i.e., $\|\nabla \phi^{\text{Sm}}(\mathbf{x}_1) - \nabla \phi^{\text{Sm}}(\mathbf{x}_2)\|_2 \leq \mathcal{L} \|\mathbf{x}_1 - \mathbf{x}_2\|_2$ for $\mathbf{x}_1, \mathbf{x}_2 \in \Re^d$.

2. ϕ^{Sm} is uniformly close to $\hat{\phi}$ in the following sense: $\sup_{\mathbf{x}} |\phi^{\text{Sm}}(\mathbf{x}) - \hat{\phi}(\mathbf{x})| \leq \tau(\mathcal{L})$, where, $\tau(\mathcal{L})$ is a function of the global Lipschitz gradient parameter $\mathcal{L}$.

In the next section we briefly describe the general technique for smoothing non-smooth functions and then in Section 3.2 (and Section 3.3) we specialize to the case of $\hat{\phi}$ (and $\check{\phi}$).

3.1 Smoothing structured convex non-smooth functions

Preliminaries. We start with some notation. Consider $\Re^d$ with a norm $\|\cdot\|$ and denote its dual norm by $\|\mathbf{s}\|^* := \max_{\mathbf{z}:\|\mathbf{z}\|=1}\langle \mathbf{s}, \mathbf{z}\rangle$, where $\langle \cdot, \cdot \rangle$ denotes the usual inner product.

Definition 1. *We say that a function* $f : \Re^d \to \Re$ *is* smooth *with parameter* $\mathcal{L}$ *if it is continuously differentiable and its gradient is Lipschitz with parameter* $\mathcal{L}$, *i.e.,* $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq \mathcal{L}\|\mathbf{x} - \mathbf{y}\|$ *for all* $\mathbf{x}, \mathbf{y} \in \Re^d$.

The framework described below follows the methodology introduced by Nesterov (2005) who describes an elegant smoothing procedure for the minimization of non-smooth functions having favorable geometrical structures. For the sake of completeness we present the general framework below. We consider the spaces $\Re^q$ and $\Re^p$ with norms $\|\cdot\|_\#$ and $\|\cdot\|_\dagger$ respectively. For a matrix $A_{p\times q}$, define its matrix norm $\|A\|_{\#,\dagger}$ induced by the norms $\|\cdot\|_\#, \|\cdot\|_\dagger$ as

$$\|A\|_{\#,\dagger} := \max_{\mathbf{u},\mathbf{v}} \left\{ \langle A\mathbf{u}, \mathbf{v}\rangle : \|\mathbf{u}\|_\# = 1, \|\mathbf{v}\|_\dagger = 1, \ \mathbf{u} \in \Re^q, \mathbf{v} \in \Re^p \right\}$$

where $\langle \cdot, \cdot \rangle$ denotes the usual Euclidean inner product. It is easy to see that

$$\|A\|_{\#,\dagger} = \|A^\top\|_{\dagger,\#} = \max_{\mathbf{u}}\{\|A\mathbf{u}\|_\dagger^* : \|\mathbf{u}\|_\# = 1\} = \max_{\mathbf{v}}\{\|A^\top\mathbf{v}\|_\#^* : \|\mathbf{v}\|_\dagger = 1\}.$$

Let $Q \subset \Re^p$ be a closed convex set and $\rho(\cdot)$ be a proximity (prox) function (see e.g., Nesterov (2005, 2004)) of the set Q. We will assume that $\rho(\cdot)$ is continuously differentiable and strongly convex on Q (with respect to the norm $\|\cdot\|_\dagger$) with strong convexity parameter one, i.e.,

$$\rho(\mathbf{u}_1) \geq \rho(\mathbf{u}_2) + \langle \nabla\rho(\mathbf{u}_2), \mathbf{u}_2 - \mathbf{u}_1\rangle + \frac{1}{2}\|\mathbf{u}_2 - \mathbf{u}_1\|_\dagger^2$$

for any $\mathbf{u}_1, \mathbf{u}_2 \in Q$. In particular, this implies that if $\mathbf{w}_0 \in \arg\min_{\mathbf{w}\in Q} \rho(\mathbf{w})$ then for any $\mathbf{w} \in Q$, we have

$$\rho(\mathbf{w}) - \rho(\mathbf{w}_0) \geq \frac{1}{2}\|\mathbf{w} - \mathbf{w}_0\|_\dagger^2.$$

Without loss of generality we can take the prox(imity) center $\mathbf{w}_0$ to satisfy $\rho(\mathbf{w}_0) = 0$.

Consider a function $\gamma : \Re^q \times (0, \infty) \to \Re$ given by

$$\gamma(\mathbf{z}; \tau) := \max_{\mathbf{w}\in Q} \ \{\langle A\mathbf{z}, \mathbf{w}\rangle - \tau\rho(\mathbf{w})\} \tag{19}$$

where $\tau > 0$ is a regularization parameter. To motivate the reader, let us consider a

simple example with Q given by the unit ℓ_1-ball, i.e., $Q = \{\mathbf{w} : \|\mathbf{w}\|_1 \leq 1\}$. Note that if $\tau = 0$ and Q as described above, $\gamma(\mathbf{z}; 0) = \|A\mathbf{z}\|_\infty$ which is a non-smooth function. The non-smoothness arises precisely due to the fact that $\hat{\mathbf{w}} \in \arg\max_{\mathbf{w} \in Q} \langle A\mathbf{z}, \mathbf{w} \rangle$ is not unique. One can get rid of the non-smoothness by using the modification suggested in (19). Since the optimization problem in (19) involves the maximization of a strongly concave function over a closed convex set, its maximum is attained and is unique. Thus the presence of the perturbation term $\tau\rho(w)$ with $\tau > 0$ has interesting consequences — it makes the function $\mathbf{z} \mapsto \gamma(\mathbf{z}; \tau)$ smooth (as in Definition 1) and the amount of smoothness imparted via the regularization can be precisely quantified. In this vein, we have the following theorem.

Lemma 1. *For any fixed $\tau > 0$, the function $\gamma(\mathbf{z}; \tau)$, defined in (19), is differentiable in $\mathbf{z}$ and its gradient is given by[1] $\nabla_1 \gamma(\mathbf{z}; \tau) = A^\top \hat{\mathbf{w}}^\tau$, where $\hat{\mathbf{w}}^\tau \in \arg\max_{\mathbf{w} \in Q} \{ \langle A\mathbf{z}, \mathbf{w} \rangle - \tau\rho(\mathbf{w}) \}$. Furthermore, the gradient map $\mathbf{z} \mapsto \nabla_1 \gamma(\mathbf{z}; \tau)$ is Lipschitz continuous with parameter $\frac{\|A\|_{\#,\dagger}^2}{\tau}$.*

We also have the following bounds describing how close $\gamma(\mathbf{z}; \tau)$ is to the unperturbed function $\gamma(\mathbf{z}; 0)$:

$$\gamma(\mathbf{z}; \tau) \geq \sup_{\mathbf{w} \in Q} \langle A\mathbf{z}, \mathbf{w} \rangle - \tau \sup_{\mathbf{w} \in Q} \rho(\mathbf{w}) = \gamma(\mathbf{z}; 0) - \tau \sup_{\mathbf{w} \in Q} \rho(\mathbf{w}), \tag{20}$$

and

$$\gamma(\mathbf{z}; \tau) = \sup_{\mathbf{w} \in Q} \{ \langle A\mathbf{z}, \mathbf{w} \rangle - \tau\rho(\mathbf{w}) \} \leq \sup_{\mathbf{w} \in Q} \langle A\mathbf{z}, \mathbf{w} \rangle = \gamma(\mathbf{z}; 0), \tag{21}$$

where the last inequality follows as a consequence of the non-negativity of the prox function.

Lemma 2. *For any $\tau \geq 0$, the perturbation $\gamma(\mathbf{z}; \tau)$ of $\gamma(\mathbf{z}; 0)$ satisfies the following uniform bound over $\mathbf{z}$:*

$$\gamma(\mathbf{z}; 0) - \tau \sup_{\mathbf{w} \in Q} \rho(\mathbf{w}) \leq \gamma(\mathbf{z}; \tau) \leq \gamma(\mathbf{z}; 0). \tag{22}$$

We present below a summary of the smoothing procedure described above:

- The function $\mathbf{z} \mapsto \gamma(\mathbf{z}; 0)$ may be non-smooth in $\mathbf{z}$. For $\tau > 0$, $\mathbf{z} \mapsto \gamma(\mathbf{z}; \tau)$ is smooth (by Theorem 1), convex and has Lipschitz continuous gradient with parameter $L = O(1/\tau)$.

---

[1] $\nabla_1 \gamma(\mathbf{z}; \tau)$ refers to the partial derivative of $\gamma(\mathbf{z}; \tau)$ with respect to $\mathbf{z}$.

- The smooth function $\gamma(\mathbf{z}; \tau)$ serves as a uniform approximation to $\gamma(\mathbf{z}; 0)$; the quality of approximation is quantified in Lemma 2. The approximation error is (upper) bounded by $\tau \sup_{\mathbf{w} \in Q} \rho(\mathbf{w})$ — a quantity that depends on Q and the choice of the function $\rho(\cdot)$.

- For a given $\rho(\cdot)$ and Q, the smoothness of the function $\mathbf{z} \mapsto \gamma(\mathbf{z}; \tau)$ is $O(1/\tau)$, which is inversely related to the order of approximation error, given by, $O(\tau)$.

3.2 Smooth post-processing of the convex LSE

We will use the above framework to smooth the convex function estimators. Consider first, the piecewise affine estimator $\hat{\phi}$. For $\mathbf{x} \in \Re^d$, we can always represent $\hat{\phi}$ as defined in (3) as:

$$\hat{\phi}(\mathbf{x}) = \max\left\{ \mathbf{a}_1^\top \mathbf{x} + b_1, \ldots, \mathbf{a}_m^\top \mathbf{x} + b_m \right\}. \tag{23}$$

The special case of $\hat{\phi}$ as defined in (3) can be expressed with $\mathbf{a}_i, b_i$'s, for $i = 1, \ldots, m$, with $m = n$, chosen as:

$$\mathbf{a}_i = \hat{\boldsymbol{\xi}}_i, \quad b_i = \hat{\theta}_i - \langle \hat{\boldsymbol{\xi}}_i, \mathbf{X}_i \rangle, \quad i = 1, \ldots, m. \tag{24}$$

Observe that $\hat{\phi}$ admits the following pointwise representation:

$$\hat{\phi}(\mathbf{x}) = \max_{\mathbf{w}} \sum_{i=1}^m w_i \left(\mathbf{a}_i^\top \mathbf{x} + b_i \right) \quad s.t. \sum_{i=1}^m w_i = 1, w_i \geq 0, i = 1, \ldots, m. \tag{25}$$

Let us denote the unit simplex appearing in the constraint set of the above optimization problem by $\Delta_m := \{\mathbf{w} : \sum_i w_i = 1, \ w_i \geq 0, i = 1, \ldots, m\}$. Observe that the non-smoothness in the function $\hat{\phi}(\cdot)$ arises due to the non-uniqueness of $\hat{\mathbf{w}}$ where $\hat{\mathbf{w}}$ is a maximizer of the linear optimization problem appearing in (25).

Following the framework developed in the previous section, it is easy to see that $\gamma(\mathbf{x}; 0)$, as defined in (19), is indeed the function $\hat{\phi}(\mathbf{x})$ and Q is Δ_m (with a suitable renaming of variables and adjusted dimensions). The smooth approximation $\tilde{\phi}$ of the non-smooth function $\hat{\phi}$ is thus obtained by the construction (19):

$$
\begin{aligned}
\tilde{\phi}(\mathbf{x}; \tau) = \quad &\max_{\mathbf{w}} \quad \sum_{i=1}^m w_i \left(\mathbf{a}_i^\top \mathbf{x} + b_i \right) - \tau \rho(\mathbf{w}) \\
&s.t. \quad \sum_{i=1}^m w_i = 1, w_i \geq 0, i = 1, \ldots, m,
\end{aligned}
\tag{26}
$$

where $\rho(\mathbf{w})$ is a prox function on the m dimensional unit simplex. In the following

we examine the consequences of using two different choices for the prox function.

3.2.1 Smoothing via the squared error prox function

A natural choice for smoothing comes from the squared error prox function, i.e., $\rho(\mathbf{w}) = \frac{1}{2}\left\|\mathbf{w} - \frac{1}{m}\mathbf{1}\right\|_2^2$ for which both norms $\|\cdot\|_\dagger, \|\cdot\|_\#$ (as appearing in Section 3.1) are taken to be the standard Euclidean norms. The smoothed approximation $\tilde{\phi}(\mathbf{x};\tau)$ can be obtained by solving problem (26) with the squared error prox function. The optimization problem in (26) is equivalent to the following convex program:

$$\operatorname*{minimize}_{\mathbf{w}} \; \left\{ \tfrac{1}{2}\sum_{i=1}^{m} w_i^2 - \sum_{i=1}^{m} w_i \tilde{c}_i \right\} \quad s.t. \; \sum_{i=1}^{m} w_i = 1, w_i \geq 0, i = 1, \ldots, m,$$

where $\tilde{c}_i = (a_i^\top \mathbf{x} + b_i)/\tau - 1/m$, for all $i = 1, \ldots, m$. The above problem is exactly equivalent to the Euclidean projection of $\tilde{\mathbf{c}} := (\tilde{c}_1, \ldots, \tilde{c}_m)$ onto the unit m dimensional simplex. Though there does not exist a closed form expression for this projection operation, it can be computed quite efficiently with complexity $O(m \log m)$ requiring a sorting operation on m numbers; see Michelot (1986). The approximation error associated with this smooth approximation $\tilde{\phi}(\mathbf{x};\tau)$ of $\hat{\phi}(\mathbf{x})$ is given by (see Lemma 2)

$$\sup_{\mathbf{x}\in\Re^d} |\tilde{\phi}(\mathbf{x};\tau) - \hat{\phi}(\mathbf{x})| \leq \tau \sup_{\mathbf{w}\in\Delta_m} \frac{1}{2}\|\mathbf{w} - \mathbf{1}/m\|_2^2 = \tau(1 - 1/m) \tag{27}$$

and the Lipschitz constant of the gradient is $\|A\|_{2,2}^2/\tau = \lambda_{\max}(A^\top A)/\tau$, since $\|A\|_{2,2}^2 = \lambda_{\max}(A^\top A)$, the maximum eigenvalue of $A^\top A$, where A is the $m \times (d+1)$ matrix whose i'th row is $(b_i, a_i^\top)$. Suppose that one seeks to obtain a smooth approximation $\tilde{\phi}(\mathbf{x};\tau)$ to $\hat{\phi}(\mathbf{x})$, with approximation error given by ϵ. This leads to a choice of τ that results in a bound of the Lipschitz constant of the gradient by $\lambda_{\max}(A^\top A)(m-1)/(m\epsilon)$.

3.2.2 Smoothing via the Entropy prox function

Let us next consider the entropy prox function on the unit simplex, i.e., $\rho(\mathbf{w}) = \sum_{i=1}^{m} w_i \log(w_i) + \log m$ and let $\|\cdot\|_\dagger$ and $\|\cdot\|_\#$ be the ℓ_1-norm in $\Re^m$. This prox function is strongly convex with respect to the standard ℓ_1-norm. For a simple proof of this fact, note that for any $\mathbf{h} \in \Re^m$ we have $\langle \nabla^2 \rho(\mathbf{w})\mathbf{h}, \mathbf{h} \rangle = \sum_i h_i^2/w_i$. By the Cauchy-Schwarz inequality it follows that $\left(\sum_i h_i^2/w_i\right)\left(\sum_i w_i\right) \geq \left(\sum_i |h_i|\right)^2$ which implies strong convexity of the entropy prox function $\rho(\cdot)$ with respect to the ℓ_1-norm.
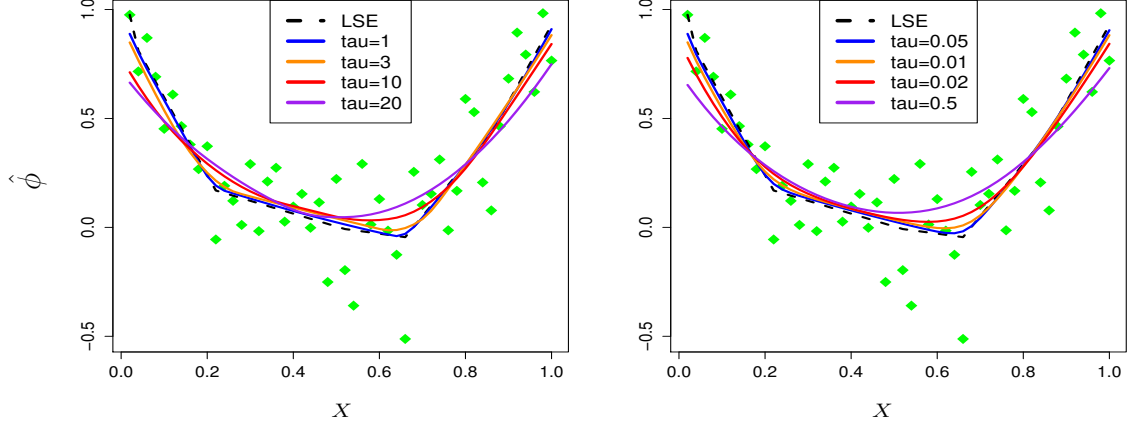
Figure 1: Plots of the data points and the convex LSE $\hat{\phi}$ with the bias corrected smoothed estimators for four different choices to τ using the squared error prox function (left panel) and entropy prox function (right panel).

Furthermore, it can be shown that for this choice of prox function

$$\sup_{\mathbf{x} \in \Re^d} |\tilde{\phi}(\mathbf{x}; \tau) - \hat{\phi}(\mathbf{x})| \leq \tau \sup_{\mathbf{w} \in \Delta_m} \rho(\mathbf{w}) = \tau \log m \tag{28}$$

and the Lipschitz constant of the gradient is given by $\|A\|_{1,1}^2/\tau = (\max_{i,j} |A_{ij}|)^2/\tau$. Thus, for an approximation error budget of ϵ, the corresponding Lipschitz constant of the gradient is given by $(\max_{i,j} |A_{ij}|)^2 \log m/\epsilon$. For the entropy prox function, there is a simple analytic expression for the smoothed approximation $\tilde{\phi}(\cdot; \tau)$ of $\hat{\phi}(\cdot)$, given by

$$
\begin{aligned}
\tilde{\phi}(\mathbf{x}; \tau) &= \sup_{\mathbf{w} \in \Delta_m} \left\{ \sum_{i=1}^m w_i \left(\mathbf{a}_i^\top \mathbf{x} + b_i \right) - \tau \left(\sum_{i=1}^m w_i \log(w_i) + \log m \right) \right\} \\
&= \tau \log \left(\sum_{i=1}^m \exp \left(\frac{\mathbf{a}_i^\top \mathbf{x} + b_i}{\tau} \right) \right) - \tau \log m.
\end{aligned}
$$

Figure 1 shows the smoothed convex estimators obtained from the two procedures (after a bias correction, described in Section 3.4) for different values of τ when $n = 50$ and $d = 1$. We see that both smoothing methods yield similar results and can produce estimators with varying degrees of smoothness. A similar phenomenon is observed when d exceeds 1.

The results described in Section 3.2.1 and Section 3.2.2 lead to the following theorem.

16

Theorem 1. *Let $\tilde{\phi}(\mathbf{x}; \tau)$, as defined via* (26), *be a smoothed approximation of the piecewise affine convex LSE fit $\hat{\phi}(\mathbf{x})$ as in* (23) *such that the following holds:*

$$\sup_{\mathbf{x}\in\Re^d} |\tilde{\phi}(\mathbf{x}; \tau) - \hat{\phi}(\mathbf{x})| \leq \epsilon,$$

for some fixed pre-specified $\epsilon > 0$. Then, we have the following:

(a) *if $\rho(\mathbf{w})$ is the squared error prox function (see Section* 3.2.1*), then $\tilde{\phi}(\mathbf{x}; \tau)$ has a Lipschitz continuous gradient with $\mathcal{L} = \lambda_{\max}(A^\top A)(m-1)/(m\epsilon)$.*

(b) *If $\rho(\mathbf{w})$ is the entropy prox function (see Section* 3.2.2*) then $\tilde{\phi}(\mathbf{x}; \tau)$ has a Lipschitz continuous gradient with $\mathcal{L} = \left(\max_{i,j} |A_{ij}|\right)^2 \log m/\epsilon$.*

Remark 1. *Note that there is a trade-off between the smoothness of the function $\tilde{\phi}(\mathbf{x}; \tau)$ and its accuracy in approximating $\hat{\phi}(\mathbf{x})$. Its smoothness, measured by the Lipschitz constant of its gradient is $O(\frac{1}{\epsilon})$ and the approximation error is $O(\epsilon)$.*

3.3   Smoothing for scheme (18)

We will now consider smoothing schemes for the interpolant which is given by the linear program (LP) in (18) with optimization variables $\alpha_1, \ldots, \alpha_n$. Observe that the smoothing scheme described above in (23) works for convex functions that admit a *max-like* representation. The function (18), on the other hand, admits a *min-like* representation, hence the smoothing scheme of Section 3.2 does not directly apply. To circumvent this we consider the dual representation of the LP appearing in (18), which is given by the following LP:

$$\check{\phi}(\mathbf{x}) = \max [-\mu - \langle \boldsymbol{\zeta}, \mathbf{x} \rangle] \quad s.t. \ \hat{\theta}_i + \mu + \langle \boldsymbol{\zeta}, \mathbf{X}_i \rangle \geq 0, \quad i = 1, \ldots, n, \quad (29)$$

where the optimization variables are $\mu \in \Re, \boldsymbol{\zeta} \in \Re^d$. Consider a modified version of the estimator (29) obtained by adding a strongly convex prox function to the objective in problem (29). This leads to the following convex program:

$$\check{\phi}^{\mathrm{Sm}}(\mathbf{x}) = \max [-\mu - \langle \boldsymbol{\zeta}, \mathbf{x} \rangle - \tau \rho(\mu, \boldsymbol{\zeta})] \quad s.t. \ \hat{\theta}_i + \mu + \langle \boldsymbol{\zeta}, \mathbf{X}_i \rangle \geq 0, \quad i = 1, \ldots, n,$$
$$(30)$$

with variables $\mu \in \Re, \boldsymbol{\zeta} \in \Re^d$. For simplicity, consider the Euclidean prox function, i.e., $\rho(\mu, \boldsymbol{\zeta}) = \frac{1}{2}(\mu^2 + \|\boldsymbol{\zeta}\|_2^2)$ with the respective spaces endowed with the standard Euclidean $\|\cdot\|_2$-norm. It follows from Theorem 1 that the function $\check{\phi}^{\mathrm{Sm}}(\mathbf{x})$ is differentiable in $\mathbf{x}$ with Lipschitz continuous gradient $-\hat{\boldsymbol{\zeta}}^\tau$, where $(\hat{\mu}^\tau, \hat{\boldsymbol{\zeta}}^\tau)$ is the unique optimal solution to problem (30).

3.4 Bias correction

Although (27) and (28) provide a two-sided bound for the maximal deviation between $\hat{\phi}$ (or $\check{\phi}$) and the smoothed estimator $\tilde{\phi}$ (or $\check{\phi}^{\mathrm{Sm}}$), Lemma 2 (see (22)) shows that the smoothed estimator is always less than $\hat{\phi}$. This bias of the smoothed estimator can be easily corrected for. It can be easily shown that the fitted $\hat{\boldsymbol{\theta}}$ always has the same average as the sample mean (see e.g., Seijo and Sen (2011, Lemma 2.4)), i.e., $\sum_{i=1}^{n} \hat{\theta}_i = \sum_{i=1}^{n} Y_i$. Let $\tilde{\boldsymbol{\theta}} = (\tilde{\theta}_1, \ldots, \tilde{\theta}_n)$ denote the vector of values of the smoothed estimator at the $\mathbf{X}_i$'s. It is natural to enforce that the bias corrected smoothed estimator should also have the same mean as the sample mean. This leads to the bias corrected smoothed estimator defined as $\tilde{\phi}_{\mathrm{BC}}(\mathbf{x}) := \tilde{\phi}(\mathbf{x}) + \frac{1}{n}\sum_{i=1}^{n}(\hat{\theta}_i - \tilde{\theta}_i)$. We can similarly define the bias corrected smoothed estimator obtained using $\check{\phi}^{\mathrm{Sm}}$. Figure 1 shows the bias corrected estimator along with the LSE when $n = 100$.

4 Lipschitz Convex Regression

The LSE described in (2) suffers from over-fitting, especially near the boundary of the convex hull of the design points $\mathbf{X}_i$'s. The norms of the fitted subgradients $\hat{\boldsymbol{\xi}}_i$'s near the boundary can become arbitrarily large as the sample size grows and there can be a large proportion of data points near the boundary, for $d > 1$. This, in turn, can deteriorate the overall performance of the LSE. In fact, even when $d = 1$ it is expected that the convex LSE will be inconsistent at the boundary; see Balabdaoui (2007) for a proof in the context of density estimation.

As a remedy to this over-fitting problem we propose LS minimization over the class of convex functions that are uniformly Lipschitz with a known bound. For a convex function $\psi : \mathfrak{X} \to \mathfrak{R}$, let us denote by $\partial\psi(\mathbf{x})$ the subdifferential (set of all subgradients) at $\mathbf{x} \in \mathfrak{X}$, and by $\|\partial\psi(\mathbf{x})\|$ the maximum $\|\cdot\|_2$-norm of vectors in $\partial\psi(\mathbf{x})$, i.e., $\|\partial\psi(\mathbf{x})\| := \sup_{\boldsymbol{\zeta}\in\partial\psi(\mathbf{x})} \|\boldsymbol{\zeta}\|_2$. For $L > 0$, consider the class $\mathcal{C}_L$ of convex functions with Lipschitz norm bounded by L, i.e.,

$$\mathcal{C}_L := \left\{ \psi : \mathfrak{X} \to \mathfrak{R} \mid \psi \text{ is convex}, \ \sup_{\mathbf{x}\in\mathfrak{X}} \|\partial\psi(\mathbf{x})\| \le L \right\}. \tag{31}$$

Let $\hat{\phi}_{n,L}$ denote the LSE when minimizing the sum of squared errors over the class $\mathcal{C}_L$, i.e.,

$$\hat{\phi}_{n,L} \in \arg\min_{\psi\in\mathcal{C}_L} \sum_{i=1}^{n} (Y_i - \psi(\mathbf{X}_i))^2. \tag{32}$$

The above problem is an infinite dimensional optimization problem. Fortunately, as before, the solution to the above problem can be obtained by solving the following finite dimensional convex optimization problem:

$$
\begin{aligned}
\underset{\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_n;\boldsymbol{\theta}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{Y} - \boldsymbol{\theta}\|_2^2 \\
\text{s.t.} \quad & \theta_j + \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \rangle \leq \theta_i;\ i,j = 1,\ldots,n; \\
& \|\boldsymbol{\xi}_j\|_2 \leq L, \quad j = 1,\ldots,n.
\end{aligned}
\tag{33}
$$

To see that Problems (32) and (33) are equivalent, it suffices to consider a solution to Problem (33) and extend it to a member of the set $\mathcal{C}_L$ using (3). Such an extension does not change the loss function and satisfies the feasibility condition of both problems (32) and (33).

We present the following result (proved in Section A.5) concerning the asymptotic rate of convergence of the convex Lipschitz LSE (i.e., Problem (33)).

Theorem 2. *Consider observations $\{(\mathbf{X}_i, Y_i) : i = 1,\ldots,n\}$ such that $Y_i = \phi(\mathbf{X}_i)+\epsilon_i$, where $\phi : \Re^d \to \Re$ is an unknown convex function. We assume that (i) the support of $\mathbf{X}$ is $\mathfrak{X} = [0,1]^d$; (ii) $\phi \in \mathcal{C}_{L_0}$ for some $L_0 > 0$; (iii) the $\mathbf{X}_i \in \mathfrak{X}$'s are fixed constants; and (iv) ϵ_i's are independent mean zero sub-Gaussian errors (i.e., there exists $\sigma^2 > 0$ such that for every $t \in \Re$ one has $\mathbb{E}[e^{t\epsilon_1}] \leq e^{\sigma^2 t^2/2}$). Given data from such a model, we have for any $L > L_0$,*

$$
\frac{1}{n}\sum_{i=1}^n (\hat{\phi}_{n,L}(\mathbf{X}_i) - \phi(\mathbf{X}_i))^2 = O_{\mathbb{P}}(r_n),
\tag{34}
$$

where

$$
r_n = \begin{cases} n^{-2/(d+4)} & \text{if } d = 1,2,3, \\ n^{-1/4}(\log n)^{1/2} & \text{if } d = 4, \\ n^{-1/d} & \text{if } d \geq 5. \end{cases}
\tag{35}
$$

Remark 2. *The above result follows from known metric entropy bounds on the class of all convex functions that are uniformly bounded and uniformly Lipschitz (under the uniform metric) and the theory on the rates of convergence of LSEs; see e.g., Van de Geer (2000, Theorem 9.1). Note that the class of all convex functions is much larger (in fact, the set is not totally bounded) and thus finding the rate of convergence of the convex LSE (see (3)) is a much harder problem; in fact it is still an open problem.*

Remark 3. *The assumption that $\mathfrak{X} = [0,1]^d$ can be extended to any compact subset of $\Re^d$.*

Remark 4. *Recently, during the preparation of the manuscript, we became aware of the work of Balázs et al. (2015) and Lim (2014), where LSEs obtained over different sub-classes of convex functions are proposed and studied. Balázs et al. (2015) propose Lipschitz convex regression (where instead of the $\|\cdot\|_2$-norm on the subgradients, the authors study the $\|\cdot\|_\infty$-norm) with an additional boundedness constraint and study the rates of convergence of the obtained LSE. Lim (2014) studies $\|\cdot\|_\infty$-Lipschitz convex regression and the main result in the paper is similar to Theorem 2. We, of course, deal with a different formulation as we constrain the $\|\cdot\|_2$-norm of the subgradients. Moreover, the proof of Theorem 2 can be mimicked to give a shorter and simpler proof of the main result in Lim (2014).*

The following lemma (proved in Section A.1) shows that the estimator obtained upon applying the smoothing method to the estimator obtained from Problem (33) lies in $\mathcal{C}_L$.

Lemma 3. *Let $\tilde{\phi}(\mathbf{x}; \tau)$ be the smoothed estimator obtained via the scheme in Section 3.2 for estimator (23) with $\hat{\theta}_i, \hat{\boldsymbol{\xi}}_i$'s obtained as solutions to Problem (33). Then $\|\nabla \tilde{\phi}(\mathbf{x}; \tau)\|_2 \leq L$.*

4.1 Algorithm for Problem (33)

The convex optimization problem (33) can be solved by interior point methods. However, as in the case of Problem (2) off-the-shelf interior point methods have difficulty scaling to large n. In this vein, we propose a simple variant of the ADMM algorithmic framework described for Problem (2) that can be applied to solve Problem (33). To see this we consider the following equivalent representation of Problem (33):

$$
\begin{aligned}
\operatorname*{minimize}_{\boldsymbol{\xi}_1,\dots,\boldsymbol{\xi}_n;\boldsymbol{\theta};\boldsymbol{\eta}} \quad & \frac{1}{2}\|\mathbf{Y} - \boldsymbol{\theta}\|_2^2 \\
s.t. \quad & \eta_{ij} = \theta_j + \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \rangle - \theta_i; \\
& \eta_{ij} \leq 0; \ i = 1,\dots,n, \ j = 1,\dots,n; \\
& \|\boldsymbol{\xi}_j\|_2 \leq L, \quad j = 1,\dots,n;
\end{aligned}
\tag{36}
$$

and the corresponding augmented Lagrangian similar to (4). We use an ADMM algorithm with a similar block splitting strategy as in Algorithm 1 — the main difference with Algorithm 1 being update (6) where, we need to solve:

Step 1. Update the subgradients $(\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_n)$:

$$
(\boldsymbol{\xi}_1^{(k+1)}, \dots, \boldsymbol{\xi}_n^{(k+1)}) \in \operatorname*{arg\,min}_{\boldsymbol{\xi}_1,\dots,\boldsymbol{\xi}_n : \|\boldsymbol{\xi}_j\|_2 \leq L, \forall j} \mathcal{L}_\rho((\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_n; \boldsymbol{\theta}^{(k)}; \boldsymbol{\eta}^{(k)}); \boldsymbol{\nu}^{(k)}).
\tag{37}
$$

The above problem is a QP in nd variables with nd constraints. Note that the optimization problem is separable in the variables $\boldsymbol{\xi}_j$'s, for $j = 1, \ldots, n$. Thus solving Problem (37) is equivalent to solving for each j the following constrained variant of the convex problem (10):

$$\underset{\boldsymbol{\xi}_j}{\text{minimize}} \quad \sum_{i=1}^{n} \left(\bar{\eta}_{ij} - \langle \Delta_{ij}, \boldsymbol{\xi}_j \rangle \right)^2 \quad s.t. \quad \|\boldsymbol{\xi}_j\|_2 \le L. \tag{38}$$

The above problem, unlike Problem (6), does not admit a closed form solution. However, Problem (38) is a Second Order Cone Program (SOCP) with d variables and since d is typically quite small (e.g., 10 or 20), solving it is quite cheap. Problem (38) can thus be solved by using an off-the-shelf interior point method. It can also be solved via specialized algorithms — we develop our own algorithm for Problem (38) that we describe in Section A.2.
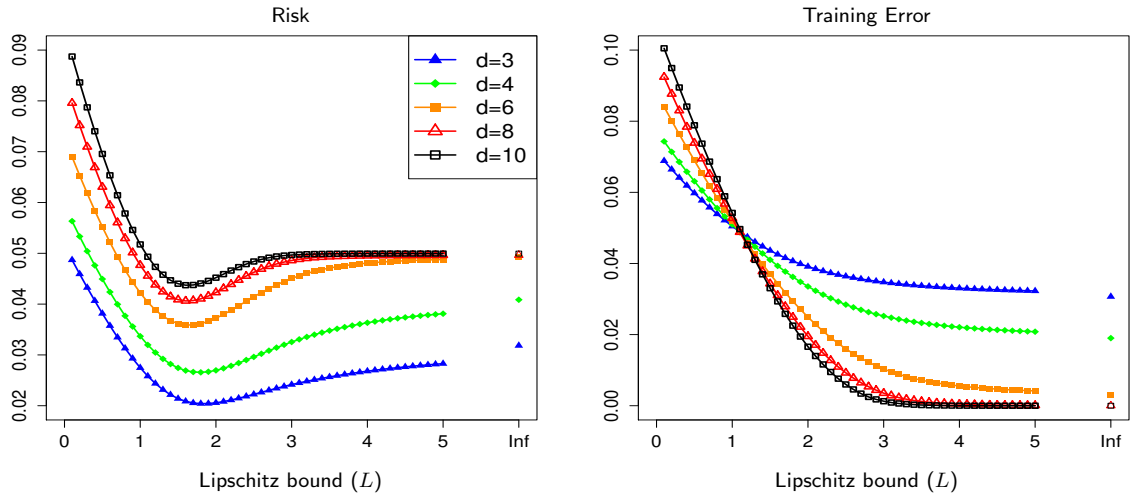

Figure 2: [Left panel]: the simulated risk of the Lipschitz convex estimator as the Lipschitz bound L varies (L = Inf gives the usual convex LSE) for 5 different dimension values (d). [Right panel]: the training error as the Lipschitz bound L varies, for the same examples appearing in the left panel.

In Figure 2 we show the performance of Lipschitz convex regression when $n = 100$, $\phi(\mathbf{x}) = \|\mathbf{x}\|_2^2$ and d varies in $\{3, 4, 6, 8, 10\}$. The left panel illustrates that a proper choice of L can lead to substantial reduction in the risk of the estimator (as measured by $\mathbb{E}_\phi \sum_{i=1}^{n} (\hat{\phi}_{n,L}(\mathbf{X}_i) - \phi(\mathbf{X}_i))^2$) in estimating ϕ. The right panel shows that as the dimension d grows, the usual convex LSE overfits the data (for $d = 10$ the training error is essentially 0).

To choose an optimal tuning parameter L, the Lipschitz constant that minimizes

21

the risk, we advocate the use of cross-validation; see e.g., Hastie et al. (2009). In our extensive simulation studies we observed that tenfold cross-validation works quite well. We use the "one-standard error" rule in cross-validation where we choose the smallest L whose error is no more than one standard error above the error of the best Lipschitz parameter.

5 Convex Functions with Coordinate-wise Monotonicity

In this section we consider a variation of the problem proposed in (1), where the underlying convex regression function, restricted to each of its coordinates, is assumed to be monotone (i.e., increasing or decreasing). Estimation of such functions have a wide range of applications, especially in demand and production frontiers in economics; see e.g., Varian (1982), Varian (1984), Matzkin (1994), Yatchew (1998), Kuosmanen (2008) and the references therein.

To fix ideas, let us assume that the function to be estimated is known to be convex and non-decreasing coordinate-wise and we have data $\{(\mathbf{X}_i, Y_i) : i = 1, \ldots, n\}$. This leads to the following infinite dimensional optimization problem:

$$\underset{\psi}{\text{minimize}} \quad \sum_{i=1}^{n} (Y_i - \psi(\mathbf{X}_i))^2 \tag{39}$$

where the minimization is carried over all convex functions ψ that are non-decreasing in each of its coordinates, i.e., $\nabla_k \psi(\mathbf{x}) \geq 0$ for all $\mathbf{x}$ and for all $k = 1, \ldots, d$ (here $\nabla_k \psi(\mathbf{x})$ denotes the k'th coordinate of a subgradient of ψ at $\mathbf{x}$). Not surprisingly, this seemingly infinite dimensional optimization problem can also be cast as the following (finite dimensional) QP:

$$\begin{aligned}
\underset{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n; \boldsymbol{\theta}}{\text{minimize}} \quad & \frac{1}{2} \|\mathbf{Y} - \boldsymbol{\theta}\|_2^2 \\
s.t. \quad & \theta_j + \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \rangle \leq \theta_i; \quad i = 1, \ldots, n, \ j = 1, \ldots, n, \\
& \xi_{ji} \geq 0; \ j = 1, \ldots, n, \ i = 1, \ldots, d,
\end{aligned} \tag{40}$$

where the notations used above are the same as in Problem (2) with $\xi_{ji}, i = 1, \ldots, d$, being the coordinates of the subgradient $\boldsymbol{\xi}_j$. Note that the constraints $\xi_{ji} \geq 0$, for $j = 1, \ldots, n$, represent that the i'th coordinate of the subgradients of the function evaluated at the data points $\mathbf{X}_j$'s are non-negative. This is equivalent to the function

restricted to the i'th coordinate being non-decreasing. The above formulation resembles Problem (2) with the exception of the additional nd constraints $\xi_{ji} \geq 0$ for all i, j.

To see why Problems (40) and (39) are equivalent, observe that any solution $\{\boldsymbol{\xi}_i^*\}_{i=1}^n$ and $\boldsymbol{\theta}^*$ of Problem (40) can be extended to a convex function on $\Re^d$ by the rule (3). Note that $\hat{\phi}_n$ thus defined is convex in $\Re^d$. Any subgradient of the function $\hat{\phi}_n$ when restricted to the k'th coordinate is non-negative and hence the function $\hat{\phi}_n$ is non-decreasing in each coordinate. Furthermore, this function has the same loss function as the optimal objective value of Problem (40). Thus solving Problem (40) is equivalent to solving Problem (39).

We use an ADMM type algorithm to solve (40) in a similar manner as (4), by considering the following equivalent representation for (40):

$$
\begin{aligned}
\underset{\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_n;\boldsymbol{\theta};\boldsymbol{\eta}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{Y} - \boldsymbol{\theta}\|_2^2 \\
s.t. \quad & \eta_{ij} = \theta_j + \langle \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \rangle - \theta_i; \\
& \eta_{ij} \leq 0; \ i = 1, \ldots, n; \ j = 1, \ldots, n, \\
& \xi_{ji} \geq 0; \ j = 1, \ldots, n; \ i = 1, \ldots, d,
\end{aligned}
\tag{41}
$$

where $\boldsymbol{\eta} = ((\eta_{ij})) \in \Re^{n \times n}$ is a matrix with (i,j)'th entry η_{ij}. We then consider the augmented Lagrangian corresponding to the above formulation, similar to (5), and employ a multiple block version of ADMM along the lines of Algorithm 1. The main difference with Algorithm 1 is the update (6) — we now need to consider the following constrained QP:

Step 1. Update the subgradients $(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n)$:

$$
(\boldsymbol{\xi}_1^{(k+1)}, \ldots, \boldsymbol{\xi}_n^{(k+1)}) \in \underset{\boldsymbol{\xi}_1,\ldots,\boldsymbol{\xi}_n:\boldsymbol{\xi}_{ji}\geq 0, \ \forall i,j}{\arg\min} \mathcal{L}_\rho((\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_n; \boldsymbol{\theta}^{(k)}; \boldsymbol{\eta}^{(k)}); \boldsymbol{\nu}^{(k)}).
\tag{42}
$$

The above problem is a QP in nd variables with nd constraints. Note that the optimization problem is separable in the variables $\boldsymbol{\xi}_j$'s, for $j = 1, \ldots, n$. Thus solving Problem (42) is equivalent to solving for each j the following non-negative LS problem:

$$
\begin{aligned}
\underset{\boldsymbol{\xi}_j}{\text{minimize}} \quad & \boldsymbol{\xi}_j^\top \left(\sum_{i=1}^n \boldsymbol{\Delta}_{ij} \boldsymbol{\Delta}_{ij}^\top \right) \boldsymbol{\xi}_j - 2 \left\langle \sum_{i=1}^n \left(\frac{1}{\rho}\nu_{ij} + \eta_{ij} - (\theta_j - \theta_i) \right) \boldsymbol{\Delta}_{ij}, \boldsymbol{\xi}_j \right\rangle \\
s.t. \quad & \xi_{jk} \geq 0, \quad k = 1, \ldots, d.
\end{aligned}
\tag{43}
$$

Unlike Problem (6), Problem (43) does not admit a closed form solution. However,

Problem (43) is a QP with d variables and since d is typically small, solving it is quite cheap. Problem (43) can be solved by using an off-the-shelf interior point method for each j. We however use our own implementation of one-at-a-time coordinate descent for the above optimization problem. The procedure is described in Section A.3.

Remark 5. *Note that the framework described above can easily accommodate other variants of (40). For example, if the convex function restricted to each of the coordinate directions is assumed to be non-increasing then one needs to use the constraints $\xi_{ij} \leq 0$ for all i, j — which can be addressed by using a minor variant of Problem (43). Similarly, it is also possible to accommodate the case where the function is assumed to be non-decreasing in some of the coordinates and non-increasing in some others.*

The following lemma (proved in Section A.4) shows that our smoothing operation does not change the sign of the coordinate-wise subgradients of the function.

Lemma 4. *Let $\tilde{\phi}(\mathbf{x}; \tau)$ be the smoothed estimator obtained via the scheme in Section 3.2 for the estimator (23), with $\hat{\theta}_i, \hat{\boldsymbol{\xi}}_i$'s obtained as solutions to Problem (40). Then $\nabla_k \tilde{\phi}(\mathbf{x}; \tau) \geq 0$, for $k = 1, \ldots, d$, i.e., the smooth estimator is also coordinate-wise increasing.*

6 Numerical Experiments

Description of datasets

In all the synthetic examples below, we generate n samples as $y_i = \mu_i + \epsilon_i, i = 1, \ldots, n$, where $\mu_i := \phi(\mathbf{X}_i)$ is the value of a function μ evaluated at the i'th datapoint $\mathbf{X}_i \in \Re^d$. The errors ϵ_i's are assumed to be i.i.d. $N(0, \sigma^2)$, for $i = 1, \ldots, n$. We define the Signal to Noise Ratio (SNR) as SNR = $\text{Var}(\mu)/\text{Var}(\epsilon)$. The different examples that are studied are given below.

Example 1: Here, we took $n = 500, d = 2$ and the covariates were generated from a random Uniform ensemble with mean zero. We took $\phi(x) = \|x\|_2^2$ and the value of σ^2 was adjusted so that SNR = 3. Both the features and the response were mean-centered and standardized to have unit $\| \cdot \|_2$-norm before being *fed* into the solvers.

Example 2: This is similar to Example 1 with a larger problem size: $n = 1000, d = 10$.

Example 3: This is a real dataset with $n = 473$ and $d = 4$. The dataset, which appears in the paper Wang and Wang (2013), was downloaded from the link http://www.nber.org/data/nbprod2005.html. Here, we took the response as the total value
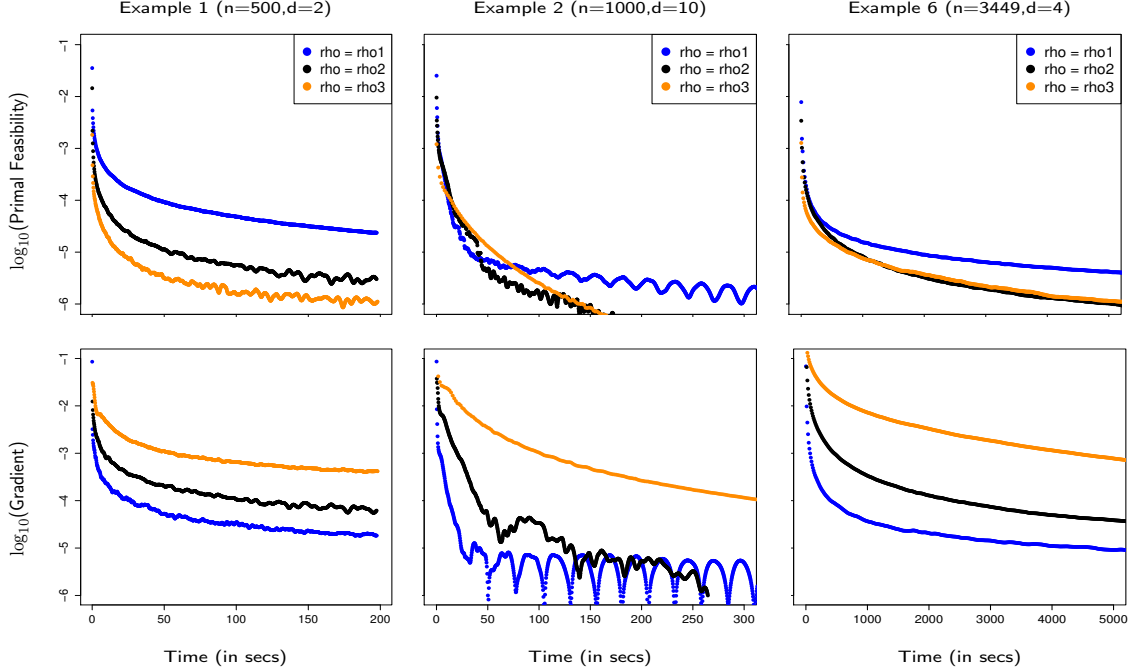
24

Figure 3: The evolution of Algorithm 1 with time, for three different examples: the right panel is a real-data example, the remaining are synthetic. The top panel shows the primal feasibility convergence with time and bottom panel shows the evolution of the $\| \cdot \|_2$-norm of the gradient with respect to $\boldsymbol{\theta}$. Three different ρ values, denoted by 'rho1', 'rho2', 'rho3', were taken to be $0.1/n, 1/n, 10/n$ respectively. Note that all the algorithms were allowed to run for a long time and hence the solutions obtained have high accuracy $\approx 10^{-6}$. The figures show that lower accuracy solutions are obtained faster, see also, Table 1.

of shipment. The four independent variables were: total real capital stock, production worker hours, number of non-production workers and number of production worker hours. Based on exploratory data-analysis, we took a log-transform of each of the covariates. We mean-centered and scaled each of the covariates and the response, so that they have unit $\| \cdot \|_2$-norm.

Example 4: In this example, $n = 558$ and $d = 3$. This dataset was taken from the link http://www.econ.kuleuven.ac.be/GME (see Verbeek (2008)) and contains production data for 569 Belgian firms, from the year 1996. The response was taken as the negative logarithm of the value added by a worker. The predictor variables were: amount of capital (in terms of total fixed assets at the end of 1995), labour and wages[2]. As in the other cases, both the features and response were mean-centered and standardized to have unit $\| \cdot \|_2$-norm.

Example 5: In this synthetic example, we took $n = 1000$ and $d = 5$, with SNR=3.

---

[2]Upon exploratory analysis, we found some possible outliers in the covariate-space, which were discarded via some simple pre-processing method.

| Example 2 ($n = 1000, d = 10$) | | | | | Example 3 ($n = 473, d = 4$) | | | |
Primal Feas Error	Gradient Error	Algorithm 2 Time (secs)	Algorithm 1 Time (secs)		Primal Feas Error	Gradient Error	Algorithm 2 Time (secs)	Algorithm 1 Time (secs)
1e-03	1e-02	2.043	2.240		1e-03	1e-02	0.523	0.535
1e-03	1e-03	15.444	20.224		1e-03	1e-03	3.413	3.537
1e-03	1e-04	28.546	40.457		1e-03	1e-04	17.034	17.290
1e-04	1e-02	14.146	14.373		1e-04	1e-02	1.870	1.935
1e-04	1e-03	15.444	20.224		1e-04	1e-03	3.413	3.537
1e-04	1e-04	28.546	40.457		1e-04	1e-04	17.034	17.290

| Example 4 ($n = 558, d = 3$) | | | | | Example 5 ($n = 1000, d = 5$) | | | |
Primal Feas Error	Gradient Error	Algorithm 2 Time (secs)	Algorithm 1 Time (secs)		Primal Feas Error	Gradient Error	Algorithm 2 Time (secs)	Algorithm 1 Time (secs)
1e-03	1e-02	0.988	0.976		1e-03	1e-02	2.474	2.486
1e-03	1e-03	16.116	16.074		1e-03	1e-03	62.338	62.493
1e-03	1e-04	143.181	103.888		1e-03	1e-04	1226.499	516.286
1e-04	1e-02	7.295	7.282		1e-04	1e-02	39.143	39.951
1e-04	1e-03	16.116	16.074		1e-04	1e-03	62.338	62.493
1e-04	1e-04	156.612	103.888		1e-04	1e-04	1226.499	516.286

| Example 6 ($n = 3449, d = 4$) | | | | | Example 7 ($n = 1500, d = 4$) | | | |
Primal Feas Error	Gradient Error	Algorithm 2 Time (secs)	Algorithm 1 Time (secs)		Primal Feas Error	Gradient Error	Algorithm 2 Time (secs)	Algorithm 1 Time (secs)
1e-03	1e-02	73.743	75.738		1e-03	1e-02	12.171	12.304
1e-03	1e-03	448.043	459.826		1e-03	1e-03	66.701	67.982
1e-03	1e-04	3445.433	2392.528		1e-03	1e-04	324.665	330.774
1e-04	1e-02	118.601	122.275		1e-04	1e-02	28.217	28.691
1e-04	1e-03	448.043	459.826		1e-04	1e-03	66.701	67.982
1e-04	1e-04	3445.433	2392.528		1e-04	1e-04	324.665	330.774

Table 1: Table showing the times (in secs) taken for Algorithm 1 and Algorithm 2 to reach solutions of different accuracy levels of the primal feasibility gap and the norm of the gradient, as defined in the text. For Algorithm 1 we took the choice $\rho = 1/n$. Observe that the performances of Algorithms 1 and 2 are quite similar in most instances.

Here, the underlying convex function was taken as: $\phi(x) = (5x_1 + 0.5x_2 + x_3)^2 + \sqrt{x_4^2 + x_5^2}$, where, x_i refers to the i'th feature, with $i = 1, \ldots, d$.

Example 6: This is a real-data example with $n = 3449$ and $d = 4$. This dataset, taken from the paper Mekaroonreung and Johnson (2012), was downloaded from http://ampd.epa.gov/ampd/. The response was the amount of heat input with the covariates corresponding to the amounts of emissions of S02, NOx, C02 (in tons) and the NOX rate. There were some samples with missing values that were removed from the dataset. We took a logarithmic transformation of the covariates and observed (based on exploratory analysis) that the relationship between the response and the individual covariates seemed to be modeled well via a convex fit. Both the response

and features were centered and scaled as in the aforementioned instances.

Example 7: This example is a smaller subset of the dataset in Example 6. Here we took the first 1500 samples giving us: $n = 1500, d = 5$.

Software Specifications: All our computations were performed in MATLAB, (R2014a (8.3.0.532) 64-bit (maci64)) which was interfaced with some of the matrix operations coded in C on a OS X 10.8.5 (12F45) operating system with a 3.4 GHz Intel Core i5 processor with 32 GB Ram, processor speed 1600 MHz and DDR3 SDRAM.

For all the examples above, we applied Algorithm 1. We observed that the algorithm converged in all the instances, which was verified by checking that the conditions of optimality (see (15)) were satisfied (approximately) up to algorithmic precision. The convergence speed was, however, found to be sensitive to the choice of ρ. We observed that after the data was standardized, i.e., features and responses set to have unit $\| \cdot \|_2$ norm, a value of ρ of the order of $1/n$ performed quite well. Figure 3 shows some of the results for different values of $\rho \in \{0.1/n, 1/n, 10/n\}$; additional examples are presented in the appendix, see Figure 4 in Section B.

Primal feasibility is measured by $\|\Gamma\|_F/n$, where, $\Gamma = ((\gamma_{ij}))$ with $\gamma_{ij} = \eta_{ij}^* - (\theta_i^* + \langle \Delta_{ij}, \xi_j^* \rangle - \theta_j^*)$, as defined in (15a) and $\|\Gamma\|_F$ denotes the Frobenius norm of Γ. The gradient condition with respect to $\boldsymbol{\theta}$ is measured by the $\| \cdot \|_2$-norm of the vector $(\boldsymbol{\theta}^* - \mathbf{Y}) - \mathbf{D}^\top \text{vec}(\boldsymbol{\nu}^*)$, as defined in (15c). These are the two metrics that have been considered in Figure 3 and Figure 4. Figure 3 shows that there is no clear best choice of ρ, but there is however, one systematic pattern: a small choice of ρ leads to faster changes in the objective value across iterations, but the primal feasibility goes to zero at a slower rate. Similarly, for a larger value of ρ we observe that the primal feasibility is (approximately) satisfied early on in the iterations, but it takes longer for the objective values to stabilize. As mentioned before, we typically found a choice of ρ of the order of $1/n$ to work quite well in our experiments.

In addition to Algorithm 1, we have also considered Algorithm 2 which is theoretically guaranteed to converge. Numerical results showing the performance of Algorithm 1 versus Algorithm 2 is presented in Table 1. We have observed in our experiments that Algorithm 2 has similar empirical behavior as Algorithm 1. In the implementation of Algorithm 2, we took a dynamically decreasing tolerance in primal feasibility and the gradient condition, following Aybat and Iyengar (2012). For Algorithm 2, the first 500 (outer) iterations (as many dual variable updates) were taken to be identical to Algorithm 1; after which each inner loop was executed till a tolerance of δ_i and this tolerance was made tighter via the schedule $\delta_i \leftarrow \tau \delta_i$, with

the starting value of $\delta_i = 10^{-1}$. Whenever the dual variable was updated, the value of ρ was increased by a factor $1/\tau$. We took $\tau = 0.9954$ and the algorithm was run for a maximum of 3000 outer iterations with an upper cap of 50 iterations for every inner loop. We find Algorithm 1 to be more appealing than Algorithm 2 because of its simplicity and its stable behavior for large values of n. However, if one indeed seeks a method with established convergence properties, we recommend the use of Algorithm 2. We consider these two methods to be close cousins of one another.

A Appendix

A.1 Proof of Lemma 3

Proof. The proof follows by observing that $\nabla\tilde{\phi}(\mathbf{x};\tau) = \sum_{i=1}^{m}\hat{w}_i\mathbf{a}_i$, where $\hat{\mathbf{w}}$ is a maximizer of the optimization Problem (26). Now, observe that

$$\|\nabla\tilde{\phi}(\mathbf{x};\tau)\|_2 = \left\|\sum_{i=1}^{m}\hat{w}_i\mathbf{a}_i\right\|_2 \le \sum_{i=1}^{m}\hat{w}_i\|\mathbf{a}_i\|_2 \le \sum_{i=1}^{m}\hat{w}_iL, \tag{44}$$

where, above, we used the triangle inequality and the fact that $\|\mathbf{a}_i\|_2 \le L$ for all i. The latter follows from the simple observation that each $\mathbf{a}_i = \hat{\boldsymbol{\xi}}_i$ and $\|\hat{\boldsymbol{\xi}}_i\|_2 \le L$ (since $\hat{\boldsymbol{\xi}}_i$'s are solutions to Problem (33)). We thus have, from (44), that $\|\nabla\tilde{\phi}(\mathbf{x};\tau)\|_2 \le \sum_i \hat{w}_iL = L$. $\qquad\square$

A.2 Algorithm for Solving SOCP (38)

Problem (38) can be rewritten as the following optimization problem:

$$\underset{\boldsymbol{\xi}_j}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{A}_j\boldsymbol{\xi}_j - \boldsymbol{b}_j\|_2^2 \quad s.t. \quad \|\boldsymbol{\xi}_j\|_2^2 \le L^2, \tag{45}$$

where $\mathbf{A}_j := [\boldsymbol{\Delta}_{1j}; \boldsymbol{\Delta}_{2j}; \ldots; \boldsymbol{\Delta}_{nj}]$ is formed by stacking the vectors $\boldsymbol{\Delta}_{ij}$'s (for fixed j) into a matrix of size $n \times d$; $\mathbf{b}_j$ is obtained by vectorizing $\bar{\eta}_{ij}, i = 1, \ldots, n$. Observe that the optimization Problem (45) can be equivalently written in the Lagrangian version:

$$\underset{\boldsymbol{\xi}_j}{\text{minimize}} \quad \left(\frac{1}{2}\|\boldsymbol{A}_j\boldsymbol{\xi}_j - \boldsymbol{b}_j\|_2^2 + \lambda_j'\|\boldsymbol{\xi}_j\|_2^2\right) \tag{46}$$

with Lagrangian parameter λ_j'. In fact, there is a choice of λ_j' for which Problems (46) and (45) are equivalent. We describe how to find λ_j' from L by using a root-finding

algorithm. Note that the solution to (46) is given by

$$\boldsymbol{\xi}_j^*(\lambda_j) = \left(\boldsymbol{A}_j^\top \boldsymbol{A}_j + \lambda_j' I\right)^{-1} \boldsymbol{A}_j^\top \boldsymbol{b}_j. \tag{47}$$

We will now simplify the above expression further. Consider the singular value decomposition of $\boldsymbol{A}_j = U_j \Gamma_j V_j^\top$ where U_j and V_j are $n \times d$ and $d \times d$ orthogonal matrices respectively and Γ_j is a $d \times d$ diagonal matrix with diagonal entries $\gamma_i, i = 1, ..., d$. Then

$$\left(\boldsymbol{A}_j^\top \boldsymbol{A}_j + \lambda_j' I\right)^{-1} = V_j \Gamma_j' V_j^\top \tag{48}$$

where $\Gamma_j' = \mathrm{diag}((\gamma_1^2 + \lambda_j)^{-1}, \ldots, (\gamma_d^2 + \lambda_j)^{-1})$. We have

$$\boldsymbol{\xi}_j^*(\lambda_j) = (V_j \Gamma_j' V_j^\top) V_j \Gamma_j U_j^\top b_j = V_j \Gamma_j''(\lambda_j) U_j^\top \boldsymbol{b}_j, \tag{49}$$

where $\Gamma_j''(\lambda_j) = \Gamma_j' \Gamma_j = \mathrm{diag}(\gamma_1/(\gamma_1^2 + \lambda_j), \ldots, (\gamma_d/(\gamma_d^2 + \lambda_j)))$, and we used the fact that $V_j^\top V_j = I_d$, the $d \times d$ identity matrix. Note that $\|\boldsymbol{\xi}_j(\lambda_j)\|_2^2$ is given by

$$\|\boldsymbol{\xi}_j^*(\lambda_j)\|_2^2 = \boldsymbol{b}_j^\top U_j \Gamma_j'''(\lambda_j) U_j^\top \boldsymbol{b}_j \tag{50}$$

where, $\Gamma_j'''(\lambda_j) = \mathrm{diag}(\gamma_1^2/(\gamma_1^2 + \lambda_j)^2, \ldots, \gamma_d^2/(\gamma_d^2 + \lambda_j)^2)$. We need to find a value of λ_j for which $\|\boldsymbol{\xi}_j^*(\lambda_j)\|_2 = L$, if such a value exists. Of course, the equality will not hold if $L > \|\boldsymbol{\xi}_j^*(\lambda_j)\|_2$ for all values of λ. Note that the largest value of $\|\boldsymbol{\xi}_j^*(\lambda_j)\|_2$ is when $\lambda_j = 0$, which corresponds to the unconstrained least squares solution $\|\boldsymbol{\xi}_j^*(0)\|_2$. Thus given a value of L we first need to check if L is larger than $\|\boldsymbol{\xi}_j^*(0)\|_2$. If yes, then the solution to Problem (45) will be $\boldsymbol{\xi}_j^*(0)$. Otherwise, there exists a value of $\lambda_j > 0$ for which the equality $\|\boldsymbol{\xi}_j^*(\lambda_j)\|_2 = L$ holds — this corresponds to the case where the Lipschitz regularization is effective. For finding the value of $\lambda_j > 0$ that corresponds to the given L, we employ a standard Newton-Raphson type root-finding method (see e.g., Nesterov (2004)) to search for λ_j that satisfies the equality $\boldsymbol{b}_j^\top U_j \Gamma_j'''(\lambda_j) U_j^\top \boldsymbol{b}_j = L^2$, up to numerical precision. Once the λ_j^* that satisfies $\|\boldsymbol{\xi}_j^*(\lambda_j^*)\|_2 = L$ is obtained, the corresponding $\boldsymbol{\xi}_j^*(\lambda_j^*)$ gives us a solution to Problem (38).

Note that the main computational cost in this problem lies in doing an SVD of $\boldsymbol{A}_j$ with cost $O(nd^2)$ (assuming $n \gg d$). However, this can be pre-computed before running the ADMM algorithm. The $\boldsymbol{b}_j$'s need to be updated after every ADMM iteration, but once the vectors $U_j^\top \boldsymbol{b}_j$'s are computed with cost $O(nd)$, the different Newton-Raphson iterations cost $O(d)$ for computing the function values and gradients. Thus, the overall cost of the algorithm described above is $O(nd)$, in addition to the SVD computations which can be done off-line. Note that there are n different vectors $\boldsymbol{\xi}_j$'s that need to be updated leading to an overall cost of $O(n^2 d)$ for the

update in all the $\boldsymbol{\xi}_j$'s.

A.3 A coordinate descent algorithm for Problem (43)

Problem (43) is a special instance of the following non-negative LS problem:

$$\underset{\mathbf{u} \in \Re^d}{\text{minimize}} \quad \mathbf{u}^\top Q \mathbf{u} + \langle \mathbf{a}, \mathbf{u} \rangle \quad s.t. \quad \mathbf{u} \geq 0, \tag{51}$$

which involves the minimization of a convex quadratic $Q \succeq \mathbf{0}$ function with separable constraints. Due to the smoothness of the convex objective function and separability of the constraints, one-at-a-time coordinate descent can be employed for the problem; see e.g., Luo and Tseng (1992), Friedman et al. (2007).

Algorithm 2 Coordinate descent for Problem (43)

---

1. Start with $\mathbf{u}^1 \geq 0$. For $m \geq 1$ do the following until $\|\mathbf{u}^{m+1} - \mathbf{u}^m\|_2 \leq \text{TOL} \|\mathbf{u}^m\|_2$ for some predefined tolerance "TOL".

2. Assign $\mathbf{u} \leftarrow \mathbf{u}^m$ and for $k \in \{1, \ldots, d\}$ do the following

   (a) Fix u_ℓ for $\ell \neq k$ and update

   $$u_k = \underset{u \geq 0}{\arg\min} \ \left(q_{kk} u^2 + \tilde{a}_k u \right) \ = \max \left\{ -\frac{\tilde{a}_k}{2 q_{kk}}, 0 \right\},$$

   where $\tilde{a}_k = 2 \sum_{\ell \neq k} q_{k\ell} u_\ell + a_k$ and $Q = ((q_{ij}))$.

3. Assign $\mathbf{u}^{m+1} \leftarrow \mathbf{u}$ and go to Step 2.

Note that the vector $\tilde{\mathbf{a}} = (\tilde{a}_1, \ldots, \tilde{a}_d)$ is defined as $\tilde{\mathbf{a}} := Q\mathbf{u} + \mathbf{a}$ and once the k'th coordinate u_k is updated by δ_k (say), then $\tilde{\mathbf{a}} \leftarrow \tilde{\mathbf{a}} + Q[, k] \delta_k$ where, $Q[, k]$ denotes the k'th column of the matrix Q and the whole update in $\tilde{\mathbf{a}}$ can be performed in $O(d)$ operations. Thus for one full cycle over the d coordinates the total cost is $O(d^2)$. Note that in the process of updating the coordinates u_k many of the coordinates that were at zero stay at zero, thus no updating is required for that coordinate — this leads to sparse updating rules in u_k and often leads to significantly improved computational performance. More precisely, if out of the d coordinates a few of the coordinates need to be updated the total cost of performing Step 2 reduces to $O(d)$. The interested reader can also see Friedman et al. (2007) for related computational tricks employed in ℓ_1-regularized problems.

Note that Step 1 of the algorithm requires a starting vector $\mathbf{u}^1$. Since the above algorithm is used as a part of the ADMM algorithm, the current solution of Problem (43) can be used as a warm-start for the above algorithm. This often leads to performing fewer cycles across the d coordinates.

A.4 Proof of Lemma 4

Proof. The proof follows by observing that $\nabla\tilde{\phi}(\mathbf{x};\tau) = \sum_{i=1}^{m}\hat{w}_i\mathbf{a}_i$, where $\hat{\mathbf{w}}$ is a maximizer of the optimization Problem (26). Since $\hat{\mathbf{w}} \in \Delta_m$, every coordinate of $\nabla\tilde{\phi}(\mathbf{x};\tau)$ is a convex combination of the coordinates of $\mathbf{a}_1,\ldots,\mathbf{a}_m$. Note that by (24), for every $i = 1,\ldots,m$, we have $\mathbf{a}_i = \hat{\boldsymbol{\xi}}_i \geq 0$ (since it is a solution to Problem (40)). Thus, for every coordinate k, we have $\nabla_k\tilde{\phi}(\mathbf{x};\tau) \geq 0$. This completes the proof of the lemma. $\qquad\square$

A.5 Proof of Theorem 2

Proof. The theorem follows from known metric entropy results on the class of uniformly bounded convex functions that are uniformly Lipschitz in conjunction with known results on the rates of convergence of LSEs; see e.g., Van de Geer (2000, Theorem 9.1). We give the details below.

The notion of covering numbers will be useful. For $\epsilon > 0$ and a subset S of functions, the ϵ-covering number of S under the metric ℓ, denoted by $N(S,\epsilon;\ell)$, is defined as the smallest number of closed balls of radius ϵ (under the metric ℓ) whose union contains S.

Fix any $B > 0$ and $L > L_0$. We define the class of uniformly bounded convex functions that are uniformly Lipschitz as

$$\mathcal{C}_{L,B} := \{\psi \in \mathcal{C}_L : \|\psi\|_{\mathfrak{x}} \leq B\}, \tag{52}$$

where $\|\psi\|_{\mathfrak{x}} := \sup_{x\in\mathfrak{x}}|\psi(x)|$. Using Theorem 3.2 of Guntuboyina and Sen (2013) (also see Bronshtein (1976)) we know that

$$\log N\left(\mathcal{C}_{L,B},\epsilon;\ell_\infty\right) \leq c\left(\frac{B+dL}{\epsilon}\right)^{d/2}, \tag{53}$$

for all $0 < \epsilon \leq \epsilon_0(B+dL)$, where $\epsilon_0 > 0$ is a fixed constant and ℓ_∞ is the uniform metric. Let $\hat{\phi}_{n,L,B}$ denote a LSE of ϕ in the class $\mathcal{C}_{L,B}$. Routine calculations and

Theorem 9.1 of Van de Geer (2000) now yields

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{\phi}_{n,L,B}(\boldsymbol{X}_i) - \phi(\boldsymbol{X}_i))^2 = O_{\mathbb{P}}(r_n), \tag{54}$$

where r_n is defined as in (35).

Define the event $A_n := \{\max_{i=1,\ldots,n} |\hat{\phi}_{n,L}(\boldsymbol{X}_i)| \le B_0\}$. Next we show that there exists $B_0 > 0$ such that

$$\mathbb{P}(A_n) \to 1, \qquad \text{as } n \to \infty. \tag{55}$$

From the characterization of the projection on the closed convex set $\mathcal{C}_L$, we know that

$$\sum_{i=1}^{n}(Y_i - \hat{\phi}_{n,L}(\boldsymbol{X}_i))(\gamma(\boldsymbol{X}_i) - \hat{\phi}_{n,L}(\boldsymbol{X}_i)) \le 0,$$

for all $\boldsymbol{\gamma} \in \mathcal{C}_L$. Letting $\mathbf{e} \equiv 1$ denote the constant 1 convex function, note that for any $c \in \Re$, $c\mathbf{e} \in \mathcal{C}_L$. Hence simple algebra yields $\sum_{i=1}^{n}(Y_i - \hat{\phi}_{n,L}(\boldsymbol{X}_i))\mathbf{e}(\boldsymbol{X}_i) = 0$, i.e., $\sum_{i=1}^{n} Y_i = \sum_{i=1}^{n} \hat{\phi}_{n,L}(\boldsymbol{X}_i)$. Now, letting $\bar{Y} = \sum_{i=1}^{n} Y_i/n$, for any $\mathbf{x} \in \mathfrak{X}$,

$$
\begin{aligned}
|\hat{\phi}_{n,L}(\mathbf{x})| &\le |\hat{\phi}_{n,L}(\mathbf{x}) - \bar{Y}| + |\bar{Y}| = \left|\hat{\phi}_{n,L}(\mathbf{x}) - \frac{1}{n}\sum_{j=1}^{n}\hat{\phi}_{n,L}(\mathbf{X}_j)\right| + |\bar{Y}| \\
&\le \frac{1}{n}\sum_{j=1}^{n}\left|\hat{\phi}_{n,L}(\mathbf{x}) - \hat{\phi}_{n,L}(\mathbf{X}_j)\right| + |\bar{Y}| \\
&\le \frac{L}{n}\sum_{j=1}^{n}\|\mathbf{x} - \boldsymbol{X}_j\|_2 + \|\phi\|_{\mathfrak{X}} + |\bar{\epsilon}| \\
&\le \sqrt{d}L + \kappa + 1 =: B_0,
\end{aligned}
$$

a.s. for large enough n, where we have used the fact that $\|\mathbf{x} - \mathbf{X}_j\|_2 \le \sqrt{d}$, $\|\phi\|_{\mathfrak{X}} < \kappa$ for some $\kappa > 0$, and that $\bar{\epsilon} := \sum_{i=1}^{n} \epsilon_i/n \to 0$ a.s. As $\mathcal{C}_{L,B_0} \subset \mathcal{C}_L$, we trivially have

$$\sum_{i=1}^{n}(\hat{\phi}_{n,L}(\mathbf{X}_i) - Y_i)^2 \le \sum_{i=1}^{n}(\hat{\phi}_{n,L,B_0}(\boldsymbol{X}_i) - Y_i)^2.$$

If A_n happens, $\hat{\phi}_{n,L} \in \mathcal{C}_{L,B_0}$, and thus,

$$\sum_{i=1}^{n}(\hat{\phi}_{n,L}(\mathbf{X}_i) - Y_i)^2 \ge \sum_{i=1}^{n}(\hat{\phi}_{n,L,B_0}(\mathbf{X}_i) - Y_i)^2.$$

From the last two inequalities and the uniqueness of the projections it follows that if

A_n occurs, then $\hat{\phi}_{n,L} = \hat{\phi}_{n,L,B_0}$ at the data points. Now using (55), (35) immediately follows from (54). $\qquad\square$
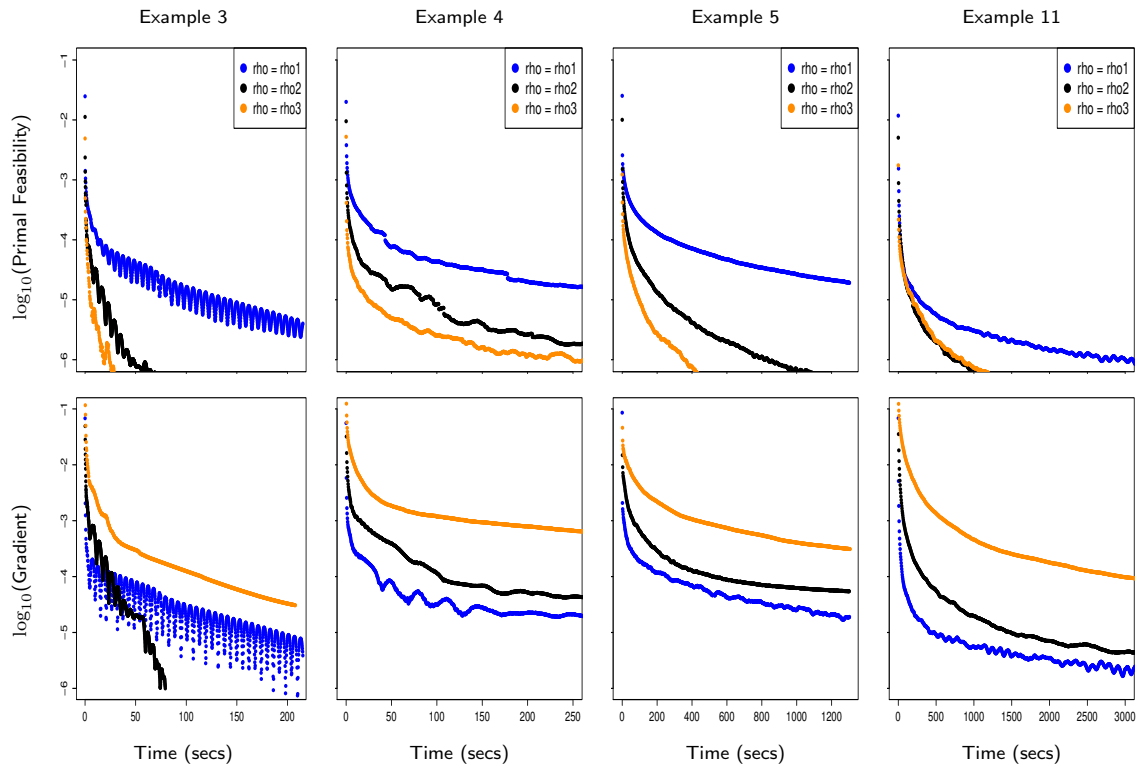
B   Additional Computational Results

Figure 4: Figure showing the convergence characteristics of Algorithm 1 for different examples, as described in the text. The legends have the same meanings as in Figure 3.

References

Aguilera, N., L. Forzani, and P. Morin (2011). On uniform consistent estimators for convex regression. *J. Nonparametr. Stat. 23*(4), 897–908.

Allon, G., M. Beenstock, S. Hackman, U. Passy, and A. Shapiro (2007). Nonparametric estimation of concave production technologies by entropic methods. *J. Appl. Econometrics 22*(4), 795–816.

Aybat, N. S. and G. Iyengar (2012). A first-order augmented lagrangian method for compressed sensing. *SIAM Journal on Optimization 22*(2), 429–459.

Aybat, N. S. and Z. Wang (2014). A parallel method for large scale convex regression problems. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pp. 5710–5717. IEEE.

Ayer, M., H. D. Brunk, G. M. Ewing, W. T. Reid, and E. Silverman (1955). An empirical distribution function for sampling with incomplete information. *Ann. Math. Statist. 26*, 641–647.

Balabdaoui, F. (2007). Consistent estimation of a convex density at the origin. *Mathematical Methods of Statistics 16*(2), 77–95.

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

Bertsekas, D. P. (1999). *Nonlinear Programming* (2nd ed.). Athena Scientific, Belmont, Massachusetts.

Birke, M. and H. Dette (2007). Estimating a convex function in nonparametric regression. *Scand. J. Statist. 34*(2), 384–404.

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

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

Bronshtein, E. M. (1976). ϵ-entropy of convex sets and functions. *Siberian Mathematical Journal 17*, 393–398.

Brunk, H. D. (1955). Maximum likelihood estimates of monotone parameters. *Ann. Math. Statist. 26*, 607–616.

Chen, C., B. He, Y. Ye, and X. Yuan (2014). The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, 1–23.

Cule, M., R. Samworth, and M. Stewart (2010). Maximum likelihood estimation of a multi-dimensional log-concave density. *J. R. Stat. Soc. Ser. B Stat. Methodol. 72*(5), 545–607.

Du, P., C. F. Parmeter, and J. S. Racine (2013). Nonparametric kernel regression with multiple predictors and multiple shape constraints. *Statist. Sinica 23*(3), 1347–1371.

Friedman, J., T. Hastie, H. Hoefling, and R. Tibshirani (2007). Pathwise coordinate optimization. *Annals of Applied Statistics 2*(1), 302–332.

Guntuboyina, A. and B. Sen (2013). Covering numbers for convex functions. *IEEE Trans. Inf. Th. 59*(4), 1957–1965.

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

Hastie, T., R. Tibshirani, and J. Friedman (2009). *The elements of statistical learning* (Second ed.). Springer Series in Statistics. Springer, New York. Data mining, inference, and prediction.

Hong, M. and Z.-Q. Luo (2012). On the linear convergence of the alternating direction method of multipliers. *arXiv preprint arXiv:1208.3922*.

Kuosmanen, T. (2008). Representation theorem for convex nonparametric least

squares. *The Econometrics Journal 11*(2), 308–325.

Lim, E. (2014). On convergence rates of convex regression in multiple dimensions. *INFORMS Journal on Computing 26*(3), 616–628.

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

Luo, Z.-Q. and P. Tseng (1992). On the convergence of coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications 72(1)*, 7–35.

Mammen, E. (1991). Estimating a smooth monotone regression function. *Ann. Statist. 19*(2), 724–740.

Matzkin, R. L. (1994). Restrictions of economic theory in nonparametric methods. *Handbook of econometrics 4*, 2523–2558.

Mekaroonreung, M. and A. L. Johnson (2012). Estimating the shadow prices of so 2 and no x for us coal power plants: a convex nonparametric least squares approach. *Energy Economics 34*(3), 723–732.

Meyer, R. F. and J. W. Pratt (1968). The consistent assessment and fairing of preference functions. *Systems Science and Cybernetics, IEEE Transactions on 4*(3), 270–278.

Michelot, C. (1986). A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbf{R}^n$. *J. Optim. Theory Appl. 50*(1), 195–200.

Mukerjee, H. (1988). Monotone nonparameteric regression. *Ann. Statist. 16*(2), 741–750.

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

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

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

Seregin, A. and J. A. Wellner (2010). Nonparametric estimation of multivariate convex-transformed densities. *Ann. Statist. 38*(6), 3751–3781. With supplementary material available online.

Shapiro, A., D. Dentcheva, and A. Ruszczyński (2009). *Lectures on stochastic programming*, Volume 9 of *MPS/SIAM Series on Optimization*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA; Mathematical Programming Society (MPS), Philadelphia, PA. Modeling and theory.

Van de Geer, S. (2000). *Applications of Empirical Process Theory.* Cambridge University Press.

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

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

Verbeek, M. (2008). *A guide to modern econometrics.* John Wiley & Sons.

Wang, Y. and S. Wang (2013). Estimating α-frontier technical efficiency with shape-restricted kernel quantile regression. *Neurocomputing 101*, 243–251.

Yatchew, A. (1998). Nonparametric regression techniques in economics. *Journal of Economic Literature*, 669–721.