15.095: Machine Learning under a Modern Optimization Lens

Lecture 6: Beyond Linear Regression

Overview

Going beyond usual linear regression:

$$\text{Linear model:} \quad \mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

$$\text{Least squares:} \quad \min_{\boldsymbol{\beta}} \sum_i \left(y_i - \mathbf{x}_i' \boldsymbol{\beta}\right)^2$$

Today:

- Nonlinear models: $y_i = f(\mathbf{x}_i) + \epsilon_i$
- Different loss functions

Motivation

Convex regression aims to find the "best" convex function that fits the given data $(\mathbf{x}_i, y_i), i = 1, \ldots, n$.

Applications in various fields such as

- Econometrics—concave demand, production, and utility functions
- Reinforcement learning
- Target reconstruction
- Resource allocation
- Queueing network performance analysis
- Geometric programming

Main problem

Solve

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

where $\mathcal{C}$ is the space of convex functions on $\mathbb{R}^p$.

How to reformulate? For now, assume f is differentiable.

- f is convex iff $f(\mathbf{x}) \geq f(\bar{\mathbf{x}}) + \langle \nabla f(\bar{\mathbf{x}}), \mathbf{x} - \bar{\mathbf{x}} \rangle \quad \forall\, \mathbf{x}, \bar{\mathbf{x}}$
- To solve estimation problem, only care about $f(\mathbf{x}_i)$ and $\nabla f(\mathbf{x}_i)$.
- Decision variables: θ_i $(= f(\mathbf{x}_i))$ and $\boldsymbol{\sigma}_i$ $(= \nabla f(\mathbf{x}_i))$
- Constraints: $\theta_j \geq \theta_i + \langle \boldsymbol{\sigma}_i, \mathbf{x}_j - \mathbf{x}_i \rangle \quad \forall\, i, j$

In general, f does not need to be differentiable, and so $\boldsymbol{\sigma}_i$ can be any subgradient of f at $\mathbf{x}_i$.

Reformulation

Based on these observations, we can reformulate the problem exactly as

$$\begin{aligned} \min_{\boldsymbol{\theta}, \{\boldsymbol{\sigma}_i\}_{i=1}^n} \quad & \sum_{i=1}^n (y_i - \theta_i)^2 \\ \text{subject to} \quad & \theta_i + \langle \boldsymbol{\sigma}_i, \mathbf{x}_j - \mathbf{x}_i \rangle \le \theta_j \quad \forall\, i, j, \\ & \boldsymbol{\theta} \in \mathbb{R}^n, \\ & \boldsymbol{\sigma}_i \in \mathbb{R}^p \quad \forall\, i. \end{aligned}$$

How to get an f defined on all of $\mathbb{R}^p$?

$$f(\mathbf{x}) := \max_i \left\{ \widehat{\theta}_i + \langle \widehat{\boldsymbol{\sigma}}_i, \mathbf{x} - \mathbf{x}_i \rangle \right\}$$

(Another) Cutting plane algorithm

Problem has $O(n^2)$ constraints $\rightsquigarrow$ doesn't scale well for $n \geq 300$.

Instead, we use a delayed constraint generation approach—adding constraints as you go, rather than all at once.

We treat the constraints as n blocks, with the ith block as:

$$\theta_i + \langle \boldsymbol{\sigma}_i, \mathbf{x}_j - \mathbf{x}_i \rangle \leq \theta_j \quad \forall\, 1 \leq j \leq n.$$

Initially, we add one constraint from each block.

Cutting plane algorithm

- After solving this, find the $j(i)$ for each block i:

$$j(i) = \arg\max_{1\le k\le n} \left\{\hat{\theta}_i - \hat{\theta}_k + \langle \hat{\boldsymbol{\sigma}}_i, \mathbf{x}_k - \mathbf{x}_i\rangle\right\},$$

  and check if the maximum value is more than `Tol`.

- Add these (at most) n constraints given by:

$$\theta_i + \langle \boldsymbol{\sigma}_i, \mathbf{x}_{j(i)} - \mathbf{x}_i\rangle \le \theta_{j(i)}.$$

- Re-solve the problem with these extra constraints.
- Iterate until there are no more violations for each block.

Computational Results - Data

- $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
- $\Phi(\mathbf{x}) = \|\mathbf{x}\|_2^2$, and $\mu_i = \Phi(\mathbf{x}_i)$.
- We set σ so that the Signal to Noise ratio (SNR) is 3, i.e., $\frac{\mathrm{Var}(\mu)}{\mathrm{Var}(\epsilon)} = 3$.
- $\epsilon_i \sim \mathcal{N}(0, \sigma^2)\ \forall i$.
- $\mathbf{y} = \boldsymbol{\mu} + \boldsymbol{\epsilon}$
- `Tol` is the numerical tolerance for each of the $n(n-1)$ constraints.

Scalability

$$\text{Infeasibility} = \frac{1}{n}\left(\sum_{ij} V_{ij}^2\right)^{1/2}$$

where $V_{ij} = (\hat{\theta}_i + \langle \hat{\boldsymbol{\sigma}}_i, \mathbf{x}_j - \mathbf{x}_i\rangle - \hat{\theta}_j)_+$.

n	p	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)
10^5	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: Run times for `Tol` $= 0.1$ and ℓ_2 convex regression.

Scalability for lower tolerance

n	p	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)

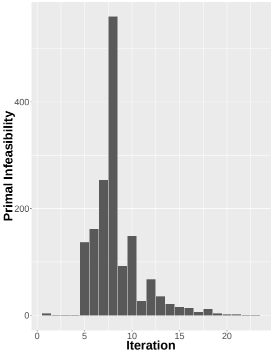
Table: Run times for $\texttt{Tol} = 0.05$ and ℓ_2 convex regression.

Discussion: Analyzing the run times

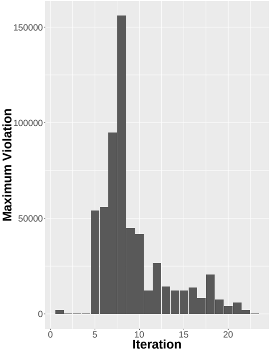
- Regressed run times T versus n, p, and `Tol` using the data.
- Almost linear relationship between T and $(n^{1.5}, p^{1.75}, \texttt{Tol})$ $(R^2 = 0.9681)$

Infeasibility as a function of iterations

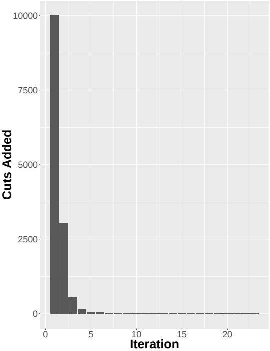
$$\text{Maximum violation} = \max_{i,j}\{\hat{\theta}_i - \hat{\theta}_j + \langle \hat{\boldsymbol{\sigma}}_i, \mathbf{x}_j - \mathbf{x}_i \rangle\}$$



(a) ("Global") primal infeasibility used previously



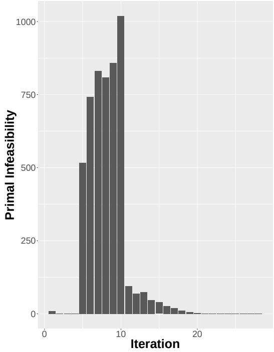
(b) Maximum violation above



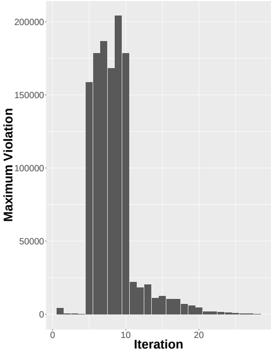
(c) Number of constraints added at each iteration

$(n, p, \texttt{Tol}) = (10^4, 10, 0.1)$

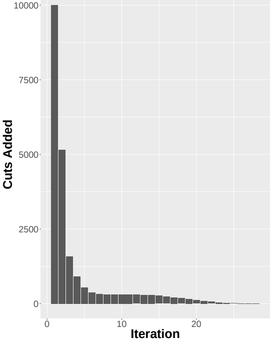
Infeasibility as a function of iterations



(a) (“Global”) primal infeasibility



(b) Maximum violation



(c) Number of constraints added at each iteration

$$(n, p, \texttt{Tol}) = (10^4, 10, 0.05)$$

Alternate approaches

- (Mazumder et al 2016) propose an ADMM approach.
  - Exploit the block structure of the problem.
  - Decompose into subproblems exclusively involving $\boldsymbol{\theta}, \boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_n$.
- (Balázs et al 2015) propose an aggregated cutting plane approach.
  - Along with usual cuts, they add (aggregate) certain convexity constraints and add (or delete) them to the QP iteratively.
  - Aggregated constraints intuitively motivated by the convex hull of points $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

Summary

Solving regression problems beyond the usual linear model can be accomplished using the same optimization techniques that we have encountered already.

Background

The traditional Least Squares (LS) estimator given by

$$\widehat{\boldsymbol{\beta}}^{(\mathrm{LS})} \in \operatorname*{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{n} r_i^2,$$

where $r_i := y_i - \mathbf{x}_i' \boldsymbol{\beta}$.

$\widehat{\boldsymbol{\beta}}^{(\mathrm{LS})}$ is not "robust": a single outlier can have an arbitrarily large effect on the estimate! ("zero breakdown point")

Background

The Least Absolute Deviation (LAD) estimator:

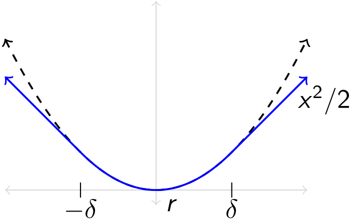
$$\widehat{\boldsymbol{\beta}}^{(\mathrm{LAD})} \in \operatorname*{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^{n} |r_i|$$

. . . but not resistant to large deviations in the covariates. Breakdown point is zero.

M-estimators: minimize a loss function $\sum_{i=1}^{n} \rho(r_i)$, where $\rho(r)$ is a symmetric function with a unique minimum at zero.

Example: Huber function:

$$\rho_\delta(r) = \begin{cases} \frac{1}{2} r^2 & |r| \le \delta \\ \delta(|r| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$$



Still affected by outliers in the covariates, but marginally better breakdown point.

Background

Rousseeuw (1984) introduced the Least Median of Squares (LMS) estimator:

$$\widehat{\boldsymbol{\beta}}^{(\mathrm{LMS})} \in \operatorname*{argmin}_{\boldsymbol{\beta}} \left(\operatorname*{median}_{i=1,\dots,n} |r_i| \right).$$

Has a limiting breakdown point of 50% (first *equivariant* estimator to achieve maximal possible breakdown point in the limit $n \to \infty$ with p fixed).

More generally: Least Quantile of Squares (LQS) estimator:

$$\widehat{\boldsymbol{\beta}}^{(\mathrm{LQS})} \in \operatorname*{argmin}_{\boldsymbol{\beta}} |r_{(q)}|,$$

where $r_{(q)}$ denotes the residual, corresponding to the qth ordered absolute residual: $|r_{(1)}| \leq |r_{(2)}| \leq \ldots \leq |r_{(n)}|$

Some properties

Theorem

The LQS problem is equivalent to

$$\min_{\boldsymbol{\beta}} \ |r_{(q)}| = \min_{\mathcal{I} \in \Omega_q} \left(\min_{\boldsymbol{\beta}} \ \ \|\mathbf{y}_{\mathcal{I}} - \mathbf{X}_{\mathcal{I}} \boldsymbol{\beta}\|_{\infty} \right),$$

where $\Omega_q := \{\mathcal{I} : \ \mathcal{I} \subseteq \{1, \ldots, n\}, |\mathcal{I}| = q\}$ *and* $(\mathbf{y}_{\mathcal{I}}, \mathbf{X}_{\mathcal{I}})$ *denotes the subsample* $(y_i, \mathbf{x}_i), i \in \mathcal{I}$.

Thus LQS is also doing subset selection, but *in the samples!*

Breakdown point of an estimator

Let $\Theta(\mathbf{y}, \mathbf{X})$ denote an estimator based on a sample $(\mathbf{y}, \mathbf{X})$.

Original sample is $(\mathbf{y}, \mathbf{X})$ and m of the sample points have been replaced arbitrarily, and let $(\mathbf{y} + \Delta_{\mathbf{y}}, \mathbf{X} + \Delta_{\mathbf{X}})$ be the perturbed sample.

Let

$$\alpha(m; \Theta; (\mathbf{y}, \mathbf{X})) = \sup_{(\Delta_{\mathbf{y}}, \Delta_{\mathbf{X}})} \|\Theta(\mathbf{y}, \mathbf{X}) - \Theta(\mathbf{y} + \Delta_{\mathbf{y}}, \mathbf{X} + \Delta_{\mathbf{X}})\|_2$$

denote the maximal change in the estimator under this perturbation.

The finite sample breakdown point of the estimator Θ is defined as follows:

$$\eta(\Theta; (\mathbf{y}, \mathbf{X})) := \min_{m} \left\{ \frac{m}{n} \,\middle|\, \alpha(m; \Theta; (\mathbf{y}, \mathbf{X})) = \infty \right\}.$$

Breakdown point of an estimator

Theorem

If $\widehat{\beta}^{(LQS)}$ denotes an optimal solution and $\Theta := \Theta(\mathbf{y}, \mathbf{X})$ denotes the optimum objective value to the LQS problem for a given dataset $(\mathbf{y}, \mathbf{X})$, then the finite sample breakdown point of Θ is $(n - q + 1)/n$.

- For the LMS problem, we have $q = n - \lfloor n/2 \rfloor$, which leads to the sample breakdown point of objective value $(\lfloor n/2 \rfloor + 1)/n$
(no dependence on number of covariates)
- It was known that LMS *solutions* have a sample breakdown point of $(\lfloor n/2 \rfloor - p + 2)/n$ (when the data is in general position)

Computation

- Bernholt (2005) showed that LMS is NP-hard

- State of the art:
  - Exact approaches based on complete enumeration: $O(n^p)$. Typically do not scale to more than $n = 50$ and $p = 5$.
  - Heuristic approaches (scale to larger sizes), but are very ad hoc.

MIO formulation

Objective: $$\min_{\boldsymbol{\beta}} \; |r_{(q)}|$$

- Introduce binary variables:

$$z_i = \begin{cases} 1, & \text{if } |r_i| \leq |r_{(q)}|, \\ 0, & \text{otherwise.} \end{cases}$$

- Auxiliary continuous variables $\mu_i \geq 0$ such that:

$$|r_i| - \mu_i \leq |r_{(q)}|,$$

with the condition

$$\text{if } |r_i| \leq |r_{(q)}|, \text{ then } \mu_i = 0.$$

(Why? Think about minimizing $\max_i(|r_i| - \mu_i)$.)

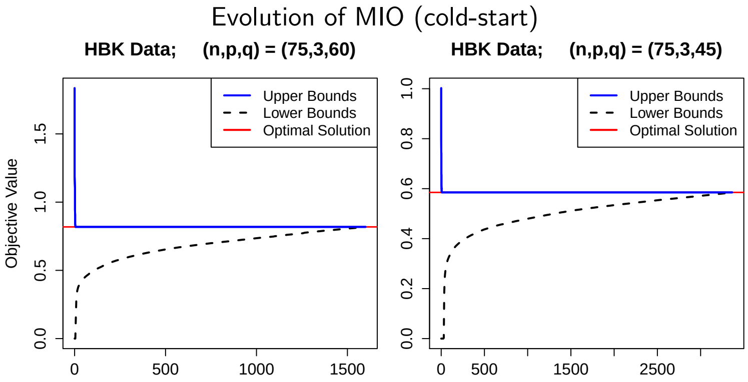
MIO formulation

This leads to the following MIO formulation for LQS:

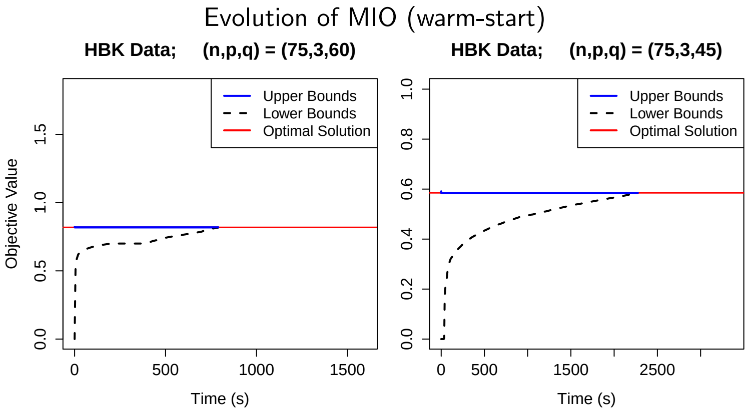
$$
\begin{aligned}
\min_{\gamma,\mathbf{z},\boldsymbol{\mu}} \quad & \gamma \\
\text{subject to} \quad & \mathbf{r} = \mathbf{y} - \mathbf{X}\beta \\
& \gamma \geq |r_i| - \mu_i, \quad i = 1\ldots, n \\
& M(1 - z_i) \geq \mu_i, \quad i = 1, \ldots, n \\
& \sum_{i=1}^{n} z_i = q \\
& \mu_i \geq 0, \quad i = 1, \ldots, n \\
& z_i \in \{0, 1\}, \quad i = 1, \ldots, n,
\end{aligned}
\tag{1}
$$

where M is a big-M constant.

Algorithm in action



Algorithm in action



Finding fast upper bounds

- Algorithm 1: Upper bounds via Sequential Linear Optimization (difference of convex programs)
- Algorithm 2: Subdifferential based algorithm
- Algorithm 3: Algorithm 2 followed by Algorithm 1.

Algorithm 1: Sequential Linear Optimization

- Decompose the qth ordered residual as:

$$|r_{(q)}| = |y_{(q)} - \mathbf{x}'_{(q)}\boldsymbol{\beta}| = \underbrace{\sum_{i=q}^{n} |y_{(i)} - \mathbf{x}'_{(i)}\boldsymbol{\beta}|}_{H_q(\boldsymbol{\beta})} - \underbrace{\sum_{i=q+1}^{n} |y_{(i)} - \mathbf{x}'_{(i)}\boldsymbol{\beta}|}_{H_{q+1}(\boldsymbol{\beta})},$$

- $H_m(\boldsymbol{\beta})$ is convex in $\boldsymbol{\beta}$.
- $|y_{(q)} - \mathbf{x}'_{(q)}\boldsymbol{\beta}|$ as a difference of convex functions
- $|y_{(q)} - \mathbf{x}'_{(q)}\boldsymbol{\beta}| = H_q(\boldsymbol{\beta}) - \underbrace{H_{q+1}(\boldsymbol{\beta})}_{\text{Linearize}}$

$$H_{q+1}(\boldsymbol{\beta}) \approx H_{q+1}(\boldsymbol{\beta}_k) + \langle \partial H_{q+1}(\boldsymbol{\beta}_k), \boldsymbol{\beta} - \boldsymbol{\beta}_k \rangle,$$

where $\partial H_{q+1}(\boldsymbol{\beta}_k)$ is a sub-gradient of $H_{q+1}(\boldsymbol{\beta}_k)$.

Algorithm 1: Sequential Linear Optimization

- The function $H_m(\boldsymbol{\beta})$ can be written as

$$
\begin{aligned}
H_m(\boldsymbol{\beta}) := \quad & \max_{\mathbf{w}} \quad \sum_{i=1}^{n} w_i |y_i - \mathbf{x}_i' \boldsymbol{\beta}| \\
& \text{subject to} \quad \sum_{i=1}^{n} w_i = n - m + 1 \\
& \qquad\qquad\quad 0 \le w_i \le 1, \quad i = 1, \ldots, n.
\end{aligned}
$$

- Dual representation of $H_m(\boldsymbol{\beta})$:

$$
\begin{aligned}
H_m(\boldsymbol{\beta}) = \quad & \min_{\theta, \boldsymbol{\nu}} \quad \theta\,(n - m + 1) + \sum_{i=1}^{n} \nu_i \\
& \text{subject to} \quad \theta + \nu_i \ge |y_i - \mathbf{x}_i' \boldsymbol{\beta}|, \qquad i = 1, \ldots, n \\
& \qquad\qquad\quad \nu_i \ge 0, \qquad i = 1, \ldots, n.
\end{aligned}
$$

Algorithm 1: Sequential Linear Optimization

- $|y_{(q)} - \mathbf{x}'_{(q)}\beta| = \underbrace{H_q(\boldsymbol{\beta})}_{\text{Dualize}} - \underbrace{H_{q+1}(\boldsymbol{\beta})}_{\text{Linearize}}$
- To get upper bounds, minimize:

$$\begin{aligned} \min_{\boldsymbol{\nu},\theta,\boldsymbol{\beta}} \quad & \theta(n-q+1) + \sum_{i=1}^{n} \nu_i - \langle \partial H_{q+1}(\boldsymbol{\beta}_k), \boldsymbol{\beta} \rangle \\ \text{subject to} \quad & \theta + \nu_i \geq |y_i - \mathbf{x}_i'\boldsymbol{\beta}|, \qquad i = 1, \ldots, n \\ & \nu_i \geq 0, \qquad i = 1, \ldots, n. \end{aligned}$$

- Get $\boldsymbol{\beta}_{k+1}$ and repeat until convergence.
- Decreasing sequence of objective values — converges to stationary point at a rate $O(1/K)$.

Algorithm 2: Subdifferential Optimization

$$\min_{\boldsymbol{\beta}} \quad f_q(\boldsymbol{\beta}) := |y_{(q)} - \mathbf{x}'_{(q)}\boldsymbol{\beta}|$$

Initialize $\boldsymbol{\beta}_1$, for $k \leq \texttt{MaxIter}$ do the following:

- $\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k - \alpha_k \partial f_q(\boldsymbol{\beta}_k)$ where α_k is a step-size.
  Subdifferential is
  $$\partial f_q(\boldsymbol{\beta}) = -\operatorname{sgn}(y_{(q)} - \mathbf{x}'_{(q)}\boldsymbol{\beta})\mathbf{x}_{(q)}.$$
- Return $\min\limits_{1 \leq k \leq \text{MaxIter}} f_q(\boldsymbol{\beta}_k)$ and $\boldsymbol{\beta}_{k^*}$ at which the minimum is attained, where
  $$k^* = \operatorname*{argmin}_{1 \leq k \leq \text{MaxIter}} f_q(\boldsymbol{\beta}_k).$$

Algorithms 1, 2, and 3

Example (n, p, π) q		**Algorithm Used** #1 (SLO)	#2 (GD)	#3 Hybrid
Ex. 1 (201,5, 0.4) $q = 121$	Error Time (s)	49.399 (2.43) 24.05	0.233 (0.03) 3.29	0.0 (0.0) 36.13
Ex. 2 (201,10, 0.5) $q = 101$	Error Time (s)	43.705 (2.39) 54.39	1.438 (0.07) 3.22	0.0 (0.0) 51.89
Ex. 3 (501,5,0.4) $q = 301$	Error Time (s)	2.897 (0.77) 83.01	0.249 (0.05) 3.75	0.0 (0.0) 120.90
Ex. 4 (501,10, 0.4) $q = 301$	Error Time (s)	8.353 (2.22) 192.02	1.158 (0.06) 3.76	0.0 155.36

Relative Accuracy $= (f_{\text{alg}} - f_*)/f_* \times 100$

Comparison with state-of-the-art

Example (n, p, π) q		Algorithm Used LQS (MASS)	#3	MIO (cold-start)	MIO (warm-start)
Ex-1 (201,5, 0.4)	Accuracy	24.163 (1.31)	0.0 (0.0)	60.880 (5.60)	0.0 (0.0)
$q = 121$	Time (s)	0.02	36.13	71.46	35.32
Ex-2 (201,10, 0.5)	Accuracy	105.387 (5.26)	0.263 (0.26)	56.0141 (3.99)	0.0 (0.0)
$q = 101$	Time (s)	0.05	51.89	193.00	141.10
Ex. 3 (501,5,0.4)	Accuracy	9.677 (0.99)	0.618 (0.27)	11.325 (1.97)	0.127 (0.11)
$q = 301$	Time (s)	0.05	120.90	280.66	159.76
Ex. 4 (501,5,0.4)	Accuracy	29.756 (1.99)	0.341 (0.33)	27.239 (2.66)	0.0 (0.0)
$q = 301$	Time (s)	0.08	155.36	330.88	175.52

LQS: from `R` package `MASS`.

Takeaway messages

- LQS is a classical, useful, and highly robust modeling tool for linear regression with potentially large outliers.
- LQS admits a tractable optimization formulation via MIO.
- Nonlinear Optimization methods for fast/high quality upper bounds. Certify optimality via MIO.
- Scalable for problems up to $n = 10k$ or even more.