Homotopic methods can significantly speed up the warm-up stage of the computation of the Lasso-type of estimators

Yujie Zhao, Xiaoming Huo

Georgia Tech

November 7, 2019

Georgia Tech

Agenda

Georgia Tech

Introduction

Background

Lasso (Tibshirani, 1996) has demonstrated to be an effective methods in model estimation and selection in the past decades. Lasso aims to enable sparsity during the estimation process when the dimension becomes increasingly large.
Let $y \in \mathbb{R}^n$ denote a response vector and $X \in \mathbb{R}^{n \times p}$ be a matrix of predictors, and vector β^* is the true regression coefficients that we want to estimate, and vector w contains white-noise entries, which are independently and identically distributed following the Normal distribution $N(0, \sigma^2)$. Accordingly, the linear regression model can be written as

$$y = X\beta^* + w.$$

Under the model above, the regular Lasso estimator $\widehat{\beta}$ is commonly written as

$$\widehat{\beta} = \arg\min_{\beta} \left\{ \frac{1}{2n} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}, \tag{1}$$

where parameter λ controls the trade-off between the sparsity and model's goodness of fit.

Essential of Lasso

- Essentially, solving equation (1) is an optimization problem, where many research in operations research devotes to this problem, aiming is to minimize the objective function

$$F(\beta) = \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1 \tag{2}$$

  with respect to β.
- Because there is no close form of $\widehat{\beta}$, most Lasso-algorithms are recursive with an iteration index noted as k. Suppose $\beta^{(k)}$ is the estimator after k iterations by a certain Lasso-algorithm.
- Many research focuses on the convergence rate, i.e., how many iterations are needed to satisfy the precision requirement in the following equation.

$$F(\beta^{(k)}) - F(\widehat{\beta}) \leq \epsilon \tag{3}$$

  In the remaining of the paper, we use the terminology *ϵ-precision* to call the precision requirement in the above equation.

How to compare the efficiency of Lasso-algorithms?

We use *order of complexity* to measure the efficiency of the algorithms.

- Definition: the order of operations needed to achieve the ϵ-precision. Mathematically speaking, in the framework of our paper, order of complexity is a function of the sample size n, the dimension p, the precision ϵ and other relevant factors.
- Reasons:
  1. the order of complexity for algorithms to achieve the same ϵ-precision may be different even for the same k, because the number of operations varies for one iteration in different algorithms.
  2. more and more algorithms nowadays involves more than one loop. Therefore, the meaning of k can be ambiguous, either it can refers to the outer loop, or to the inner loop.

Georgia Tech

Objective

- **Objective:**
  The objective of our paper is the proposal of a new Lasso-algorithm.
- **Contribution:**
  The strength of our proposed algorithm is that, to arrive at the same ϵ-precision, our algorithm owns less order of complexity than the state-of-art Lasso-algorithms.
- **How to calculate the order of complexity?**
  The order of complexity is decided by three factors:
  1. the number of total iterations to achieve the ϵ-precision,
  2. the number of operations in each iteration,
  3. the desired precision ϵ.

Remark

It is worth noting that, in our paper, we assume that the basic operations – such as addition, subtraction, multiplication, division, and comparison – is equal.

Georgia Tech

State-of-the-Art Lasso Algorithms

Iterative Shrinkage-Thresholding Algorithms (ISTA)

- What can ISTA solve?
  ISTA aims at the minimization of a summation of two functions, $g + f$, where the first function $g : \mathbb{R}^p \to \mathbb{R}$ is continuous convex and the other function $f : \mathbb{R}^p \to \mathbb{R}$ is smooth convex with Lipschitz continuous gradient.
- Lasso is a specially case of ISTA.
  If we let $g(\beta) = \lambda\|\beta\|_1$ and $f(\beta) = \frac{1}{2n}\|Y - X\beta\|_2^2$ with Lipschitz continuous gradient L taking the largest eigenvalue of matrix $X'X/n$.
- What is the updating rule from $\beta^{(k)}$ to $\beta^{(k+1)}$, i.e., $\beta^{(k)} \to \beta^{(k+1)}$?
  It is realized by updating $\beta^{(k+1)}$ through the quadratic approximation function of $f(\beta)$ at value $\beta^{(k)}$:

$$\beta^{(k+1)} = \arg\min_{\beta} f(\beta^{(k)}) + \langle(\beta - \beta^{(k)}), \nabla f(\beta^{(k)})\rangle + \frac{\sigma_{\max}(X'X/n)}{2}\|\beta - \beta^{(k)}\|_2^2 + \lambda\|\beta\|_1.$$

Simple algebra shows that (ignoring constant terms in β), minimization of equation above is equal to the minimization of the following equation:

$$\beta^{(k+1)} = \arg\min_{\beta} \frac{\sigma_{\max}(X'X/n)}{2}\left\|\beta - (\beta^{(k)} - \frac{\frac{1}{n}(X'X\beta^{(k)} - X'y)}{\sigma_{\max}(X'X/n)})\right\|_2^2 + \lambda\|\beta\|_1,$$

where soft-thresholding function can be used.

Iterative Shrinkage-Thresholding Algorithms (ISTA)

Algorithm 1: Iterative Shrinkage-Thresholding Algorithms (ISTA)

Input: $y_{n\times 1}$, $X_{n\times p}$, $L = \sigma_{\max}(X'X/n)$
Output: an estimator of β satisfies the ϵ-precision, noted as $\beta^{(k)}$

initialization;
$\beta^{(0)}$, $k = 0$
while $F(\beta^{(k)}) - F(\widehat{\beta}) > \epsilon$ **do**
  $\beta^{(k+1)} = S(\beta^{(k)} - \frac{1}{nL}(X'X\beta^{(k)} - X'y), \lambda/L)$
  $k = k + 1$
end

- Number of operations in each iteration: $O(p^2)$
- Number of iterations needed to achieve the ϵ-precision: $\frac{\sigma_{\max}(X'X/n)\|\beta^{(0)} - \widehat{\beta}\|_2^2}{\epsilon}$ (Beck and Teboulle, 2009, Theorem 3.1)
- Order of complexity: $O(p^2/\epsilon)$

Fast Iterative Shrinkage-Thresholding Algorithms (FISTA)

- Motivation:
  Based on ISTA and AGD.
- Difference of ISTA and FISTA:
  FISTA employs the second-order Taylor expansion in equation (9) at the an auxiliary variable $\alpha^{(k)}$ to update from $\beta^{(k)}$ to $\beta^{(k+1)}$, i.e.,

$$\beta^{(k+1)} = \arg\min_{\alpha} f(\alpha^{(k)}) + \langle (\alpha - \alpha^{(k)}), \nabla f(\alpha^{(k)}) \rangle + \frac{\sigma_{\max}(X'X/n)}{2} \|\alpha - \alpha^{(k)}\|_2^2 + \lambda \|\alpha\|_1,$$

  where $\alpha^{(k)}$ is a specific linear combination of the previous two estimator $\beta^{(k-1)}, \beta^{(k-2)}$, i.e., we have $\alpha^{(k)} = \beta^{(k-1)} + \frac{t_{k-1}-1}{t_k}(\beta^{(k-1)} - \beta^{(k-2)})$.

FISTA falls in the framework of Accelerate Gradient Descent(AGD), because it takes additional past information to take an extra gradient step via the auxiliary sequence $\alpha^{(k)}$, which is constructed by adding a "momentum" term $\beta^{(k-1)} - \beta^{(k-2)}$ that incorporates the effect of second-order changes.

Fast Iterative Shrinkage-Thresholding Algorithms (FISTA)

Algorithm 2: Fast Iterative Shrinkage-Thresholding Algorithms (FISTA)

Input: $y_{n\times 1}$, $X_{n\times p}$, $L = \sigma_{\max}(X'X/n)$

Output: an estimator of β, noted as $\beta^{(k)}$, which satisfies the ϵ-precision.

initialization;

$\beta^{(0)}$, $t_1 = 1$, $k = 0$

while $F(\beta^{(k)}) - F(\widehat{\beta}) > \epsilon$ **do**

  $\beta^{(k)} = S(\alpha^{(k)} - \frac{1}{nL}(X'X\alpha^{(k)} - X'y), \lambda/L)$

  $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$

  $\alpha^{(k+1)} = \beta^{(k)} + \frac{t_k - 1}{t_{k+1}}(\beta^{(k)} - \beta^{(k-1)})$

  $k = k + 1$

end

- Number of operations in each iteration: $O(p^2)$
- Number of iterations needed to achieve the ϵ-precision: $\frac{2\sigma_{\max}(X'X/n)\|\beta^{(0)} - \widehat{\beta}\|_2^2}{\sqrt{\epsilon}}$ (Beck and Teboulle, 2009, Theorem 4.4)
- Order of complexity: $O(p^2/\sqrt{\epsilon})$

Georgia Tech

Coordinate Descent (CD)

- Difference between CD and the two previous algorithms:
  The updating rule in both ISTA and FISTA is global. In contrast, Friedman et al. (2010) proposed a Lasso-algorithm which cyclically chooses one entry at a time and performs a simple analytical update through coordinate gradient descent.
- What is the updating rule from $\beta^{(k)}$ to $\beta^{(k+1)}$, i.e., $\beta^{(k)} \rightarrow \beta^{(k+1)}$?
  The updating rule from $\beta^{(k)}$ to $\beta^{(k+1)}$ in CD is that, it partially optimizes with respect to only j-th entry of $\beta^{(k+1)}$, $(j = 1 \cdots p)$, where the gradient at $\beta_j^{(k)}$ in the following equation is used for the updating process.

$$\frac{\partial}{\partial \beta_j} F(\beta^{(k)}) = \frac{1}{n}\left(e_j' X' X \beta^{(k)} - y' X e_j\right) + \lambda \mathrm{sign}(\beta_j)$$

Coordinate Descent (CD)

Algorithm 3: Coordinate Descent(CD)

Input: $y_{n\times 1}$, $X_{n\times p}$, λ

Output: an estimator of β, noted as $\beta^{(k)}$, which satisfies the ϵ-precision.

initialization;

$\beta^{(0)}$, $k = 0$

while $F(\beta^{(k)}) - F(\widehat{\beta}) > \epsilon$ **do**

  for $j = 1 \cdots p$ **do**

    $$\beta_j^{(k+1)} = S\left(y'Xe_j - \sum_{l\neq j}\left(X'X\right)_{jl}\beta_k^{(k)}, n\lambda\right) / \left(X'X\right)_{jj}$$

  end

end

- Number of operations in each iteration:
  $O(p^2)$
- Number of iterations needed to achieve the ϵ-precision:
  $\frac{4\sigma_{\max}(X'X/n)(1+p)\|\beta^{(0)}-\widehat{\beta}\|_2^2}{\epsilon} - \frac{8}{p}$
  ((Beck and Tetruashvili, 2013, Corollary 3.8)
- Order of complexity:
  $O(p^2/\epsilon)$

Smooth Lasso (SL)

- Main idea:
  It use a smooth function—$\phi_\alpha(u) = \frac{2}{u}\log(1 + e^{\alpha u}) - u$— to approximate ℓ_1 penalty, and Accelerated Gradient Descent(AGD) algorithm is applied after the replacement.

Smooth Lasso (SL)

Algorithm 4: Smooth Lasso (SL)

Input: $y_{n\times 1}$, $X_{n\times p}$,

$$\mu = \left[\sigma^2_{\max}(X/\sqrt{n}) + \lambda\alpha/2\right]^{-1}$$

Output: an estimator of β, noted as $\beta^{(k)}$, which satisfies the ϵ-precision.

initialization;

$\beta^{(0)}$, $k = 0$

while $F(\beta^{(k)}) - F(\widehat{\beta}) > \epsilon$ **do**

  $w^{(k+1)} = \beta^{(k)} + \frac{k-2}{k+1}(\beta^{(k)} - \beta^{(k-1)})$

  $\beta^{(k+1)} = w^{(k+1)} - \mu\nabla F_\alpha(w^{(k)})$

  $k = k + 1$

end

- Number of operations in each iteration: $O(p^2)$
- Number of iterations needed to achieve the ϵ-precision: $O(1/\epsilon)$ (Mukherjee and Seelamantula (2016))
- Order of complexity: $O(p^2/\epsilon)$

Comparison between the state-of-the-art algorithms with ours

Table: Comparison of order of complexity between the existing Lasso-algorithms and ours

method	ISTA	FISTA	CD	SL	our's
runtime	$O(p^2(1/\epsilon))$	$O(p^2(1/\sqrt{\epsilon}))$	$O(p^2(1/\epsilon))$	$O(p^2(1/\epsilon))$	$O\left(p^2 \log(1/\epsilon)\right)$

Our proposed Algorithm

Motivation

- It is widely acknowledge that, if the objective function is strongly convex, then the gradient descent can achieve very fast convergence rate.
- It is also known that the ℓ_1 norm ($\|\beta\|_1$ in our paper) is not stronly convex, while the quadratic function (such as $\|\beta\|_2^2$) can be easily proved to be strongly convex.
- Motivated by the poor behavior of the ℓ_1 penalty ($\|\beta\|_1$) around 0, we design this algorithm by solving this issue through gradually modifying the ℓ_1 norm around 0 (starting with a large modification, and slowly reducing it).

Motivated by these three facts, we try to replace the ℓ_1 penalty ($\|\beta\|_1$ in $F(\beta) = \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1$ by $f_t(\beta)$), which is quadratic near 0 and almost linear outside.

Condition for the surrogate function

To narrow down the difference between the surrogate ($F_t(\beta)$) and the origins ($F(\beta)$), $f_t(\beta)$ is designed to get more and more close to the ℓ_1 penalty when t approaches 0, i.e., $t \to 0$. The condition below lists all the requirement of $f_t(x)$.

Condition

Assume function $f_t(x)$ satisfies the following conditions.

1. When $t \to 0$, we have $f_0(x) = |x|$, where $|x|$ is the absolute value function.
2. For fixed $t > 0$, function $x \mapsto f_t(x)$ is quadratic on $[-t, t]$, here $\mapsto$ indicates that the left hand side (i.e., x) is the variable in the function in the right hand side (i.e., $f_t(x)$). We following this convention in the rest of this paper.
3. Functions $x \mapsto f_t(x)$ and $t \mapsto f_t(x)$ are C^1. The meaning of C_1 is the set of all continuously differentiable functions.

The Designed surrogate function

- Following the requirements in condition we mentioned, we design $f_t(x)$ in the following equation, where the input variable x is a scalar.

$$f_t(x) = \begin{cases} \frac{1}{3t^3}\left[\log(1+t)\right]^2 x^2, & \text{if } |x| \le t, \\ \left[\frac{\log(1+t)}{t}\right]^2 |x| + \frac{1}{3|x|}\left[\log(1+t)\right]^2 - \frac{1}{t}\left[\log(1+t)\right]^2, & \text{otherwise.} \end{cases} \tag{4}$$

- After the replacement of the surrogate function, we have $F_t(\beta) = \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda f_t(\beta)$

The Designed surrogate function

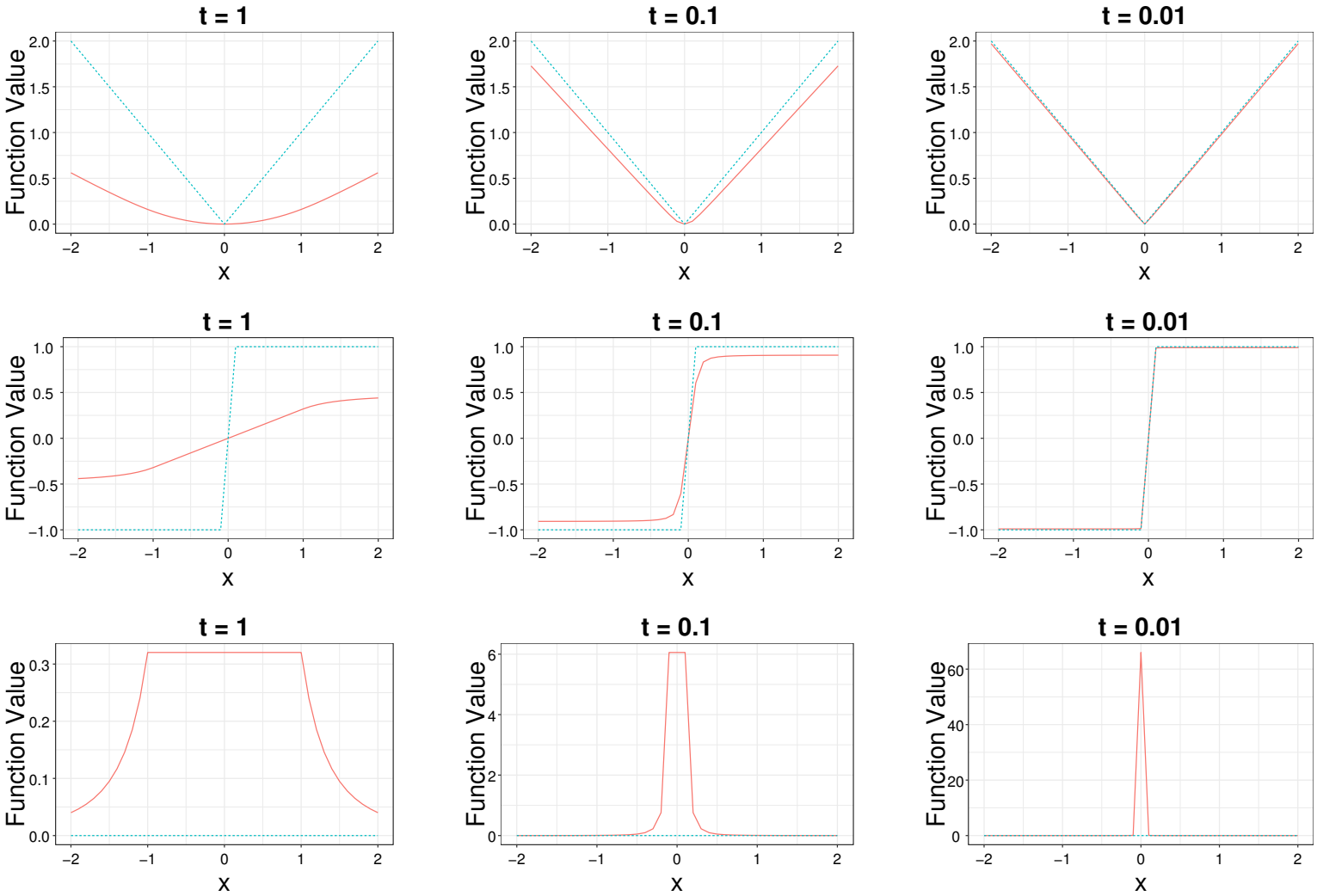

Figure: Closeness between $f_t(x)$ and $|x|$ when t changes (the first row), the closeness of their first derivative (the second row) and the closeness of their second derivative (the third row),

Algorithm

- Main idea: Replace $|\beta_i|$ by $f_t(\beta_i)$.
- Pseudo code:

Algorithm 5: Pseudo code for our algorithm

Input: $y, X, \lambda, t_0 > t_1 > t_2 > \ldots t_k > \ldots 0$
Output: an estimator of β, noted as $\beta^{(k)}$, which satisfies the ϵ-precision.

```
initialization β^(0)
for k = 0, 1, 2, 3, ... do
    Use AGD to minimize F_{t_k}(β) up to precision ε̃_k.
end
```

Algorithm

Algorithm 6: Estimation of β by HS algorithm

Input: $y_{n\times 1}$, $X_{n\times p}$, λ, t_0, h_0, μ, v

Output: an estimator of β, noted as $\beta^{(k)}$, which satisfies the ϵ-precision.

initialization $t_0, k = 1, \beta^{(0)} = \left[X'X + \frac{2n\lambda\left[\log(1+t_0)\right]^2}{3t_0^2} I\right]^{-1} X'y$

▶ *Outer-Iteration:* ◀ **while** $F(\beta^{(k)}) - F(\widehat{\beta}) > \epsilon$ **do**

  $s = 1$

  $\theta_0 = 1$

  $\alpha^{(k)[0]} = \beta^{(k-1)}$

  $\beta^{(k)[0]} = \beta^{(k-1)}$

  ▶ *Inner-Iteration:* ◀ **while** $F_{t_k}(\beta^{(k)[i+1]}) - F_{t_k}(\widehat{\beta}^{(k)}) > \widetilde{\epsilon}_k$ **do**

    $\theta_s = \frac{1+\sqrt{1+4\theta_{s-1}^2}}{2}$

    $\gamma_s = \frac{1-\theta_{s-1}}{\theta_s}$

    $\alpha^{(k)[s]} = \beta^{(k)[s-1]} - \mu \frac{\partial}{\partial \beta^{(k)[s-1]}} F_{t_k}(\beta^{(k)[s-1]})$

    $\beta^{(k)[i]} = (1-\gamma_s)\alpha^{(k)[s]} + \gamma_s \alpha^{(k)[s-1]}$

    $s = s + 1$

  end

  $\beta^{(k)} = \beta^{(k)[s]}$

  $t_{k+1} = t_k(1 - h_k)$

  $h_{k+1} = vh_k$

  $k = k + 1$

end

Initial Point

We need a well-designed initial point t_0, because starting with large t_0 ends up more shrinkage steps, which in turns costs more operations. To save operation times, we derive a minimal value of t_0 to start with in the following lemma.

Lemma

If one choose

$$t_0 = \inf_t \left\{ t : \left| \sum_{i=1}^{p} \left[X'X + \frac{2n\lambda \left[\log(1+t)\right]^2}{3t^2} I \right]^{-1}_{ij} (X'y)_j \right| \leq t \right\},$$

and accordingly

$$\beta^{(0)} = \left[X'X + \frac{2n\lambda \left[\log(1+t_0)\right]^2}{3t_0^2} I \right]^{-1} X'y, \tag{5}$$

then we have, for any $1 \leq j \leq p$, $\left|\beta_j^{(0)}\right| \leq t_0$.

Order of Complexity of Our Algorithm

Recall that, the order or complexity is affected by the following three factors.

1. Number of operations in each inner-iteration
2. Number of inner-iteration
3. Number of outer-iteration

And we will discuss these three factors one by one.

Number of operations in each inner-iteration

Algorithm 7: Estimation of β by HS algorithm

Input: $y_{n\times 1}$, $X_{n\times p}$, λ, t_0, h_0, μ, v

Output: an estimator of β, noted as $\beta^{(k)}$, which satisfies the ϵ-precision.

initialization t_0, $k = 1$, $\beta^{(0)} = \left[X'X + \frac{2n\lambda\left[\log(1+t_0)\right]^2}{3t_0^2} I\right]^{-1} X'y$

▶ ***Outer-Iteration:*** ◀ **while** $F(\beta^{(k)}) - F(\widehat{\beta}) > \epsilon$ **do**

  $s = 1$, $\theta_0 = 1$

  $\alpha^{(k)}[0] = \beta^{(k-1)}$, $\beta^{(k)}[0] = \beta^{(k-1)}$

  ▶ ***Inner-Iteration:*** ◀ **while** $F_{t_k}(\beta^{(k)[i+1]}) - F_{t_k}(\widehat{\beta}^{(k)}) > \widetilde{\epsilon}_k$ **do**

    $\theta_s = \frac{1+\sqrt{1+4\theta_{s-1}^2}}{2}$

    $\gamma_s = \frac{1-\theta_{s-1}}{\theta_s}$

    $\alpha^{(k)}[s] = \beta^{(k)[s-1]} - \mu \frac{\partial}{\partial \beta^{(k)[s-1]}} F_{t_k}(\beta^{(k)[s-1]})$

    $\beta^{(k)[i]} = (1-\gamma_s)\alpha^{(k)}[s] + \gamma_s \alpha^{(k)}[s-1]$

    $s = s + 1$

  end

  $\beta^{(k)} = \beta^{(k)}[s]$

  $t_{k+1} = t_k(1 - h_k)$

  $h_{k+1} = vh_k$

  $k = k + 1$

end

Conclusion: Number of operations in each inner-iteration is of order $O(p^2)$.

Number of Inner-iteration

Theorem (Inner-Iteration)

For k-th outer-iteration of our algorithm, with the following conditions:

1. *t stops shrinkage when it is smaller than τ.*
2. *Suppose that, from the begining of the HS algorithm to the end of the algorithm, for any $i \in \{1, 2 \dots p\}$ and $k \in \{1, 2 \dots\}$, we have that $\beta_i^{(k)} \leq B$.*

Then the inner-iteration outputs $\beta^{(k)}$ such that

$$F_{t_k}(\beta^{(k)}) - F_{t_k}(\widehat{\beta}^{(k)}) \leq \widetilde{\epsilon}_k$$

in $O(\log(1/\widetilde{\epsilon}_k))$

Georgia Tech

Number of outer-iteration

Theorem (Number of outer-iteration)

With the following conditions and the conditions in the previous theorem satisfies,

1. t_0 *is chosen according to Lemma 3.*
2. *For* $i = 1, 2, \ldots$, $t_k = t_{k-1}(1 - h_{k-1})$, *and* $h_k = vh_{k-1}$, *where* $0 < v < 1$.
3. *The precision of AGD in minimizing function* $F_{t_k}(\beta)$ *is set as* $\widetilde{\epsilon}_k = \frac{\lambda p}{3B}[\log(1 + t_k)]^2$.
4. $\lambda p t_0(2B + 1) \geq \epsilon \geq \lambda p t_0(2B + 1) \exp\left(-\frac{h_0}{1-v}\right)$.

The HS algorithm finds a point $\beta^{(k)}$ *such that*

$$F(\beta^{(k)}) - F(\widehat{\beta}) \leq \epsilon$$

in $k = \frac{1}{h_0} \log\left(\frac{\lambda p t_0(2B+1)}{\epsilon}\right)$ *outer-iterations.*

Order of Complexity

Theorem (Main Theory)

With the conditions listed in the previous two theorem satisfy, we can find $\beta^{(k)}$ *such that*

$$F(\beta^{(k)}) - F(\widehat{\beta}) \leq \epsilon$$

with order of complexity $p^2 O\left(\left[\frac{1}{h_0}\log\left(\frac{\lambda p t_0 (2B+1)}{\epsilon}\right)\right]^2\right)$.

Numerical Example

Data Generation

- We generated Gaussian data with n observations and p predictors, with each pair of predictors noted as X_j, and the explanatory matrix is noted as $X = (X_1 \cdots X_j \cdots X_p)$.
- Here we assume that X_j having the same population correlation ρ and we tried a number of ρ varying from 0 to 0.9. (We choose $\rho = 0.5$ in this section.)
- Accordingly, the outcome values were generated by

$$y = X\beta + qz.$$

  where the i-th ($1 \leq i \leq p$) entry in vector $\beta = (\beta_1 \cdots \beta_p)'$ is generated by $\beta_i = (-1)^i \exp\left(-2(i-1)/20\right)$, which are constructed to have alternating signs and to be exponentially decreasing.
- Besides, $z = (z_1 \cdots z_p)'$ is the white noise with normal distribution of z_i as $N(0, 1)$ and q is chosen so that the signal-to-noise ratio is 3.0.

Simulation Result

Table: Number of Operations to achieve the different ϵ

method	Precision ϵ 10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
ISTA	0.3599	8.7971	343.3121	1881.4000	6430.9977	16327.7063
	(0.86464)	(49.8660)	(3954.3000)	(13234.0000)	(4981.1932)	(123430.03991)
FISTA	0.11279	0.2462	0.3700	1.6157	8.8244	47.0859
	(0.2421)	(0.6350)	(0.8089)	(5.2405)	(47.0446)	(267.1121)
Ours	0.01184	0.0165	0.0207	0.0327	0.0656	0.0871
	(0.0159)	(0.0260)	(0.0373)	(0.0527)	(0.0813)	(0.1210)

[1] The unit of the number of operations are in 10^7, and $n = 50$, $p = 20$.

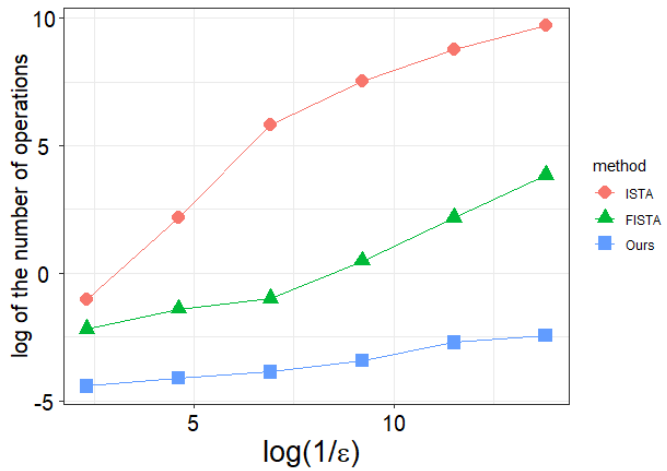

Figure: Number of Operations under different ϵ

The End