Fast Newton Hard Thresholding Pursuit for Sparsity Constrained Nonconvex Optimization

Jinghui Chen
Department of Computer Science
University of Virginia
Charlottesville, Virginia, USA 22904
jc4zg@virginia.edu

Quanquan Gu
Department of Computer Science
University of Virginia
Charlottesville, Virginia, USA 22904
qg5w@virginia.edu

ABSTRACT

We propose a fast Newton hard thresholding pursuit algorithm for sparsity constrained nonconvex optimization. Our proposed algorithm reduces the per-iteration time complexity to linear in the data dimension d compared with cubic time complexity in Newton's method, while preserving faster computational and statistical convergence rates. In particular, we prove that the proposed algorithm converges to the unknown sparse model parameter at a composite rate, namely quadratic at first and linear when it gets close to the true parameter, up to the minimax optimal statistical precision of the underlying model. Thorough experiments on both synthetic and real datasets demonstrate that our algorithm outperforms the state-of-the-art optimization algorithms for sparsity constrained optimization.

KEYWORDS

Sparse Learning; Sparsity Constrained Optimization; Nonconvex Optimization; Newton's Method

ACM Reference format:
Jinghui Chen and Quanquan Gu. 2017. Fast Newton Hard Thresholding Pursuit for Sparsity Constrained Nonconvex Optimization. In *Proceedings of KDD'17, August 13-17, 2017, Halifax, NS, Canada, ,* 10 pages.
DOI: 10.1145/3097983.3098165

1 INTRODUCTION

High-dimensional data, where the dimension is comparable to or even larger than the sample size, are ubiquitous in modern Big Data applications, ranging from texts, images, videos, to electronic medical records, genomics data and neuroscience data. In order to learn from high-dimensional data, it is crucial to exploit the intrinsic low-dimensional structure of the data, which is often achieved by imposing certain structural assumptions on the parameter of the underlying model. In the past decade, various types of model structures [30, 31] have been investigated, such as sparse vectors and low-rank matrices, among which sparsity is probably the most acknowledged one. Therefore, we will focus on sparse estimation in this paper, while the proposed algorithm and theory can potentially be extended to other structural estimation problems as well.

The high-dimensional sparse estimation problem is often cast as ℓ_1 norm regularized estimation methods such as Lasso [45] and ℓ_1 norm regularized logistic regression [21, 31]. However, recent studies [12, 15, 47, 48, 54, 57] showed that ℓ_1 norm regularization tends to cause large estimation bias, and incur worse empirical performance than the vanilla ℓ_0 pseudo norm regularization or constraint. This motivates a family of sparsity (i.e., ℓ_0 pseudo norm) constrained optimization problems as follows:

$$\min_{\boldsymbol{\beta}} F(\boldsymbol{\beta}) \quad \text{subject to} \quad \|\boldsymbol{\beta}\|_0 \le s, \tag{1.1}$$

where $F(\boldsymbol{\beta}) = n^{-1}\sum_{i=1}^{n} f_i(\boldsymbol{\beta})$ is a loss function, which can be written as the sum of a finite number of convex and smooth component functions, $\|\boldsymbol{\beta}\|_0$ is the number of nonzero elements in $\boldsymbol{\beta} \in \mathbb{R}^d$, and s is a tuning parameter that controls the sparsity of $\boldsymbol{\beta}$. Due to the nonconvexity of the ℓ_0 norm constraint, the problem in (1.1) is nonconvex and in general NP hard. Thus, it is much more challenging than solving ℓ_1 norm regularized estimation problems [21, 31, 45]. In order to obtain an approximate solution to (1.1), many first-order optimization algorithms have been developed. For example, when the objective function $F(\boldsymbol{\beta})$ is the square loss function, it can be solved approximately by matching pursuit [26], orthogonal matching pursuit [46], CoSaMP [29], hard thresholding pursuit [13] and iterative hard thresholding [6]. For general loss functions, there also exists a set of algorithms such as forward feature selection [4, 41], forward backward feature selection [44, 55], and iterative gradient hard thresholding [16, 51]. Several stochastic algorithms [8, 24, 34] are also proposed to solve this problem.

While the aforementioned first-order methods, including both deterministic and stochastic ones, have been extensively studied for solving the sparsity constrained optimization in (1.1), second-order algorithms such as Newton's method and its variants received much less attention. To the best of our knowledge, the only existing second-order method along this line is Newton greedy pursuit [52], which borrows the idea of gradient hard thresholding algorithm [51] to Newton's method. It achieves a quadratic rate of convergence in optimization and is demonstrated to be able to reduce the per iteration complexity. However, its per iteration complexity is still quite high, because in each iteration, it needs to solve a fairly complex subproblem using another iterative optimization procedure. To this end, efficient and practical second-order algorithms for solving the nonconvex optimization problem in (1.1) are still in demand.

In this paper, we aim to develop a fast second-order algorithm for sparsity constrained optimization in (1.1), which is able to achieve not only fast rates of convergence in both optimization and statistical estimation, but also a low per-iteration complexity. In detail, we adapt an unbiased stochastic inverse Hessian estimator [2] and propose a fast Newton hard thresholding pursuit (FNHTP) based on iterative hard thresholding. More specifically, instead of calculating the Hessian and its inverse directly, our algorithm estimates the inverse of the Hessian in a stochastic and recursive way. Therefore, the proposed algorithm significantly reduces the per iteration complexity from the typical $O(nd^2 + d^3)$ to $O(s \cdot K_1 K_2 d)$, where $K_1 \gtrsim \log d$ and K_2 is in the same order as the restricted condition number of the Hessian of $F(\boldsymbol{\beta})$ in (1.1), which will be discussed in the theoretical analysis later. In other words, the per iteration complexity of our algorithm is linear in the data dimension d, which almost matches the per iteration complexity of first-order methods. In the meantime, we show that the proposed algorithm enjoys a composite convergence rate, which starts with a quadratic rate and later transforms into a linear rate when it gets close to the unknown true parameter. This together with the $O(d)$ per iteration complexity enables our algorithm to be faster than the aforementioned first-order greedy methods. Furthermore, the estimator returned by our algorithm converges to the unknown true parameter up to the minimax optimal statistical error rate for a broad family of sparse statistical models such as sparse linear regression and sparse generalized linear models. We compare our algorithm with the state-of-the-art baseline algorithms on both synthetic and real high-dimensional datasets, and the experimental results verify the superiority of our algorithm against the existing best algorithms.

The remainder of this paper is organized as follows. In Section 2, we briefly review related work. In Section 3, we review two illustrative examples of the sparsity constrained optimization. We present our algorithm in Section 4, and analyze it in Section 5. Then we apply our algorithm to the two specific examples and illustrate the corresponding theory in Section 6. We present the experimental results in Section 7. Finally, we conclude this paper in Section 8.

Notation We use $[n]$ to denote the index set $\{1, \ldots, n\}$. Let $\mathbf{A} = [A_{ij}] \in \mathbb{R}^{d\times d}$ be a matrix and $\mathbf{x} = [x_1, \ldots, x_d]^\top \in \mathbb{R}^d$ be vector. Denote the i-th row of $\mathbf{A}$ as $\mathbf{A}_{i*}$ and the j-th column of $\mathbf{A}$ as $\mathbf{A}_{*j}$. For $0 < q < \infty$, we define the ℓ_0, ℓ_q and ℓ_∞ vector norms as $\|\mathbf{x}\|_0 = \sum_{i=1}^d \mathbb{1}(x_i \neq 0)$, $\|\mathbf{x}\|_q = \left(\sum_{i=1}^d |x_i|^q\right)^{1/q}$, and $\|\mathbf{x}\|_\infty = \max_{1\le i\le d} |x_i|$, where $\mathbb{1}(\cdot)$ represents the indicator function. For a vector $\mathbf{x}$, we define $\text{supp}(\mathbf{x})$ as the index set of nonzero entries of $\mathbf{x}$, and $\text{supp}(\mathbf{x}, s)$ as the index set of the top s entries of $\mathbf{x}$ in terms of magnitude. In addition, we denote by $\mathbf{x}_S$ the restriction of $\mathbf{x}$ onto a index set S, such that $[\mathbf{x}_S]_i = x_i$ if $i \in S$, and $[\mathbf{x}_S]_i = 0$ if $i \notin S$. Also for matrix $\mathbf{A}$, we denote $\mathbf{A}_S$ as the restriction of $\mathbf{A}$ to index set $\widetilde{S}$, i.e., $[[A]_S]_{ij} = [A]_{ij}$ if $(i, j) \in \widetilde{S}$, and $[[A]_S]_{ij} = 0$ otherwise. Furthermore, we denote by $\mathbf{x}_s$ the restriction of $\mathbf{x}$ onto the top s entries in terms of magnitude, i.e., $[\mathbf{x}_s]_i = x_i$ if $i \in \text{supp}(\mathbf{x}, s)$, and $[\mathbf{x}_s]_i = 0$ if $i \notin \text{supp}(\mathbf{x}, s)$. Also, we denote by $\nabla^{-2}F(\boldsymbol{\beta})$ the inverse of the Hessian matrix $\nabla^2 F(\boldsymbol{\beta})$.

2 RELATED WORK

First-order Methods: For general-purpose convex optimization, gradient descent attains a linear rate of convergence under certain conditions. However, when the number of component functions n is large, gradient descent can be computationally expensive at each iteration, because the time complexity for calculating the full gradient is linear in n. To overcome this problem, stochastic gradient descent is often used. The time complexity for calculating a stochastic gradient descent is independent of n. Nevertheless, due to the undiminished variance of stochastic gradient descent, it has a sublinear rate of convergence . To accelerate stochastic gradient descent, Johnson and Zhang [18] proposed stochastic variance reduced gradient (SVRG), which achieves the best of both worlds: linear rate of convergence and low per-iteration complexity. Other variance reduction based first-order algorithms include [9, 40, 42, 53], to mention a few. For general-purpose nonconvex optimization, gradient descent algorithm [32] and its stochastic variants [3, 14, 38] achieve sublinear convergence rate to the stationary point. Unlike general nonconvex optimization where the objective function is nonconvex, the objective function in (1.1) is convex, and the nonconvexity is only introduced by the sparsity constraint. Based on this observation, Yuan et al. [51] proposed a gradient hard thresholding pursuit algorithm for solving the nonconvex problem in (1.1), which achieves the linear rate of convergence. Similar algorithm was proposed and analyzed in [16]. In [24], iterative hard thresholding is used together with SVRG to achieve a better iteration complexity for high-dimensional sparse estimation. In [8], a randomized block coordinate descent algorithm together with iterative hard thresholding and variance reduction is proposed for sparsity constrained optimization. While the first-order algorithms do enjoy light per-iteration computational cost, their convergence rates cannot be better than linear.

Second-order Methods: In contrast, second-order algorithms generally achieve faster convergence rates than first-order methods. For instance, Newton's method is probably the most well-recognized second-order method. It achieves a quadratic rate of convergence under some certain conditions, which is significantly advantageous over first-order methods. For a long time, the major obstacle of applying Newton's method is the high computational costs. It requires $O(nd^2 + d^3)$ per iteration time complexity, where the first term reflects the cost for computing the Hessian matrix and the second term represents the cost for inversing the Hessian matrix. Therefore, Newton's method is computationally intractable for high dimensional problems with big d.

In order to reduce the per iteration complexity of the Newton's method while still preserving a relatively fast convergence rate, various Quasi-Newton methods have been proposed in the past decades, in which the gradients and the intermediate iterates are used to approximate the inverse Hessian matrix, leading to a less expensive update in each iteration. A celebrated Quasi-Newton method is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, which requires $O(nd + d^2)$ per-iteration cost [5]. Recently, Erdogdu and Montanari [11] proposed a new sub-sampled Newton method via rank thresholding. Its per iteration complexity is reduced to $O(nd + d^2|\mathcal{S}|)$, where $|\mathcal{S}|$ is the subsample size. Similar methods and follow-up work have been proposed in [7, 10, 39, 49, 50]. Very recently, Agarwal et al. [2] proposed an unbiased stochastic inverse Hessian estimator, which reduces the per-iteration complexity to be linear in the data dimension. Mutny [28] gave a new analysis of the algorithm proposed in [2]. In [27], a new stochastic L-BFGS algorithm is proposed, which attains a linear rate of convergence. However, all these methods are only applicable to the classical regime ($n \gg d$), rather than the high dimensional regime ($n \ll d$). For general nonconvex optimization, several second-order optimization algorithms [1, 33] have been proposed, which are guaranteed to converge to a local minimum. By leveraging the convexity of the objective function, Yuan and Liu [52] proposed Newton greedy pursuit , which combines the idea of gradient hard thresholding algorithm [51] and Newton's method. While it achieves quadratic rate of convergence to the unknown sparse model parameter, its computational complexity is still quite high, and not scalable to high-dimensional data. Our proposed algorithm falls into the category of second-order methods, and its per-iteration complexity is linear in the data dimension. Therefore, it can be scaled up to high-dimensional data.

3 ILLUSTRATIVE EXAMPLES OF SPARSITY CONSTRAINED OPTIMIZATION

In this section, we give two examples of sparse estimation problems, which fall in the sparsity constrained optimization problem in (1.1). We will later demonstrate the implication of our general algorithm and theory to these examples in Section 6.

Example 3.1 (Sparse Linear Regression). Consider the following linear regression model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon}, \tag{3.1}$$

where $\mathbf{y} \in \mathbb{R}^N$ denotes a vector of the responses, and $\mathbf{X} \in \mathbb{R}^{N\times d}$ is the design matrix, $\boldsymbol{\beta}^* \in \mathbb{R}^d$ is the unknown regression coefficient vector such that $\|\boldsymbol{\beta}^*\|_0 \le s^*$, and $\boldsymbol{\epsilon} \in \mathbb{R}^N$ is a zero mean sub-Gaussian noise vector. A commonly used estimator for the above sparse linear regression problem is the Lasso estimator [45] with ℓ_1 norm penalty. An alternative estimator is the sparsity constrained estimator

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \frac{1}{2N}\|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2 \quad \text{subject to} \quad \|\boldsymbol{\beta}\|_0 \le s, \tag{3.2}$$

where s is a tuning parameter, which controls the sparsity of $\boldsymbol{\beta}$. This is indeed an example of the nonconvex optimization problem in (1.1), where $F(\boldsymbol{\beta}) = 1/(2N)\|\mathbf{X}\boldsymbol{\beta}-\mathbf{y}\|_2^2$, $f_i(\boldsymbol{\beta}) = 1/(2b)\sum_{i_k\in B_i}(\mathbf{x}_{i_k}^\top\boldsymbol{\beta} - y_{i_k})^2$ where B_i stands for the i-th mini-batch, $b = |B_i|$ is the size of the mini-batch, $\mathbf{x}_{i_k} \in \mathbb{R}^d$ is the i_k-th row of $\mathbf{X}$ and y_{i_k} is the i_k-th element of $\mathbf{y}$. Here we choose the mini-batch size b to be in the order of s. The reason will be clear in the theoretical analysis of our proposed algorithm. Similar estimator has been studied by [16, 46, 55], to mention a few.

Example 3.2 (Sparse Generalized Linear Models). We assume that the observations in each task are generated from generalized linear models

$$\mathbb{P}(y|\mathbf{x}, \boldsymbol{\beta}^*, \sigma) = \exp\left\{[y\langle\boldsymbol{\beta}^*, \mathbf{x}\rangle - \Phi(\boldsymbol{\beta}^{*\top}\mathbf{x})]/c(\sigma)\right\}, \tag{3.3}$$

where $\Phi(\cdot) : \mathbb{R} \to \mathbb{R}$ is a link function, $y \in \mathbb{R}$ is the response variable, $\mathbf{x} \in \mathbb{R}^d$ is the predictor vector, $\boldsymbol{\beta}^* \in \mathbb{R}^d$ is the parameter such that $\|\boldsymbol{\beta}^*\|_0 \le s^*$, and $c(\sigma) \in \mathbb{R}$ is fixed and known scale parameter. A special example of generalized linear model is the linear regression model where the noise follows from a Gaussian distribution, which corresponds to $c(\sigma) = \sigma^2$ and $\Phi(t) = t^2$. Logistic regression is another special case of the generalized linear model, where $\Phi(t) = \log(1 + \exp(t)), c(\sigma) = 1$ and $y \in \{0, 1\}$. Given $\{\mathbf{x}_i, y_i\}_{i=1}^N$, a widely used estimator for $\boldsymbol{\beta}^*$ is the ℓ_1 regularized maximum likelihood estimator [21, 25, 31]. An alternative estimator is the sparsity constrained maximum likelihood estimator as follows

$$\min_{\boldsymbol{\beta}} -\frac{1}{N}\sum_{i=1}^N \frac{1}{c(\sigma)}\left[y_i\langle\boldsymbol{\beta}, \mathbf{x}_i\rangle - \Phi(\boldsymbol{\beta}^{*\top}\mathbf{x})\right] \quad \text{subject to} \quad \|\boldsymbol{\beta}\|_0 \le s. \tag{3.4}$$

Following a similar construction of $f_i(\boldsymbol{\beta})$ as before, we can show that (3.4) also fits in the generic framework in (1.1). The estimator in (3.4) has been investigated by [17, 24, 51]. For more examples, please refer to [16, 51] and references therein.

4 THE PROPOSED ALGORITHM

Standard Newton's method works by conducting the following update in each iteration:

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} - \nabla^{-2}F(\boldsymbol{\beta}^{(t)}) \cdot \nabla F(\boldsymbol{\beta}^{(t)}),$$

where $\nabla F(\boldsymbol{\beta}^{(t)})$ is the gradient of F at $\boldsymbol{\beta}^{(t)}$, $\nabla^{-2}F(\boldsymbol{\beta}^{(t)})$ is the inverse Hessian matrix. Due to the curvature information incorporated by the inverse Hessian matrix, the update of Newton's method is insensitive to the condition number of the Hessian and guides the quickest way descent by minimizing second-order Taylor expansion. However, computing the inverse of the Hessian matrix requires $O(d^3)$ operations, which is computationally intractable for the high dimensional data.

In order to address this problem, we adapt an unbiased inverse Hessian estimator from [2, 28], which is inspired by the von Neumann series in the following proposition.

Proposition 4.1. *For a matrix* $\mathbf{A} \in \mathbb{R}^{d\times d}$ *such that* $\mathbf{A} \succeq 0$ *and* $\|\mathbf{A}\|_2 \le 1$, *we have that,*

$$\mathbf{A}^{-1} = \textstyle\sum_{i=0}^\infty(\mathbf{I} - \mathbf{A})^i. \tag{4.1}$$

Our key idea is to use the series expansion in (4.1) to construct an unbiased estimator for the inverse Hessian matrix by the summation of the polynomial terms. Note that the series expansion in (4.1) can actually be computed in a recursive way. In particular, if we define $\mathbf{A}_j^{-1}$ as the sum of the first j terms in the series expansion of $\mathbf{A}^{-1}$, we can obtain the following recursive formula:

$$\mathbf{A}_j^{-1} = \textstyle\sum_{i=0}^j(\mathbf{I} - \mathbf{A})^i = \mathbf{I} + (\mathbf{I} - \mathbf{A})\mathbf{A}_{j-1}^{-1}.$$

As a result, $\mathbf{A}_j^{-1}$ can be computed recursively. In addition, when j is sufficiently large, $\mathbf{A}_j^{-1}$ will become an accurate approximation to $\mathbf{A}^{-1}$.

On the other hand, inspired by stochastic gradient descent (SGD) where the stochastic gradient $\nabla f_k(\boldsymbol{\beta})$ is sampled uniformly from $k \in [n]$ as an unbiased estimator for the full gradient $\nabla F(\boldsymbol{\beta})$, we use stochastic Hessian matrix $\nabla^2 f_k(\boldsymbol{\beta})$ drawn from $k \in [n]$ as an unbiased estimator for the full Hessian matrix $\nabla^2 F(\boldsymbol{\beta})$.

Putting the above pieces together, we introduce an unbiased estimator $\widetilde{\nabla}_j^{-2}F(\boldsymbol{\beta})$ for the inverse full Hessian matrix $\nabla^{-2}F(\boldsymbol{\beta})$ as follows.

Definition 4.2. Given j independent and unbiased samples of $\nabla^2 f_k(\boldsymbol{\beta}), k \in [n]$, $\widetilde{\nabla}_j^{-2}F(\boldsymbol{\beta})$ is defined in the following recursive way:

$$\widetilde{\nabla}_0^{-2}F(\boldsymbol{\beta}) = \mathbf{I},$$

$$\widetilde{\nabla}_j^{-2}F(\boldsymbol{\beta}) = \mathbf{I} + \Big(\mathbf{I} - \nabla^2 f_{k_j}(\boldsymbol{\beta})\Big)\widetilde{\nabla}_{j-1}^{-2}F(\boldsymbol{\beta}).$$

It can be shown that $\mathbb{E}\big[\widetilde{\nabla}_j^{-2}F(\boldsymbol{\beta})\big] = \nabla_j^{-2}F(\boldsymbol{\beta})$ and furthermore we have $\lim_{j\to\infty}\mathbb{E}\big[\widetilde{\nabla}_j^{-2}F(\boldsymbol{\beta})\big] = \nabla^{-2}F(\boldsymbol{\beta})$. In fact, it is not necessary to maintain the inverse Hessian matrix estimator directly, since only the product of the inverse Hessian matrix and the full gradient vector matters in the update of Newton's method. Therefore, we only need to maintain an estimator for the product of the inverse Hessian matrix and the full gradient. In detail, let $\mathbf{g}_j = \widetilde{\nabla}_j^{-2}F(\boldsymbol{\beta}) \cdot \nabla F(\boldsymbol{\beta})$, by invoking Definition 4.2, we can obtain the recursive definition of $\mathbf{g}_j$ as follows:

$$\begin{aligned}\mathbf{g}_j &= \nabla F(\boldsymbol{\beta}) + \Big(\mathbf{I} - \nabla^2 f_{k_j}(\boldsymbol{\beta})\Big)\mathbf{g}_{j-1}\\ &= \nabla F(\boldsymbol{\beta}) + \mathbf{g}_{j-1} - \nabla^2 f_{k_j}(\boldsymbol{\beta})\mathbf{g}_{j-1}. \qquad (4.2)\end{aligned}$$

It can also be shown that $\mathbb{E}[\mathbf{g}_j] = \nabla_j^{-2}F(\boldsymbol{\beta})\nabla F(\boldsymbol{\beta})$, and $\lim_{j\to\infty}\mathbb{E}\big[\mathbf{g}_j\big] = \nabla^{-2}F(\boldsymbol{\beta})\nabla F(\boldsymbol{\beta})$. This reduces the space complexity from $O(d^2)$ to $O(d)$.

Given an unbiased estimator for the product of the inverse Hessian matrix and the full gradient, we present a fast Newton hard thresholding pursuit (FNHTP) algorithm in Algorithm 1.

Algorithm 1 Fast Newton Hard Thresholding Pursuit (FNHTP)

```
1:  **Input:** $K_1, K_2$, sparsity $s$
2:  **Initialization:** $\boldsymbol{\beta}^{(0)}$ with $\|\boldsymbol{\beta}^{(0)}\|_0 \le s$
3:  for $t = 1, 2, \ldots T$ do
4:      $\mathbf{g} = \nabla F(\boldsymbol{\beta}^{(t)})$
5:      for $i = 1, 2, \ldots, K_1$ do
6:          $\mathbf{g}_0^i = \mathbf{g}$
7:          for $j = 1, 2, \ldots, K_2$ do
8:              Sample $\nabla^2 f_{k_j}(\boldsymbol{\beta}^{(t)})$ uniformly from $\{\nabla^2 f_k(\boldsymbol{\beta}^{(t)}) | k \in [n]\}$
9:              $\mathbf{g}_j^i = \nabla F(\boldsymbol{\beta}^{(t)}) + \mathbf{g}_{j-1}^i - \nabla^2 f_{k_j}(\boldsymbol{\beta}^{(t)}) \cdot \mathbf{g}_{j-1}^i$
10:         end for
11:     end for
12:     $\bar{\mathbf{g}} = 1/K_1\big(\sum_{i=1}^{K_1}\mathbf{g}_{K_2}^i\big)$
13:     $\boldsymbol{\beta}^{(t+0.5)} = \boldsymbol{\beta}^{(t)} - \bar{\mathbf{g}}$
14:     $\boldsymbol{\beta}^{(t+1)} = \mathrm{HT}\big(\boldsymbol{\beta}^{(t+0.5)}, s\big)$
15: end for
16: **Return:** $\boldsymbol{\beta}^{(T+1)}$
```

Note that in each iteration of Algorithm 1, we have two layers of loops. In the outer loop, $\mathbf{g}_0^i$ is set to be $\nabla F(\boldsymbol{\beta}^{(t)})$ as an initialization. In the inner loop, Algorithm 1 uniformly samples a stochastic Hessian matrix from $\{\nabla^2 f_k(\boldsymbol{\beta}^{(t)}) | k \in [n]\}$, which is an unbiased estimator of the full Hessian matrix $\nabla^2 F(\boldsymbol{\beta}^{(t)})$. It then updates the unbiased estimator for the product of inverse Hessian matrix and the full gradient vector recursively using (4.2) in the inner loop. After the outer loop is complete, our proposed algorithm computes the aggregated estimator $\bar{\mathbf{g}}$ for the product of inverse Hessian matrix and the full gradient, which is the averaged estimator over each iteration in the outer loop. And $\boldsymbol{\beta}^{(t+0.5)}$ is the output of our fast Newton update step. Since $\boldsymbol{\beta}^{(t+0.5)}$ is not necessarily sparse after the fast Newton update, in order to make it sparse, we apply a hard thresholding procedure [16, 51] right after fast Newton update step. The hard thresholding operator is defined as follows:

$$[\mathrm{HT}(\boldsymbol{\beta}, s)]_i = \begin{cases} \beta_i, & \text{if } i \in \mathrm{supp}(\boldsymbol{\beta}, s) \\ 0, & \text{otherwise} \end{cases}. \qquad (4.3)$$

The hard thresholding step preserves the entries of $\boldsymbol{\beta}^{(t+0.5)}$ with the top s large magnitudes and sets the rest to zero, where s is a tuning parameter that controls the sparsity level. This yields a sparse $\boldsymbol{\beta}^{(t+1)}$ with sparsity s.

Notice that the time complexity to compute $\nabla^2 f_{k_j}(\boldsymbol{\beta}^{(t)}) \cdot \mathbf{g}_{j-1}^i$ is not necessarily $O(d^2)$. The reason is, for many machine learning and statistics problems we are interested in, the stochastic Hessian matrix $\nabla^2 f_i(\boldsymbol{\beta})$ can be written as the summation of several outer product of two vectors. For example, in linear regression where $f_i(\boldsymbol{\beta}) = 1/(2b)\sum_{k\in B_i}(\boldsymbol{\beta}^\top \mathbf{x}_k - y_k)^2$, we have $\nabla^2 f_i(\boldsymbol{\beta}) = 1/b\sum_{k\in B_i}\mathbf{x}_k\mathbf{x}_k^\top$. Therefore, the time complexity to compute $\nabla^2 f_{k_j}(\boldsymbol{\beta}^{(t)}) \cdot \mathbf{g}_{j-1}^i$ can be reduced to $O(bd)$. This gives rise to the $O(bK_1K_2d)$ per iteration complexity for our proposed algorithm. Table 1 summarizes the comparison of the per iteration complexity for the state-of-the art sparsity constrained nonconvex optimization algorithms. Since we choose b to be in the order of s, as will be illustrated in our main theory, $O((N+bK_1K_2)d)$ is hence in the order of $O((N+sK_1\kappa_{\widetilde{s}}\log d)d)$, where $\kappa_{\widetilde{s}}$ is the restricted condition number of the Hessian. Therefore, the per iteration complexity of our algorithm is very similar to that of first-order algorithms. And in our experiment, we found that by setting $K_1 = 1$, we have already achieved a decent performance, which suggests that the per iteration complexity of our algorithm can be further reduced to $O((N + s\kappa_{\widetilde{s}}\log d)d)$ empirically.

Methods	Per Iteration Complexity
GraHTP	$O(Nd)$
SVRGHT	$O((N + \kappa_{\widetilde{s}})d)$
NTGP	$O((Nd + s^3)M)$
FNHTP (Ours)	$O((N + b \cdot K_1K_2)d)$

Table 1: A comparison of nonconvex optimization algorithms in terms of per iteration complexity ($N = bn$, where b is the minibatch size and n is the number of component functions, M is the number of inner loop for NTGP algorithm)

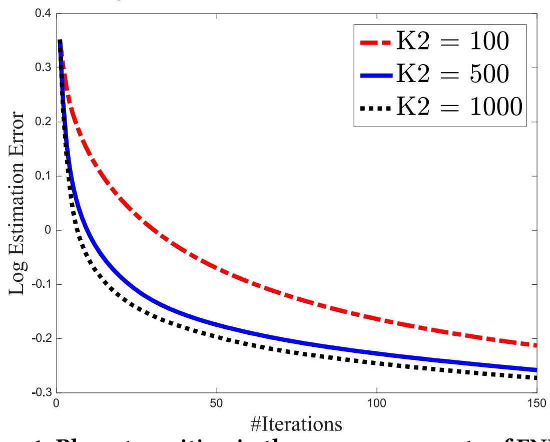


Figure 1: Phase transition in the convergence rate of FNHTP algorithm for sparse linear regression on a synthetic data.

5 MAIN THEORY

We first layout a set of definition and assumptions, that are essential for our main theory.

Definition 5.1 (Sparse Eigenvalues). Let s be a positive integer. The largest and smallest s-sparse eigenvalues of a Hessian matrix $\nabla^2 F(\boldsymbol{\beta})$ are

$$\rho^+(\widetilde{s}) = \sup_{\mathbf{v}} \Big\{ \mathbf{v}^\top \nabla^2 F(\boldsymbol{\beta}) \mathbf{v} : \|\mathbf{v}\|_0 \le \widetilde{s}, \|\mathbf{v}\|_2 = 1, \boldsymbol{\beta} \in \mathbb{R}^d \Big\},$$

$$\rho^-(\widetilde{s}) = \inf_{\mathbf{v}} \Big\{ \mathbf{v}^\top \nabla^2 F(\boldsymbol{\beta}) \mathbf{v} : \|\mathbf{v}\|_0 \le \widetilde{s}, \|\mathbf{v}\|_2 = 1, \boldsymbol{\beta} \in \mathbb{R}^d \Big\}.$$

Based on the sparse eigenvalues, we make the following assumptions on $F(\boldsymbol{\beta})$ and $f_i(\boldsymbol{\beta})$ with respect to $\rho^+(\widetilde{s})$ and $\rho^-(\widetilde{s})$ mentioned above.

Assumption 5.2. $F(\boldsymbol{\beta})$ satisfies restricted smoothness condition at sparsity level $\widetilde{s}$ with constants $\rho^+(\widetilde{s})$: for all $\boldsymbol{\beta}, \boldsymbol{\beta}'$ such that $\|\boldsymbol{\beta} - \boldsymbol{\beta}'\|_0 \le \widetilde{s}$, we have

$$F(\boldsymbol{\beta}) - F(\boldsymbol{\beta}') - \nabla F(\boldsymbol{\beta}')^\top (\boldsymbol{\beta} - \boldsymbol{\beta}') \le \frac{\rho^+(\widetilde{s})}{2} \|\boldsymbol{\beta} - \boldsymbol{\beta}'\|_2^2.$$

We further assume that

$$\rho^+(\widetilde{s}) \le L.$$

Recall that in order to apply Proposition 4.1 to approximate the inverse Hessian matrix, we require $L \le 1$. This condition seems to be strong but actually most of the problem can be rescaled such that this condition holds. Without loss of generality, in the rest of this paper, we assume that $L \le 1$ holds.

Further we need the assumption of restricted strong convexity for each $f_i(\cdot)$.

Assumption 5.3. $f_i(\boldsymbol{\beta})$ satisfies restricted strong convexity at sparsity level $\widetilde{s}$ with constants $\rho_i^-(\widetilde{s}) > 0$: for all $i = 1, \ldots, n$ and all $\boldsymbol{\beta}, \boldsymbol{\beta}'$ such that $\|\boldsymbol{\beta} - \boldsymbol{\beta}'\|_0 \le \widetilde{s}$, we have

$$f_i(\boldsymbol{\beta}) - f_i(\boldsymbol{\beta}') - \nabla f_i(\boldsymbol{\beta}')^\top (\boldsymbol{\beta} - \boldsymbol{\beta}') \ge \frac{\rho_i^-(\widetilde{s})}{2} \|\boldsymbol{\beta} - \boldsymbol{\beta}'\|_2^2$$

We further assume that

$$\rho_{\min}^-(\widetilde{s}) := \min_i \rho_i^-(\widetilde{s}) \ge \mu > 0.$$

This immediately implies that function $F(\boldsymbol{\beta})$ is also restricted strong convex with constant $\rho_{\min}^-(\widetilde{s})$. Note that in order for Assumption 5.3 to hold for the illustrative examples, specifically for $\mu > 0$ holds, we generally need the mini-batch size b to be in order of s.

Assumption 5.2 and Assumption 5.3 indicate that function $F(\boldsymbol{\beta})$ is strongly smooth and functions $f_i(\boldsymbol{\beta})$ are strongly convex when restricted on to a sparse subspace. These restricted strong smoothness and strong convexity conditions ensure that the standard convex optimization results for strongly convex and smooth objective functions [32] can be applied to our problem settings as well.

In addition, we need the following assumption on the Hessian, which is a standard condition to prove the quadratic rate of Newton's method.

Assumption 5.4 (Restricted Lipschitz Hessian). Suppose that $F(\boldsymbol{\beta})$ is twice continuously differentiable. We say F has Restricted Lipschitz Hessian with constant $\gamma_{\widetilde{s}}$ if,

$$\Big\| \big[\nabla^2 F(\boldsymbol{\beta}) - \nabla^2 F(\boldsymbol{\beta}') \big]_{\widetilde{S}} \Big\|_2 \le \gamma_{\widetilde{s}} \|\boldsymbol{\beta} - \boldsymbol{\beta}'\|_2$$

for all index set $\widetilde{S}$ with cardinality $|\widetilde{S}| \le \widetilde{s}$ and all $\boldsymbol{\beta}, \boldsymbol{\beta}'$ with $\mathrm{supp}(\boldsymbol{\beta}) \cup \mathrm{supp}(\boldsymbol{\beta}') \subseteq \widetilde{S}$.

Now we are ready to present our main theorem.

Theorem 5.5. *Suppose Assumptions 5.2 and 5.4 hold with $\widetilde{s} = 2s + s^*$. In addition, assume that $K_1 \ge 16/3 \cdot \log d$, $K_2 \ge \log(16/\mu)/\mu - 1$, then with probability at least $1 - 2/d$, the estimator $\boldsymbol{\beta}^{(t)}$ from Algorithm 1 satisfies*

$$\Big\| \boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^* \Big\|_2 \le \psi_1 \|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\|_2 + \psi_2 \|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\|_2^2 + \frac{\alpha\sqrt{s}}{\mu} \cdot \Big\| \nabla F(\boldsymbol{\beta}^*) \Big\|_\infty, \tag{5.1}$$

where $\alpha = \sqrt{1 + (\rho + \sqrt{(4+\rho)\rho})/2}$, $\rho = \min\{s^*, d - s\}/(s - s^* + \min\{s^*, d - s\})$, $\psi_1 = \alpha L \big(4\sqrt{\log d/(3K_1)}/\mu + 1/16\big)$, $\psi_2 = \kappa_{\widetilde{s}} \gamma_{\widetilde{s}}/(2\mu)$.

Remark 5.6. Theorem 5.5 demonstrates that our proposed algorithm has a composite rate of convergence in optimization. It starts with a quadratic rate of convergence, then transits into a linear rate of convergence to the unknown true parameter up to the statistical error, which is represented by the last term in (5.1). The composite rate of convergence is illustrated in Figure 1 using a synthetic data. Similar composite rates of convergence have been proved in [10, 11], but for convex optimization in the classical regime instead of nonconvex optimization in the high-dimensional regime here. Note that by choosing K_1 to be large enough, we can always make the linear contraction parameter ψ_1 less than 1 since $L \le 1$ and a sufficiently large s will ensure that $\alpha/16 \le 1$. This guarantees the contraction for the linear convergence. And in order to guarantee the contraction for the quadratic term, we require the initial point satisfying $\|\boldsymbol{\beta}^{(0)} - \boldsymbol{\beta}^*\|_2 < 1/\psi_2$. Such requirement can be achieved by initialing the proposed algorithm with gradient hard thresholding algorithm [51].

Theorem 5.5 immediately implies the following result.

Corollary 5.7. *Under the same conditions of Theorem 5.5, if initial point $\boldsymbol{\beta}^{(0)}$ satisfies $\|\boldsymbol{\beta}^{(0)} - \boldsymbol{\beta}^*\|_2 \le \psi_1/\psi_2$, then with probability at least $1 - 2/d$, the estimator $\boldsymbol{\beta}^{(t)}$ from Algorithm 1 satisfies*

$$\Big\| \boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^* \Big\|_2 \le \psi_1 \|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\|_2 + \frac{\alpha\sqrt{s}}{\mu} \cdot \Big\| \nabla F(\boldsymbol{\beta}^*) \Big\|_\infty,$$

where $\alpha = \sqrt{1 + (\rho + \sqrt{(4+\rho)\rho})/2}$, $\rho = \min\{s^*, d - s\}/(s - s^* + \min\{s^*, d - s\})$, $\psi_1 = \alpha L \big(4\sqrt{\log d/(3K_1)}/\mu + 1/16\big)$.

Remark 5.8. Corollary 5.7 suggests that our proposed algorithm achieves a linear rate of convergence, provided that the initial solution $\boldsymbol{\beta}^{(0)}$ is sufficiently close to the unknown true parameter.

6 IMPLICATIONS FOR SPECIFIC STATISTICAL ESTIMATION PROBLEMS

Our main theory for the generic optimization problem in (1.1) can be readily applied to specific examples in Section 3, which are highlighted in this section. Specifically, we will show the benefits brought to sparse linear regression and sparse generalized linear models.

6.1 Sparse Linear Regression Models

To begin with, we assume that the noise vector $\boldsymbol{\epsilon}$ in (3.1) is zero-mean and has sub-Gaussian tails.

Assumption 6.1. $\boldsymbol{\epsilon}$ is a zero mean random vector, and there exists a constant $\sigma > 0$ such that for any fixed $\|\mathbf{v}\|_2 = 1$, we have

$$\mathbb{P}\Big(|\mathbf{v}^\top \boldsymbol{\epsilon}| > \delta\Big) \leq 2\exp\Big(-\delta^2/(2\sigma^2)\Big) \quad \text{for all} \quad \delta > 0.$$

In addition, we make an additional assumption on the design matrix $\mathbf{X}$ in (3.1).

Assumption 6.2. For all columns in $\mathbf{X} \in \mathbb{R}^{N\times d}$, we have $\|\mathbf{X}_{*j}\|_2 \leq \sqrt{N}$, where $\mathbf{X}_{*j}$ is the j-th column of $\mathbf{X}$.

Note that Assumption 6.2 is often made in the analysis of Lasso estimator [31, 56].

Corollary 6.3. *Under the same conditions as Theorem 5.5, if Assumptions 6.1 and 6.2 hold, then with probability at least* $1 - 2/d$, *the estimator* $\boldsymbol{\beta}^{(t)}$ *from Algorithm 1 for sparse linear regression in* (3.2) *satisfies*

$$\begin{aligned}&\Big\|\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^*\Big\|_2\\ &\leq \underbrace{\psi_1\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\|_2 + \psi_2\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\|_2^2}_{\text{Optimization Error}} + \underbrace{\frac{C\alpha\sigma}{\mu}\sqrt{\frac{s^*\log d}{N}}}_{\text{Statistical Error}},\end{aligned} \qquad (6.1)$$

where σ *is the variance proxy of the sub-Gaussian random vector* $\boldsymbol{\epsilon}$, $\alpha = \sqrt{1 + (\rho + \sqrt{(4+\rho)\rho})/2}$, $\rho = \min\{s^*, d-s\}/(s-s^*+\min\{s^*, d-s\})$, $\psi_1 = \alpha L\Big(4\sqrt{\log d/(3K_1)}/\mu + 1/16\Big)$, $\psi_2 = \kappa_{\tilde{s}}\gamma_{\tilde{s}}/(2\mu)$.

On the right hand side of (6.1), there are two types of errors. The first error term is the optimization error, which goes to zero when t goes to infinity. The second error term is the statistical error, which is independent of t but depends on the sample size N. Corollary 6.3 suggests that when applying our algorithm to sparse linear regression, as long as the number of iterations is sufficiently large, it achieves $O(\sqrt{s^*\log d/N})$ statistical error, that matches the minimax optimal rate for sparse linear regression [36].

6.2 Sparse Generalized Linear Models

For generalized linear model, we need the following assumption on its link function $\Phi(t)$, which is introduced in (3.3).

Assumption 6.4. There exists one $\alpha_u > 0$ such that the second derivative of the link function satisfies $\Phi''(t) \leq \alpha_u$ for all $t \in \mathbb{R}$.

Similar assumption has been made in [25]. We now provide a corollary for the problem of sparse generalized linear model estimation, as introduced in Example 3.2.

Corollary 6.5. *Under the same conditions as Theorem 5.5, if Assumptions 6.2 and 6.4 hold, then with probability at least* $1 - 2/d$, *the estimator* $\boldsymbol{\beta}^{(t)}$ *from Algorithm 1 for sparse generalized linear models in* (3.4) *satisfies*

$$\begin{aligned}&\Big\|\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^*\Big\|_2\\ &\leq \underbrace{\psi_1\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\|_2 + \psi_2\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\|_2^2}_{\text{Optimization Error}} + \underbrace{\frac{C\alpha\alpha_u}{\mu}\sqrt{\frac{s^*\log d}{N}}}_{\text{Statistical Error}},\end{aligned} \qquad (6.2)$$

where α_u *is an upper bound on the second derivative of the link function* $\Phi(t)$, *where* $\alpha = \sqrt{1 + (\rho + \sqrt{(4+\rho)\rho})/2}$, $\rho = \min\{s^*, d-s\}/(s-s^*+\min\{s^*, d-s\})$, $\psi_1 = \alpha L\Big(4\sqrt{\log d/(3K_1)}/\mu + 1/16\Big)$, $\psi_2 = \kappa_{\tilde{s}}\gamma_{\tilde{s}}/(2\mu)$.

Corollary 6.5 demonstrates that when applying our algorithm to sparse generalized linear models, it also achieves $O(\sqrt{s^*\log d/N})$ statistical error rate, which is minimax rate-optimal.

7 EXPERIMENTS

In this section, we apply Algorithm 1 to sparse regression problems, and present numerical results on both synthetic and large-scale real datasets to evaluate the performance of the proposed algorithm.

7.1 Benchmark Problems

The first problem is the sparse linear regression shown in Example 3.1. The second problem is the sparse logistic regression for classification, which is a special instance of sparse generalized linear models (Example 3.2). To be more specific, in Example 3.2, let

$$y_i = \begin{cases} 1, & \text{with probability } 1/(1+\exp(\mathbf{x}_i^\top\boldsymbol{\beta}^*)) \\ 0, & \text{with probability } 1 - 1/(1+\exp(\mathbf{x}_i^\top\boldsymbol{\beta}^*)) \end{cases}.$$

Then the sparsity constrained optimization is given by

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{N}\Big[-y_i\cdot\mathbf{x}_i^\top\boldsymbol{\beta} + \log\Big(1+\exp(\mathbf{x}_i^\top\boldsymbol{\beta})\Big)\Big] \text{ subject to } \|\boldsymbol{\beta}\|_0 \leq s,$$

where $y_i \in \{0, 1\}$. The same problem has been studied by [51].

7.2 Datasets

7.2.1 *Sparse Linear Regression.*

Synthetic Data: We generate an $N \times d$ design matrix $\mathbf{X}$ with rows drawn independently from a multivariate normal distribution $N(\mathbf{0}, \Sigma)$, where each element of Σ is defined by $\Sigma_{ij} = 0.6^{|i-j|}$. The true regression coefficient vector $\boldsymbol{\beta}^*$ has s^* nonzero entries that are drawn independently from the standard normal distribution. The response vector is generated by $\mathbf{y} = \mathbf{X}^\top\boldsymbol{\beta}^* + \boldsymbol{\epsilon}$, where each entry of $\boldsymbol{\epsilon}$ follows a normal distribution with zero mean and variance $\sigma^2 = 0.01$. We generate the synthetic data in two settings: (1) $N = 2,500, d = 5,000, s^* = 250$, (2) $N = 5,000, d = 10,000, s^* = 500$.

E2006-TFIDF Data: For lasso, we use E2006-TFIDF dataset to test the performance on sparsity constrained linear regression, which predicts risk from financial reports from thousands of publicly traded U.S. companies [19]. It contains 16,087 training instances, 3,308 testing instances and we randomly sample 50,000 features for this experiment.

7.2.2 *Sparse Logistic Regression.*

Synthetic Data: We generate an $N \times d$ design matrix $\mathbf{X}$ with rows drawn independently from a multivariate normal distribution $N(\mathbf{0}, \mathbf{I})$, where $\mathbf{I}$ is a $d \times d$ identity matrix. The true regression coefficient vector $\boldsymbol{\beta}^*$ is generated in the same way as sparse linear regression. Each response variable is generated from the logistic distribution. We generate the synthetic data in two settings: (1) $N = 5,000, d = 1,000, s^* = 50$, (2) $N = 10,000, d = 2,000, s^* = 100$.

RCV1 Data: RCV1 dataset is a Reuters Corpus Volume I dataset for text categorization research [23]. Reuters Corpus Volume I (RCV1)

is an archive of over 800,000 manually categorized newswire stories made available by Reuters, Ltd. for research purposes. This dataset contains $20,242$ training instances, $677,399$ testing instances and $47,236$ features. We use the whole training set and a subset of the test set, which contains $20,000$ testing instances for our experiment.
Astro-ph Data: The astro-ph dataset [22] collects the abstracts of papers submitted to Astro Physics category in arXiv and their author graph. This dataset classifies whether the papers belong in the astro-physics section in arXiv or not according to their abstracts. This dataset contains $29,882$ training instances, $32,478$ testing instances and $99,757$ features.
20 Newsgroups Data: The 20 Newsgroups data [20] collects $19,996$ newsgroup documents, from across 20 different newsgroups. We divided the dataset randomly into a training set which contains $10,000$ training instances, and a test set with $9,996$ testing instances. A total number of $100,000$ features are selected out of the original $1,355,191$ features via Fisher information.

7.3 Evaluation Measures and Experimental Environment

We test our proposed algorithm and other baseline methods in terms of convergence rate and estimation/prediction performance. For the evaluation of convergence, we use the measure of function value gap, i.e., $F(\boldsymbol{\beta}^{(T+1)}) - F(\boldsymbol{\beta}^*)$. As for estimation/prediction performance, we adopt different measures for different datasets.
Estimation Error: On synthetic datasets, we use the estimation error, i.e., $\|\boldsymbol{\beta}^{(T+1)} - \boldsymbol{\beta}^*\|_2$, as the performance measure. A lower estimation error means a higher testing accuracy.
Mean Squared Error (MSE): For sparse linear regression on real datasets, MSE is adopted. A lower MSE means a higher testing accuracy.
Classification Error: For sparse logistic regression on real datasets, we simply use classification error as the indicator for testing accuracy. A lower classification error means a higher testing accuracy.

The experiments are conducted on a computer with an 4-core 2.7GHz CPU and a 16GB RAM.

7.4 Baseline Methods

We compare our algorithm (FNHTP) with the state-of-the-art sparsity constrained optimization algorithms:
Gradient Hard Thresholding Pursuit (GraHTP) [51]: This work is based on standard gradient descent algorithm and combined with hard thresholding to deal with the sparsity constraint.
Stochastic Variance Reduced Gradient with Hard Thresholding (SVRGHT) [24]: This method incorporates SVRG [18] with iterative hard thresholding to further improve the efficiency of full gradient based algorithms.
Newton Greedy Pursuit (NTGP) [52]: This method tries to solve a sparsity constrained Newton's method by iteratively solving an exact minimization of a quadratic problem.
Subsampling and eigenvalue thresholding Newton (NewSamp) [11]: This method adopted sub-sampling together with eigenvalue thresholding to reduce per-iteration cost of Newton's method. The original algorithm was designed for classical regime but we modify it to fit the high-dimensional regime here.

Since SVRGHT algorithm has two layer of iteration loops. To be clear, our later comparison over iterations refers to the outer loop iteration for SVRGHT algorithm.

All the algorithms are implemented in Matlab, and the implementations are optimized for all algorithms.

7.5 Parameter Settings

For synthetic data, the sparsity parameter s is set to $s = 4s^*$ in sparse linear regression and $s = 1.2s^*$ in sparse logistic regression for all algorithms. For real data, we choose $s = 2,000$ for all algorithms.

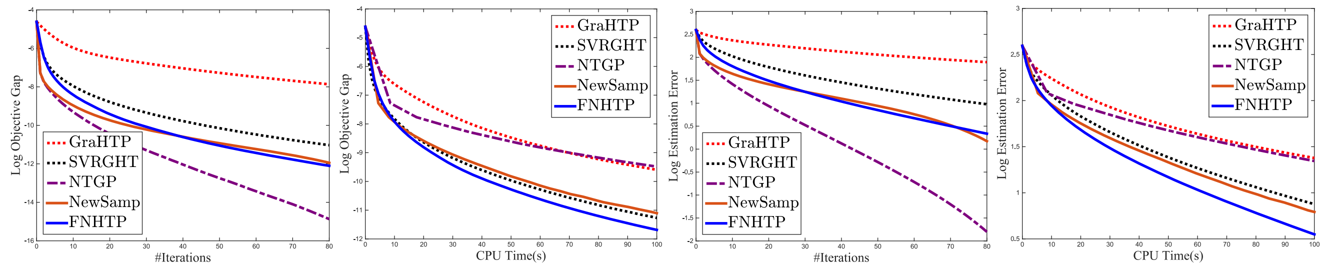
As to our algorithm, for synthetic data, we set $K_1 = 1$, $K_2 = N/2$ for sparse linear regression, and $K_1 = 1$, $K_2 = N/5$ for sparse logistic regression. For real datasets, we chose $K_1 = 1$, $K_2 = 1000$. And for batch size we choose $b = 3s$, where s is the sparsity parameter. On both synthetic and real data, we chose $K_1 = 1$ because we found in practice, by setting $K_1 = 1$, we have already achieved a satisfying performance. Thus, even though choosing $K_1 = 1$ means a little less accurate in approximating the inverse Hessian matrix, it saves more computation time and therefore further improves the efficiency of our algorithm.

For all the baseline algorithms, the step size parameters are chosen around the theoretical values to give the fastest convergence under the five-fold cross validation. In addition, for SVRGHT, we set the number of inner loop to be $m = n$, as suggested by [24]. And for NewSamp, we set the sample size $b = 100$ and $r = 10$.

7.6 Experimental Results

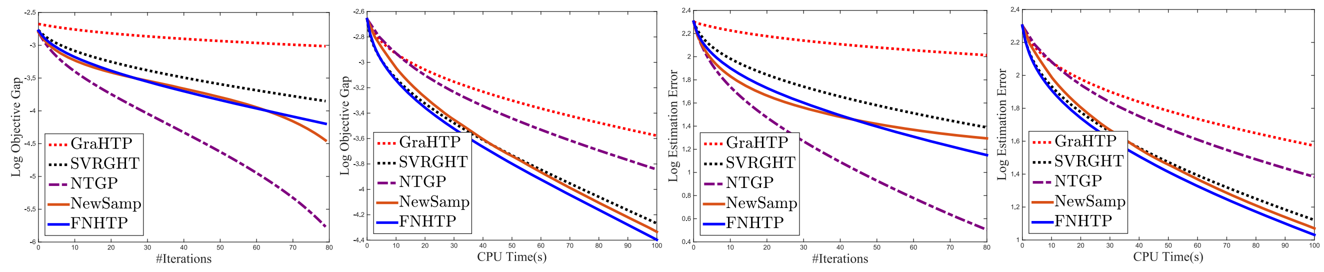
Due to the intrinsic randomness of SVRGHT, NewSamp and FNHTP, all the experimental results for these three algorithms are obtained from 10 replications. Both the mean and standard deviation values are reported in Tables 2, 3, 4 and 5. And Figures 2– 4 plot the mean value of the results from all these replications. Due to space limit, more experimental results can be found in the longer version of this paper.

7.6.1 Synthetic Data Experiments. We first investigate the sparsity constrained linear regression problem in (3.2) on synthetic data. we plot the logarithm of the objective function value gap and the estimation error $\|\boldsymbol{\beta}^{(t)} - \boldsymbol{\beta}^*\|_2$ of each algorithm for comparison. In Figure 2, we compare the above measures for all algorithms after the same number of iterations and the same CPU time. As we can see from the figure, in terms of iteration number, NTGP algorithm is the fastest due to the quadratic convergence rate. Our algorithm FNHTP follows next and takes a clear lead over the other gradient based algorithms. This is consistent with the composite rate shown in our theory. When considering the performance under the same CPU time, NTGP falls behind due to the high computational complexity in each iteration. Our proposed algorithm outperforms all the baseline methods. The estimation error follows the similar pattern and we can observe that the result is consistent with our theory in Corollary 6.3, i.e., the estimation error of our algorithm consists of two terms: the optimization error that goes to zero at a composite rate, and the statistical error that depends on the problem parameters (d, n, s^* and so on). Note that NewSamp is worse than our algorithm. It suggests that the approximation technique for Hessian matrix used by NewSamp is not as good as



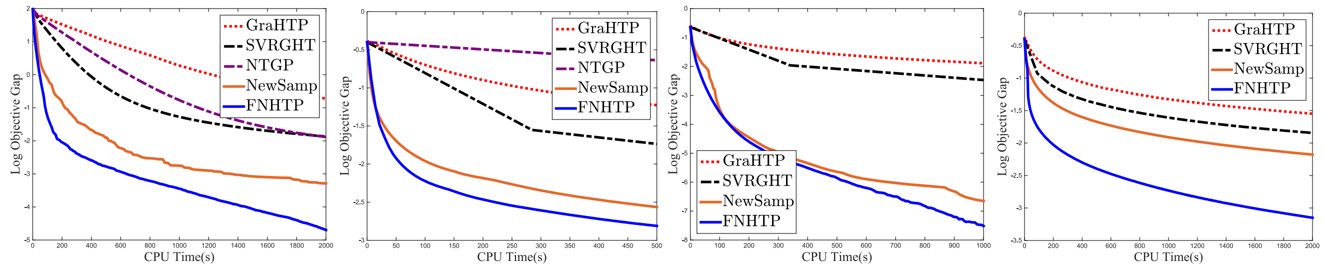
(a) (b) (c) (d)

Figure 2: Comparison of different algorithms for sparsity constrained sparse linear regression on synthetic dataset with $N = 5000, d = 10000, s^* = 500$. (a) and (b) show the logarithm of the function value gap over iterations and CPU time respectively. (c) and (d) demonstrate the logarithm of the estimation error over iterations and CPU time respectively.



(a) (b) (c) (d)

Figure 3: Comparison of different algorithms for sparsity constrained logistic regression on synthetic dataset with $N = 10000, d = 2000, s^* = 100$. (a) and (b) show the logarithm of the function value gap over iterations and CPU time respectively. (c) and (d) demonstrate the logarithm of the estimation error over iterations and CPU time respectively.



(a) E2006 TFIDF (b) RCV1 (c) astro-ph (d) 20 Newsgroups

Figure 4: Comparison of different algorithms in terms of the logarithm of objective function value gap on real regression and classification datasets.

Table 2: Regression on E2006-TFIDF: MSE comparison of algorithms under the same CPU time averaged over 10 replications. The boldfaced numbers denote the lowest MSE among all the algorithms under the same CPU time.

Method	Time = 400s	Time = 800s	Time = 1200s	Time = 1600s	Time = 2000s
GraHTP	3.1068	1.4970	0.8839	0.6237	0.5029
SVRGHT	1.6278±0.0005	0.8044±0.0008	0.6129±0.0010	0.5335±0.0006	0.4852±0.0008
NTGP	3.5250	1.2641	0.7248	0.5536	0.4801
NewSamp	0.5942±0.0081	0.3916±0.0061	0.3491±0.0059	0.3329±0.0042	0.3206±0.0035
FNHTP	**0.3671**± 0.0063	**0.3242**±0.0027	**0.3027**±0.0024	**0.2867**±0.0021	**0.2725**±0.0022

the von Neumann series based Hessian approximation used by our algorithm.

For sparse logistic regression problem, we plot the logarithm of the objective function value gap and the estimation error $\|\boldsymbol{\beta}^{(t)} -$

Table 3: Classification on RCV1: classification error comparison of algorithms for the same CPU time averaged over 10 replications. The boldfaced numbers denote the lowest classification error among all the algorithms for the same CPU time.

Method	Time = 100s	Time = 200s	Time = 300s	Time = 400s	Time = 500s
GraHTP	0.0726	0.0663	0.0636	0.0607	0.0584
SVRGHT	0.0602±0.0009	0.0549±0.0007	0.0521±0.0010	0.0506±0.0006	0.0494±0.0008
NTGP	–	–	–	–	0.0431
NewSamp	0.0493±0.0009	0.0447±0.0005	0.0426±0.0007	0.0410±0.0006	0.0404±0.0008
FNHTP	**0.0436**±0.0006	**0.0416**±0.0005	**0.0402**±0.0004	**0.0395**±0.0006	**0.0389**±0.0003

Table 4: Classification on astro-ph: classification error comparison of algorithms for the same CPU time averaged over 10 replications. The boldfaced numbers denote the lowest classification error among all the algorithms for the same CPU time.

Method	Time = 200s	Time = 400s	Time = 600s	Time = 800s	Time = 1000s
GraHTP	0.1207	0.1013	0.0899	0.0829	0.0793
SVRGHT	0.0822±0.0007	0.0742±0.0009	0.0698±0.0005	0.0675±0.0005	0.0660±0.0006
NTGP	–	–	–	–	0.0845
NewSamp	0.0522±0.0043	0.0497 ±0.0026	0.0488±0.0033	0.0487± 0.0022	0.0484±0.0020
FNHTP	**0.0505**±0.0042	**0.0476**±0.0039	**0.0468**±0.0025	**0.0467**±0.0015	**0.0466**±0.0025

Table 5: Classification on 20 Newsgroups: classification error comparison of algorithms for the same CPU time averaged over 10 replications. The boldfaced numbers denote the lowest classification error among all the algorithms for the same CPU time.

Method	Time = 400s	Time = 800s	Time = 1200s	Time = 1600s	Time = 2000s
GraHTP	0.1291	0.1070	0.1003	0.0941	0.0887
SVRGHT	0.1011±0.0006	0.0903±0.0007	0.0836±0.0004	0.0790±0.0006	0.0773±0.0008
NTGP	–	–	–	–	0.0831
NewSamp	0.0848±0.0003	0.0768±0.0004	0.0732±0.0002	0.0708± 0.0003	0.0681±0.0007
FNHTP	**0.0665**±0.0005	**0.0653**±0.0007	**0.0652**±0.0006	**0.0650**±0.0005	**0.0650**±0.0004

$\boldsymbol{\beta}^*\|_2$ for comparison in Figure 3. Similar to the sparse linear regression case, our proposed algorithm outperforms the baseline algorithms under the same CPU time and demonstrates a composite rate of convergence, which is consistent with Corollary 6.5.

7.6.2 Real Data Experiments. For sparse linear regression problem, we evaluate the performance of all algorithms on E2006-TFIDF dataset. Figure 4 (1) illustrates the logarithm of the objective function value gap for all the algorithms. We can see that our algorithm converges faster than all the baselines in terms of CPU time. In Table 2, we detailedly show the mean value as well as the standard error of MSE for all the algorithms with respect to the total CPU time. Since there is no randomness in GraHTP and NTGP, their standard errors are zero. It can be seen that our algorithm outperforms all the baseline algorithms in terms of the mean square error under the same CPU time. And for the NewSamp method, even though it achieves better performance than gradient based algorithms and NTGP method, it still falls behind compared with our proposed method. This again suggests that von Neumann series based Hessian approximation used in our algorithm is better than subsampling based approximation.

For sparse logistic regression problem, we test the performance of all algorithms on three classification datasets. Tables 3, 4 and 5 demonstrate the classification results for all algorithms including ours on RCV1, astro-ph and 20 newsgroups datasets respectively. The blank in NTGP method means it did not generate any result up to that time. It is obvious that our proposed algorithm achieves the lowest test error on all real datasets during all the time and beats all the state-of-the-art baseline algorithms. Figure 4 (b) (c) and (d) further illustrate the logarithm of the objective function value gaps for both the baseline algorithms and ours. This clearly demonstrates the superiority of our algorithm over the baseline methods. Note that in Figure 4 (c) and (d) we choose not to report the result of NTGP, since NTGP is not as scalable as other methods and can not produce enough outputs in the given time.

8 CONCLUSIONS

We proposed a fast Newton hard thresholding pursuit algorithm for sparsity constrained nonconvex optimization problems. We proved that the algorithm enjoys a composite rate of convergence to the unknown true parameter up to the optimal statistical error. Experiments on both synthetic and real datasets verified the effectiveness and efficiency of our algorithm. In our future work, we will extend the proposed algorithm and theory to estimating other structured model parameters under rank constraint [35].

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their helpful comments. This research was sponsored in part by the National Science Foundation IIS-1618948 and IIS-1652539.

REFERENCES

[1] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, and Tengyu Ma. 2016. Finding approximate local minima for nonconvex optimization in linear time. *arXiv preprint arXiv:1611.01146* (2016).

[2] Naman Agarwal, Brian Bullins, and Elad Hazan. 2016. Second Order Stochastic Optimization in Linear Time. *arXiv preprint arXiv:1602.03943* (2016).

[3] Zeyuan Allen-Zhu and Elad Hazan. 2016. Variance reduction for faster non-convex optimization. *arXiv preprint arXiv:1603.05643* (2016).

[4] Sohail Bahmani, Bhiksha Raj, and Petros T Boufounos. 2013. Greedy sparsity-constrained optimization. *The Journal of Machine Learning Research* 14, 1 (2013), 807–841.

[5] Christopher M Bishop. 1995. *Neural networks for pattern recognition.* Oxford university press.

[6] Thomas Blumensath and Mike E Davies. 2009. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis* 27, 3 (2009), 265–274.

[7] Raghu Bollapragada, Richard Byrd, and Jorge Nocedal. 2016. Exact and Inexact Subsampled Newton Methods for Optimization. *arXiv preprint arXiv:1609.08502* (2016).

[8] Jinghui Chen and Quanquan Gu. 2016. Accelerated stochastic block coordinate gradient descent for sparsity constrained nonconvex optimization. In *Conference on Uncertainty in Artificial Intelligence.*

[9] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. 2014. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems.* 1646–1654.

[10] Murat A Erdogdu. 2015. Newton-Stein Method: A Second Order Method for GLMs via Stein's Lemma. In *Advances in Neural Information Processing Systems.* 1216–1224.

[11] Murat A Erdogdu and Andrea Montanari. 2015. Convergence rates of sub-sampled newton methods. In *Advances in Neural Information Processing Systems.* 3052–3060.

[12] Jianqing Fan and Runze Li. 2001. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association* 96, 456 (2001), 1348–1360.

[13] Simon Foucart. 2011. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM J. Numer. Anal.* 49, 6 (2011), 2543–2563.

[14] Saeed Ghadimi and Guanghui Lan. 2013. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization* 23, 4 (2013), 2341–2368.

[15] Quanquan Gu, Zhaoran Wang, and Han Liu. 2014. Sparse pca with oracle property. In *Advances in neural information processing systems.* 1529–1537.

[16] Prateek Jain, Ambuj Tewari, and Purushottam Kar. 2014. On Iterative Hard Thresholding Methods for High-dimensional M-Estimation. In *NIPS.* 685–693.

[17] Ali Jalali, Christopher C Johnson, and Pradeep K Ravikumar. 2011. On learning discrete graphical models using greedy methods. In *NIPS.* 1935–1943.

[18] Rie Johnson and Tong Zhang. 2013. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS.* 315–323.

[19] Shimon Kogan, Dimitry Levin, Bryan R Routledge, Jacob S Sagi, and Noah A Smith. 2009. Predicting risk from financial reports with regression. In *ACL.* Association for Computational Linguistics, 272–280.

[20] Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning.* 331–339.

[21] Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Y Ng. 2006. Efficient l˜ 1 regularized logistic regression. In *AAAI,* Vol. 6. 401–408.

[22] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2.

[23] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research* 5 (2004), 361–397.

[24] Xingguo Li, Tuo Zhao, Roman Arora, Han Liu, and Jarvis Haupt. 2016. Stochastic variance reduced optimization for nonconvex sparse learning. *International Conference on Machine Learning* (2016).

[25] Po-Ling Loh and Martin J Wainwright. 2013. Regularized m-estimators with nonconvexity: Statistical and algorithmic theory for local optima. In *NIPS.* 476–484.

[26] Stéphane G Mallat and Zhifeng Zhang. 1993. Matching pursuits with time-frequency dictionaries. *Signal Processing, IEEE Transactions on* 41, 12 (1993), 3397–3415.

[27] Philipp Moritz, Robert Nishihara, and Michael I Jordan. 2015. A Linearly-Convergent Stochastic L-BFGS Algorithm. *arXiv preprint arXiv:1508.02087* (2015).

[28] Mojmir Mutny. 2016. Stochastic Second-Order Optimization via von Neumann Series. *arXiv preprint arXiv:1612.04694* (2016).

[29] Deanna Needell and Joel A Tropp. 2009. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis* 26, 3 (2009), 301–321.

[30] Sahand Negahban and Martin J Wainwright. 2011. Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *The Annals of Statistics* (2011), 1069–1097.

[31] Sahand Negahban, Bin Yu, Martin J Wainwright, and Pradeep K Ravikumar. 2009. A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers. In *NIPS.* 1348–1356.

[32] Yurii Nesterov. 2013. *Introductory lectures on convex optimization: A basic course.* Vol. 87. Springer Science & Business Media.

[33] Yurii Nesterov and Boris T Polyak. 2006. Cubic regularization of Newton method and its global performance. *Mathematical Programming* 108, 1 (2006), 177–205.

[34] Nam Nguyen, Deanna Needell, and Tina Woolf. 2014. Linear convergence of stochastic iterative greedy algorithms with sparse constraints. *arXiv preprint arXiv:1407.0088* (2014).

[35] Renkun Ni and Quanquan Gu. 2016. Optimal statistical and computational rates for one bit matrix completion. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics.* 426–434.

[36] Garvesh Raskutti, Martin J Wainwright, and Bin Yu. 2011. Minimax rates of estimation for high-dimensional linear regression over-balls. *Information Theory, IEEE Transactions on* 57, 10 (2011), 6976–6994.

[37] Benjamin Recht. 2011. A simpler approach to matrix completion. *The Journal of Machine Learning Research* 12 (2011), 3413–3430.

[38] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. 2016. Stochastic variance reduction for nonconvex optimization. *arXiv preprint arXiv:1603.06160* (2016).

[39] Farbod Roosta-Khorasani and Michael W Mahoney. 2016. Sub-Sampled Newton Methods I: Globally Convergent Algorithms. *arXiv preprint arXiv:1601.04737* (2016).

[40] Nicolas L Roux, Mark Schmidt, and Francis R Bach. 2012. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems.* 2663–2671.

[41] Shai Shalev-Shwartz, Nathan Srebro, and Tong Zhang. 2010. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization* 20, 6 (2010), 2807–2832.

[42] Shai Shalev-Shwartz and Tong Zhang. 2013. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research* 14, 1 (2013), 567–599.

[43] Jie Shen and Ping Li. 2016. A Tight Bound of Hard Thresholding. *arXiv preprint arXiv:1605.01656* (2016).

[44] Lu Tian, Pan Xu, and Quanquan Gu. 2016. Forward backward greedy algorithms for multi-task learning with faster rates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence.* AUAI Press, 735–744.

[45] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), 267–288.

[46] Joel A Tropp and Anna C Gilbert. 2007. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory, IEEE Transactions on* 53, 12 (2007), 4655–4666.

[47] Lingxiao Wang, Xiang Ren, and Quanquan Gu. 2016. Precision matrix estimation in high dimensional gaussian graphical models with faster rates. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics.* 177–185.

[48] Pan Xu and Quanquan Gu. 2016. Semiparametric Differential Graph Models. In *Advances in Neural Information Processing Systems.* 1064–1072.

[49] Peng Xu, Jiyan Yang, Farbod Roosta-Khorasani, Christopher Ré, and Michael W Mahoney. 2016. Sub-sampled Newton Methods with Non-uniform Sampling. In *Advances in Neural Information Processing Systems.* 3000–3008.

[50] Haishan Ye, Luo Luo, and Zhihua Zhang. 2017. A Unifying Framework for Convergence Analysis of Approximate Newton Methods. *arXiv preprint arXiv:1702.08124* (2017).

[51] Xiao-Tong Yuan, Ping Li, and Tong Zhang. 2013. Gradient hard thresholding pursuit for sparsity-constrained optimization. *arXiv preprint arXiv:1311.5750* (2013).

[52] Xiao-Tong Yuan and Qingshan Liu. 2014. Newton Greedy Pursuit: A Quadratic Approximation Method for Sparsity-Constrained Optimization. In *CVPR.* 4122–4129.

[53] Aston Zhang and Quanquan Gu. 2016. Accelerated Stochastic Block Coordinate Descent with Optimal Sampling. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 2035–2044.

[54] C. Zhang. 2010. Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics* 38, 2 (2010), 894–942.

[55] Tong Zhang. 2011. Adaptive forward-backward greedy algorithm for learning sparse representations. *Information Theory, IEEE Transactions on* 57, 7 (2011), 4689–4708.

[56] Tong Zhang and others. 2009. Some sharp performance bounds for least squares regression with L1 regularization. *The Annals of Statistics* 37, 5A (2009), 2109–2144.

[57] Rongda Zhu and Quanquan Gu. 2015. Towards a lower sample complexity for robust one-bit compressed sensing. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15).* 739–747.