Stochastic Iterative Hard Thresholding for Graph-structured Sparsity Optimization

Baojian Zhou [1] **Feng Chen** [1] **Yiming Ying** [2]

Abstract

Stochastic optimization algorithms update models with cheap per-iteration costs sequentially, which makes them amenable for large-scale data analysis. Such algorithms have been widely studied for structured sparse models where the sparsity information is very specific, e.g., convex sparsity-inducing norms or ℓ^0-norm. However, these norms cannot be directly applied to the problem of complex (non-convex) graph-structured sparsity models, which have important application in disease outbreak and social networks, etc. In this paper, we propose a stochastic gradient-based method for solving graph-structured sparsity constraint problems, not restricted to the least square loss. We prove that our algorithm enjoys a linear convergence up to a constant error, which is competitive with the counterparts in the batch learning setting. We conduct extensive experiments to show the efficiency and effectiveness of the proposed algorithms.

1. Introduction

Structured sparse learning models have received increasing attention. They can be formulated as follows

$$\min_{\boldsymbol{x}\in\mathcal{M}} F(\boldsymbol{x}), \quad F(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}). \tag{1}$$

Here, each $f_i(\boldsymbol{x})$ is convex and differentiable and the structured sparsity is reflected by the constraint set $\mathcal{M} \subseteq \mathbb{R}^p$ on $\boldsymbol{x}$. Typically, one can encode the sparsity by introducing sparsity-inducing penalties such as ℓ^1-norm (Tibshirani, 1996; Chen et al., 2001), ℓ^1/ℓ^q mixed norms (Turlach et al., 2005; Yuan & Lin, 2006) and more structured norms built on either disjoint or overlapping groups of variables (Bach et al., 2012b; Jenatton et al., 2011; Obozinski & Bach, 2016; Morales et al., 2010). Such models of sparsity-inducing penalties are often convex and can be solved by convex optimization algorithms (Bach et al., 2012a).

There is a wide range of significant application, including the search of connected subgraphs in networks, where the constraint set $\mathcal{M}$ cannot be encoded by sparsity-inducing norms. Notable application examples include disease outbreak as a connected cluster (Arias-Castro et al., 2011; Qian et al., 2014; Aksoylar et al., 2017), and social events as connected dense subgraphs (Rozenshtein et al., 2014). To capture these complex graph structures such as trees and connected graphs, recent works by Baraniuk et al. (2010) and Hegde et al. (2014; 2016; 2015b) have proposed to use structured sparsity model $\mathbb{M}$ to define allowed supports $\{S_1, S_2, \ldots, S_k\}$ where $\mathcal{M}$ can be represented by $\mathcal{M}(\mathbb{M}) = \{\boldsymbol{x} : \text{supp}(\boldsymbol{x}) \subseteq S_i, \text{ for some } S_i \in \mathbb{M}\}$. These complex models are non-convex and (batch) gradient descent algorithms involve a projection operator $\mathrm{P}(\cdot, \mathcal{M}(\mathbb{M}))$ which is usually NP-hard. In a series of seminal work, Hegde et al. (2015b; 2016; 2015a) used two approximated projections (head and tail) without sacrificing too much precision. However, their work only focused on the least square loss, and computing full gradient per iteration resulted in expensive per-iteration cost $\mathcal{O}(mp)$ if m is the number of examples and p is the data dimension. This largely limits its practical application to big data setting where typically the data is contaminated by non-Gaussian noise and the data volume is huge (i.e., m or/and p is very large). Therefore, it is desirable to develop efficient optimization algorithms for graph-structured sparsity constrained problems scalable to large-scale data.

Stochastic optimization such as stochastic gradient descent (SGD) has become a standard tool for solving large-scale convex and non-convex learning problems (e.g., Bach & Moulines (2013); Bousquet & Bottou (2008); Jin et al. (2017); Kingma & Ba (2014); Li & Orabona (2018); Rakhlin et al. (2012); Shamir & Zhang (2013); Yang & Lin (2015); Ying & Zhou (2017; 2006)). SGD-type algorithms have a cheap per-iteration computation of the gradient over one or

[1]Department of Computer Science, SUNY at Albany, Albany, NY, USA [2]Department of Mathematics and Statistics, SUNY at Albany, Albany, NY, USA. Correspondence to: Baojian Zhou <bzhou6@albany.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

a few training examples, which makes them amenable for large-scale data analysis. Along this research line, SGD-type algorithms for sparse models have been proposed including stochastic proximal gradient methods (Rosasco et al., 2014; Duchi et al., 2011) and stochastic gradient hard thresholding (Nguyen et al., 2017; Zhou et al., 2018a; Murata & Suzuki, 2018; Shen & Li, 2017; Liu et al., 2017). However, these algorithms cannot address the important case of complex (non-convex) graph-structured sparsity constraint problems.

In this paper we leverage the recent success of SGD algorithms for non-convex problems and propose an efficient stochastic optimization algorithm for graph-structured sparsity constraint problems. In particular, we use the structure sparsity model $\mathcal{M}(\mathbb{M})$ to capture more complex graph-structured information of interest. Our main contributions can be summarized as follows:

- We propose a stochastic gradient-based algorithm for solving graph-structured sparsity constraint problems. To our best knowledge, our work is the first attempt to provide stochastic gradient descent-based algorithm for graph-structured sparsity constraint problems.

- The proposed algorithm enjoys linear convergence property under proper conditions.[1] It is proved applicable to a broad spectrum of loss functions. Specifically, for the least square loss, it enjoys a linear convergence rate which is competitive to the algorithm proposed in Hegde et al. (2016). For the logistic loss, we show that it also satisfies the linear convergence property with high probability. Our proofs could be easily applied to other types of projections.

- We conduct extensive experiments to validate the proposed stochastic algorithm. In particular, we focus on two applications: graph sparse linear regression and graph logistic regression on breast cancer data. Our experiment results show that the proposed stochastic algorithm consistently outperforms the deterministic ones.

Outline. We organize the rest of the paper as follows. In Section 2, we introduce the notations, definitions and key assumptions. Our proposed algorithm is presented in Section 3. Section 4 provides the theoretical analysis of the algorithm. Numerical experiments and discussion are respectively in Section 5 and 6. Due to the limited space, all of our formal proofs and further experiment results are provided in the Supplementary Material. The source code and datasets are accessible at: https://github.com/baojianzhou/graph-sto-iht.

[1] Linear convergence up to a tolerance error.

2. Preliminary

Notation. We use boldface letters, e.g., $\boldsymbol{X}$ to denote matrices. The boldface lower-case letters, e.g., $\boldsymbol{x}, \boldsymbol{y}$ are column vectors. The upper-case letters, e.g., H, T, S stand for subsets of $[p] := \{1, 2, \ldots, p\}$. $\boldsymbol{X}_S \in \mathbb{R}^{|S| \times p}$ is the submatrix by keeping rows only in set S. Given the standard basis $\{\boldsymbol{e}_i : 1 \leq i \leq p\}$ of $\mathbb{R}^p$, we also use H, T, S to represent subspaces. For example, the subspace induced by H is the subspace defined as $\text{span}\{\boldsymbol{e}_i : i \in H\}$. We will clarify the difference only if confusion might occur. The restriction of $\boldsymbol{x}$ on S is the vector $\boldsymbol{x}_S$ such that i-th entry $(\boldsymbol{x}_S)_i = x_i$ if $i \in S$; otherwise 0. We denote the support of $\boldsymbol{x}$ as $\text{supp}(\boldsymbol{x}) := \{i : x_i \neq 0\}$. We use $\|\boldsymbol{x}\|$ to denote ℓ^2 norm of $\boldsymbol{x}$. The set complement is defined as $S^c = [p] \backslash S$.

Definition 1 (Subspace model (Hegde et al., 2016))**.** *Given the space $\mathbb{R}^p$, a subspace model $\mathbb{M}$ is defined as a family of linear subspaces of $\mathbb{R}^p$: $\mathbb{M} = \{S_1, S_2, \ldots, S_k, \ldots,\}$, where each S_k is a subspace of $\mathbb{R}^p$. The set of corresponding vectors in these subspaces is denoted as*

$$\mathcal{M}(\mathbb{M}) = \{\boldsymbol{x} : \boldsymbol{x} \in V, \text{ for some } V \in \mathbb{M}\}. \tag{2}$$

Equation (2) is general enough to include many sparse models. For example, the s-sparse set $\mathcal{M}(\mathbb{M}_s) = \{\boldsymbol{x} : |\text{supp}(\boldsymbol{x})| \leq s\}$ (Nguyen et al., 2017; Yuan et al., 2018; Jain et al., 2014; Bahmani et al., 2013), where $\mathbb{M}_s$ contains subspaces spanned by s different standard basis vectors, i.e., $\mathbb{M}_s = \{\text{span}\{\boldsymbol{e}_{i_1}, \boldsymbol{e}_{i_2}, \ldots, \boldsymbol{e}_{i_s}\} : \{i_1, i_2, \ldots, i_s\} \subseteq [p]\}$. Although the proposed algorithm and theorems apply to any $\mathbb{M}$ which admits efficient projections, in this paper, we mainly consider graph-structured sparsity constraint models such as the weighted graph model in Hegde et al. (2015b).

Definition 2 (Weighted Graph Model (Hegde et al., 2015b))**.** *Given an underlying graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$ defined on the coefficients of the unknown vector $\boldsymbol{x}$, where $\mathbb{V} = [p]$ and $\mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$, then the weighted graph model $(\mathbb{G}, s, g, C)$-WGM can be defined as the following set of supports*

$$\begin{aligned}\mathbb{M} = \{S : |S| \leq s, \text{ there is an } F \subseteq \mathbb{V} \text{ with} \\ \mathbb{V}_F = S, \gamma(F) = g, \text{ and } w(F) \leq C\},\end{aligned}$$

where C is the budget on weight of edges w, g is the number of connected components of F, and s is the sparsity.

$(\mathbb{G}, s, g, C)$-WGM captures a broad range of graph structures such as groups, clusters, trees, and connected subgraphs. This flexibility requires an efficient projection operator onto $\mathcal{M}(\mathbb{M})$, i.e., $\text{P}(\cdot, \mathcal{M}(\mathbb{M})) : \mathbb{R}^p \to \mathbb{R}^p$ defined as

$$\text{P}(\boldsymbol{x}, \mathcal{M}(\mathbb{M})) = \underset{\boldsymbol{y} \in \mathcal{M}(\mathbb{M})}{\arg\min} \|\boldsymbol{x} - \boldsymbol{y}\|^2. \tag{3}$$

The above operator is crucial for projected gradient descent-based methods, but to solve the operator is NP-hard in gen-

eral. To explore (3), one needs to solve the following equivalent minimization problem

$$\min_{\boldsymbol{y}\in\mathcal{M}(\mathbb{M})} \|\boldsymbol{x}-\boldsymbol{y}\|^2 \Leftrightarrow \min_{S\in\mathbb{M}} \|\boldsymbol{x}-\mathrm{P}(\boldsymbol{x},S)\|^2.$$

Since $\mathrm{P}(\boldsymbol{x},S)$ is an orthogonal projection operator that projects $\boldsymbol{x}$ onto subspace S, by projection theorem, it always has the following property

$$\|\boldsymbol{x}\|^2 - \|\mathrm{P}(\boldsymbol{x},S)\|^2 = \|\boldsymbol{x}-\mathrm{P}(\boldsymbol{x},S)\|^2.$$

By adding minimization to both sides with respect to subspace S, we obtain

$$\begin{aligned}\min_{S\in\mathbb{M}} \left\{\|\boldsymbol{x}\|^2 - \|\mathrm{P}(\boldsymbol{x},S)\|^2\right\} &= \min_{S\in\mathbb{M}} \|\boldsymbol{x}-\mathrm{P}(\boldsymbol{x},S)\|^2 \\ \|\boldsymbol{x}\|^2 + \max_{S\in\mathbb{M}} \|\mathrm{P}(\boldsymbol{x},S)\|^2 &= \min_{S\in\mathbb{M}} \|\boldsymbol{x}-\mathrm{P}(\boldsymbol{x},S)\|^2. \quad (4)\end{aligned}$$

The above observations lead to a key insight that the NP-hard problem (3) can be solved either by maximizing $\|\mathrm{P}(\boldsymbol{x},S)\|^2$ or by minimizing $\|\mathbf{x}-\mathrm{P}(\boldsymbol{x},S)\|^2$ over S. Instead of minimizing or maximizing these two squared norms over S exactly, Hegde et al. (2015a;b) turned to approximated algorithms, and gave two important projections—head projection and tail projection.

Assumption 1 (Head Projection (Hegde et al., 2016))**.** *Let $\mathbb{M}$ and $\mathbb{M}_{\mathcal{H}}$ be the predefined subspace models. Given any vector $\boldsymbol{x}$, there exists a $(c_{\mathcal{H}},\mathbb{M},\mathbb{M}_{\mathcal{H}})$-Head-Projection which is to find a subspace $H\in\mathbb{M}_{\mathcal{H}}$ such that*

$$\|\mathrm{P}(\boldsymbol{x},H)\|^2 \geq c_{\mathcal{H}}\cdot\max_{S\in\mathbb{M}}\|\mathrm{P}(\boldsymbol{x},S)\|^2, \quad (5)$$

where $0<c_{\mathcal{H}}\leq 1$. We denote $\mathrm{P}(\boldsymbol{x},H)$ as $\mathrm{P}(\boldsymbol{x},\mathbb{M},\mathbb{M}_{\mathcal{H}})$.

Assumption 2 (Tail Projection (Hegde et al., 2016))**.** *Let $\mathbb{M}$ and $\mathbb{M}_{\mathcal{T}}$ be the predefined subspace models. Given any vector $\boldsymbol{x}$, there exists a $(c_{\mathcal{T}},\mathbb{M},\mathbb{M}_{\mathcal{T}})$-Tail-Projection which is to find a subspace $T\in\mathbb{M}_{\mathcal{T}}$ such that*

$$\|\mathrm{P}(\boldsymbol{x},T)-\boldsymbol{x}\|^2 \leq c_{\mathcal{T}}\cdot\min_{S\in\mathbb{M}}\|\boldsymbol{x}-\mathrm{P}(\boldsymbol{x},S)\|^2, \quad (6)$$

where $c_{\mathcal{T}}\geq 1$. We denote $\mathrm{P}(\boldsymbol{x},T)$ as $\mathrm{P}(\boldsymbol{x},\mathbb{M},\mathbb{M}_{\mathcal{T}})$.

Intuitively, head projection keeps large magnitudes while tail projection discards small magnitudes. Take $\mathbb{M}_s$ (a complete graph) as an example. Head projection is to find subspace $H = \mathrm{span}\{\boldsymbol{e}_{i_1},\boldsymbol{e}_{i_2},\ldots,\boldsymbol{e}_{i_s}\}$ where $\{i_1,i_2,\ldots,i_s\}\subseteq[p]$ is the set of indices that keeps the largest s components of $\boldsymbol{x}$, i.e, $|x_{i_1}|\geq|x_{i_2}|\geq\cdots\geq|x_{i_s}|$. As (3) could have multiple solutions, we could have $H\neq T$, but we must have $\|\boldsymbol{x}-\boldsymbol{x}_H\| = \|\boldsymbol{x}-\boldsymbol{x}_T\|$.

Definition 3 ($(\alpha,\beta,\mathcal{M}(\mathbb{M}))$-RSC/RSS)**.** *We say a differentiable function $f(\cdot)$ satisfies the $(\alpha,\beta,\mathcal{M}(\mathbb{M}))$-Restricted Strong Convexity (RSC)/Smoothness (RSS) property if there exist positive constants α and β such that*

$$\frac{\alpha}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2 \leq B_f(\boldsymbol{x},\boldsymbol{y}) \leq \frac{\beta}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2, \quad (7)$$

for all $\boldsymbol{x},\boldsymbol{y}\in\mathcal{M}(\mathbb{M})$, where $B_f(\boldsymbol{x},\boldsymbol{y})$ is the Bregman divergence of f, i.e., $B_f(\boldsymbol{x},\boldsymbol{y}) = f(\boldsymbol{x})-f(\boldsymbol{y})-\langle\nabla f(\boldsymbol{y}),\boldsymbol{x}-\boldsymbol{y}\rangle$. α and β are strong convexity parameter and strong smoothness parameter respectively.

RSC/RSS property was firstly introduced in Agarwal et al. (2012)[2]. Since it captures sparsity of many functions, it has been widely used (e.g., Jain et al. (2014); Yuan et al. (2018); Elenberg et al. (2018); Johnson & Zhang (2013); Shen & Li (2017); Zhou et al. (2018a); Nguyen et al. (2017); Zhou et al. (2018b)). For example, if f is the least square loss, i.e., $f(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2$, then RSC/RSS property can be reduced to the canonical subspace Restricted Isometry Property (RIP) (Candes & Tao, 2005). Next, we characterize the property of F and each f_i of (1) in Assumption 3.

Assumption 3. *Given the objective function $F(\boldsymbol{x})$ in (1), we assume that $F(\boldsymbol{x})$ satisfies α-RSC in subspace model $\mathcal{M}(\mathbb{M}\oplus\mathbb{M}_{\mathcal{H}}\oplus\mathbb{M}_{\mathcal{T}})$. Each function $f_i(\boldsymbol{x})$ satisfies β-RSS in $\mathcal{M}(\mathbb{M}\oplus\mathbb{M}_{\mathcal{H}}\oplus\mathbb{M}_{\mathcal{T}})$, where $\oplus$ of two models $\mathbb{M}_1$ and $\mathbb{M}_2$ is defined as $\mathbb{M}_1\oplus\mathbb{M}_2 := \{S_1\cup S_2 : S_1\in\mathbb{M}_1, S_2\in\mathbb{M}_2\}$.*

3. Algorithm: GRAPHSTOIHT

Algorithm 1 GRAPHSTOIHT

1: **Input**: $\eta_t, F(\cdot), \mathbb{M}, \mathbb{M}_{\mathcal{H}}, \mathbb{M}_{\mathcal{T}}$
2: **Initialize**: $\boldsymbol{x}^0$ such that $\mathrm{supp}(\boldsymbol{x})\in\mathbb{M}$ and $t=0$
3: **for** $t=0,1,2,\ldots$ **do**
4: Choose ξ_t from $[n]$ with probability $Pr(\xi_t)$
5: $\boldsymbol{b}^t = \mathrm{P}(\nabla f_{\xi_t}(\boldsymbol{x}^t),\mathbb{M}\oplus\mathbb{M}_{\mathcal{T}},\mathbb{M}_{\mathcal{H}})$
6: $\boldsymbol{x}^{t+1} = \mathrm{P}(\boldsymbol{x}^t-\eta_t\boldsymbol{b}^t,\mathbb{M},\mathbb{M}_{\mathcal{T}})$
7: **end for**
8: **Return** $\boldsymbol{x}^{t+1}$

The proposed algorithm is named GRAPHSTOIHT presented in Algorithm 1, a stochastic-based method for graph-structured sparsity problems. Initially, $\boldsymbol{x}^0=\mathbf{0}$ and $t=0$. At each iteration, it works as the following three steps:

- **Step 1:** in Line 4, it randomly selects ξ_t from $[n]$ with probability mass function $Pr(\xi_t)$, that is, the current loss $f_{\xi_t}(\cdot)$ has been chosen with probability $Pr(\xi_t)$;
- **Step 2:** in Line 5, the head projection inputs the gradient of $f_{\xi_t}(\boldsymbol{x})$ and outputs the projected vector $\boldsymbol{b}^t$ so that $\mathrm{supp}(\boldsymbol{b}^t)\in\mathcal{M}(\mathbb{M}_{\mathcal{H}})$;
- **Step 3:** in Line 6, the next estimated $\boldsymbol{x}^{t+1}$ is then updated by using the tail projection.

The algorithm repeats the above three steps until stop condition is satisfied. The difference between GRAPHSTOIHT and STOIHT (Nguyen et al., 2017), the latter of which is

[2]This property is related with the theory of paraconvexity (Van Ngai & Penot, 2008) as pointed out by Agarwal et al. (2012).

essentially a stochastic projected gradient descent, is twofold:

1) instead of directly using the gradient $\nabla f_{\xi_t}(\cdot)$ at each iteration as StoIHT did, GraphStoIHT uses a thresholded gradient. Thresholding $\nabla f_{\xi_t}(\cdot)$ at the first stage could be helpful under sparse setting, because most nonzero entries of $\nabla f_{\xi_t}(\cdot)$ are irrelevant.

2) instead of projecting on $\mathcal{M}(\mathbb{M}_s)$ as StoIHT did, the tail projection projects each estimation onto graph-structured subspaces $\mathcal{M}(\mathbb{M}_{\mathcal{T}})$.

We call a batch version ($n = 1$) of this algorithm as GraphIHT, where we simply apply the head and tail projection to AS-IHT in Hegde et al. (2016) by considering a general loss function.

Time complexity Analysis. Due to the NP-hardness of the projection, there is a trade-off between the time cost and the quality of projection. The time complexity of two projections depends on the graph size p and the number of edges $|\mathbb{E}|$. As proved in Hegde et al. (2015b), the running time of both head projection and tail projection is bounded by $\mathcal{O}(|\mathbb{E}|\log^3(p))$. If the graph is sparse as common in real-world applications, i.e., $|\mathbb{E}| = \mathcal{O}(p)$, two projections have nearly-linear time complexity: $\mathcal{O}(p\log^3(p))$ with respect to the feature dimension p. Hence, per-iteration cost $\mathcal{O}(p\log^3(p))$, is still cheaper than that of the deterministic GD algorithms GraphIHT in which the computation of the full gradient will be of cost $\mathcal{O}(np)$.

4. Convergence Analysis of GraphStoIHT

In this section, firstly we give the convergence analysis of GraphStoIHT by characterizing the ***estimation error*** between $\boldsymbol{x}^{t+1}$ and $\boldsymbol{x}^*$, i.e, $\|\boldsymbol{x}^{t+1} - \boldsymbol{x}^*\|$, where $\boldsymbol{x}^*$ is an optimal solution of (1). Then we analyze two commonly used objective functions and prove our algorithm achieves a linear convergence up to a constant error. Our analysis is applicable not only to graph-structured sparsity model but also to any other head and tail projection.

We denote $\xi_{[t]} = (\xi_0, \xi_1, \ldots, \xi_t)$ as the history of stochastic process $\xi_0, \xi_1, \ldots$ up to time t, and all random variables ξ_t are independent of each other. Define the *inverse condition number* $\mu := \alpha/\beta$. To simplify the analysis, we define the probability mass function $Pr(\xi_t = i) = 1/n, 1 \le i \le n$. Before presenting our main Theorem 1, let's take a look at the following key lemma.

Lemma 1. *If each $f_{\xi_t}(\cdot)$ and $F(\boldsymbol{x})$ satisfy Assumption 3 and, given head projection model $(c_{\mathcal{H}}, \mathbb{M} \oplus \mathbb{M}_{\mathcal{T}}, \mathbb{M}_{\mathcal{H}})$ and tail projection model $(c_{\mathcal{T}}, \mathbb{M}, \mathbb{M}_{\mathcal{T}})$, then we have the following inequality*

$$\mathbb{E}_{\xi_t}\|(\boldsymbol{x}^t - \boldsymbol{x}^*)_{H^c}\| \le \sqrt{1-\alpha_0^2}\,\mathbb{E}_{\xi_t}\|\boldsymbol{x}^t - \boldsymbol{x}^*\| + \sigma_1, \quad (8)$$

where

$$\sigma_1 = \Big(\frac{\beta_0}{\alpha_0} + \frac{\alpha_0\beta_0}{\sqrt{1-\alpha_0^2}}\Big)\mathbb{E}_{\xi_t}\|\nabla_I f_{\xi_t}(\boldsymbol{x}^*)\|,$$
$$H = supp(\mathrm{P}(\nabla f_{\xi_t}(\boldsymbol{x}^t), \mathbb{M} \oplus \mathbb{M}_{\mathcal{T}}, \mathbb{M}_{\mathcal{H}})),$$
$$\alpha_0 = c_{\mathcal{H}}\alpha\tau - \sqrt{\alpha\beta\tau^2 - 2\alpha\tau + 1}, \quad \beta_0 = (1 + c_{\mathcal{H}})\tau,$$
$$I = \underset{S \in \mathbb{M} \oplus \mathbb{M}_{\mathcal{T}} \oplus \mathbb{M}_{\mathcal{H}}}{\arg\max}\ \mathbb{E}_{\xi_t}\|\nabla_S f_{\xi_t}(\boldsymbol{x}^*)\|, \text{ and } \tau \in (0, 2/\beta).$$

Inequality (8) means that the head projection always excludes a small fraction of the residual vector $\boldsymbol{x}^t - \boldsymbol{x}^*$ at each iteration. That is to say, most of the large magnitudes in $\boldsymbol{x}^t - \boldsymbol{x}^*$ are captured by the head projection. More specifically, the head projection thresholds small magnitude entries of $\nabla f_{\xi_t}(\boldsymbol{x}^t)$ to zeros. These small magnitude entries could lead the algorithm to a wrong direction. When $F(\boldsymbol{x})$ is the least square loss, Lemma 1 here is similar to Lemma 13 in Hegde et al. (2016). However, there are two important differences: 1) our Lemma 1 can be applied to any functions that satisfy Assumption 3 above, while the RIP condition used in Hegde et al. (2016) can be only applied to the least square loss; 2) since each $f_{\xi_t}(\boldsymbol{x})$ is not strongly convex, the proof in Hegde et al. (2016) cannot be directly used. Instead, we use *co-coercivity* in Nguyen et al. (2017) and then obtain the main theorem below.

Theorem 1. *Let $\boldsymbol{x}^0$ be the start point of Algorithm 1. If we choose a constant learning rate with $\eta_t = \eta$ and use Lemma 1, then the solution $\boldsymbol{x}^{t+1}$ of Algorithm 1 satisfies*

$$\mathbb{E}_{\xi_{[t]}}\|\boldsymbol{x}^{t+1} - \boldsymbol{x}^*\| \le \kappa^{t+1}\|\boldsymbol{x}^0 - \boldsymbol{x}^*\| + \frac{\sigma_2}{1-\kappa}, \quad (9)$$

where

$$\kappa = (1 + c_{\mathcal{T}})\Big(\sqrt{\alpha\beta\eta^2 - 2\alpha\eta + 1} + \sqrt{1-\alpha_0^2}\Big),$$
$$\sigma_2 = \sigma_1 + \eta\mathbb{E}_{\xi_t}\|\nabla_I f_{\xi_t}(\boldsymbol{x}^*)\|, \text{ and } \eta, \tau \in (0, 2/\beta).$$

Theorem 1 shows that Algorithm 1 still possibly enjoys linear convergence even if the full gradient is not available. In order to make more detailed analysis of Theorem 1, we call κ in (9) the ***contraction factor***. A natural question to ask is: under what condition can we get an effective contraction factor, κ, i.e., $\kappa < 1$. When $\kappa < 1$, the estimation error is dominated by the term $\frac{\sigma_2}{1-\kappa}$. Suppose the random distribution of ξ_t is uniform and the approximation factor of the head projection $c_{\mathcal{H}}$ can be arbitrarily boosted close to 1. Taking $\eta = 1/\beta$, in order to get an effective contraction factor, $\kappa < 1$, the inverse of the condition number μ and tail projection factor $c_{\mathcal{T}}$ need to satisfy

$$\kappa = (1 + c_{\mathcal{T}})(1 + 2\sqrt{\mu})\sqrt{1-\mu} < 1. \quad (10)$$

To be more specific, if κ is a function of η, then it takes minimum at $\eta = 1/\beta$. By letting $\eta = 1/\beta$, then κ is simplified as $(1 + c_{\mathcal{T}})(\sqrt{1-\mu} + \sqrt{1-\alpha_0^2})$. Furthermore,

α_0 is a concave function with respect to τ (recall that τ is a free parameter in $(0, 2/\beta)$), and then we have its maximum at $\tau = \frac{1}{\beta}(1 + c_{\mathcal{H}}\sqrt{\frac{\beta-\alpha}{\beta-c_{\mathcal{H}}^2\alpha}})$ after calculation. By taking $c_{\mathcal{H}} \to 1^-$, we present κ as $(1 + c_{\mathcal{T}})(1 + 2\sqrt{\mu})\sqrt{1-\mu}$

One of the assumptions of (10) is that $c_{\mathcal{H}}$ can be arbitrarily close to 1. One can boost the head projection by using the method proposed in Hegde et al. (2016). In our experiment, we find that it is not necessary to boost the head projection because executing the head projection once is sufficient enough to obtain good performance. In the remainder of the section, we consider two popular loss functions to discuss the conditions of getting effective contraction factors.

Graph sparse linear regression. Given a design matrix $\boldsymbol{A} \in \mathbb{R}^{m\times p}$ and corresponding observed noisy vector $\boldsymbol{y} \in \mathbb{R}^m$ that are linked via the linear relationship

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^* + \boldsymbol{\epsilon}, \tag{11}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$. The graph sparse linear regression is to estimate the underlying sparse vector $\boldsymbol{x}^*$. The underlying graph $\mathbb{G}(\mathbb{V}, \mathbb{E})$ is defined on $\boldsymbol{x}^*$. For example, supp$(\boldsymbol{x}^*)$ induces a connected subgraph with s nodes, showcased in Figure 1 in the experiment section. To estimate $\boldsymbol{x}^*$, naturally we consider the least square loss and formulate it as

$$\underset{\text{supp}(\boldsymbol{x})\in\mathcal{M}(\mathbb{M})}{\arg\min} \quad F(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^{n}\frac{n}{2m}\|\boldsymbol{A}_{B_i}\boldsymbol{x} - \boldsymbol{y}_{B_i}\|^2, \tag{12}$$

where m observations have been partitioned into n blocks, $B_1, B_2, \ldots, B_n$. Each block B_i is indexed by i with block size $b = m/n$. In Corollary 1, we show that it is possible to get a linear convergence rate under RSC/RSS assumption.

Corollary 1. *If $F(\boldsymbol{x})$ in (12) satisfies the α-RSC property and each $f_i(\boldsymbol{x}) = \frac{n}{2m}\|\boldsymbol{A}_{B_i}\boldsymbol{x} - \boldsymbol{y}_{B_i}\|^2$ satisfies the β-RSS property, then we have the following condition*

$$\frac{\alpha}{2}\|\boldsymbol{x}\|^2 \le \frac{1}{2m}\|\boldsymbol{A}\boldsymbol{x}\|^2 \le \frac{\beta}{2}\|\boldsymbol{x}\|^2.$$

Let the strong convexity parameter $\alpha = 1 - \delta$ and strong smoothness parameter $\beta = 1 + \delta$, where $0 < \delta < 1$. The condition of effective contraction factor is as the following

$$(1 + c_{\mathcal{T}})\Big(\sqrt{\frac{2}{1+\delta}} + \frac{2\sqrt{2(1-\delta)}}{1+\delta}\Big)\sqrt{\delta} < 1.$$

Table 1 gives a contrast of contraction factor κ in our method and in Hegde et al. (2016). Both conditions are obtained by letting $c_{\mathcal{H}} \to 1^-$. Table 1 shows that κ of the two algorithms could be effective if $\delta \to 0$. By using a sufficiently large number of observations m, we always have $\delta \to 0$. According to the preceding analysis, GRAPHIHT uses a constant learning rate $\eta = 1$ while GRAPHSTOIHT uses the learning rate $\eta = 1/(1 + \delta)$. However, the difference between the two learning rates disappears when $\delta \to 0$. To be more specific, κ of GRAPHIHT is controlled by $\mathcal{O}(\sqrt{\delta}\cdot 2(1 + c_{\mathcal{T}}))$ while for GRAPHSTOIHT, κ is controlled by $\mathcal{O}(\sqrt{\delta}\cdot 3\sqrt{2}(1 + c_{\mathcal{T}}))$. Consequently, the contraction factor κ of GRAPHSTOIHT is competitive with that of GRAPHIHT while GRAPHSTOIHT has the advantage of cheaper per-iteration cost $\mathcal{O}(mp/n)$, if the size of blocks n is large, compared to $\mathcal{O}(mp)$ of GRAPHIHT. To obtain $\kappa < 1$, $\delta \le 0.0527$ for GRAPHIHT while $\delta \le 0.0142$ for GRAPHSTOIHT. The gap between the two κ is mainly due to the randomness introduced in our algorithm.

Table 1. κ of two algorithms

Algorithm	κ
GRAPHIHT	$(1 + c_{\mathcal{T}})\Big(\sqrt{\delta} + 2\sqrt{1-\delta}\Big)\sqrt{\delta}$
GRAPHSTOIHT	$(1 + c_{\mathcal{T}})\Big(\sqrt{\frac{2}{1+\delta}} + \frac{2\sqrt{2(1-\delta)}}{1+\delta}\Big)\sqrt{\delta}$

The error term σ_2 in Equation (9) mainly depends on b and σ. The gradient of f_{ξ_t} at $\boldsymbol{x}^*$ is $\boldsymbol{A}_{B_{\xi_t}}^\top\boldsymbol{\epsilon}_{B_{\xi_t}}/b$. As shown in Nguyen et al. (2017), it can be bounded as $C\sqrt{\sigma^2|I|\log p/b}$, where C is a constant independent of b and σ.

Though GRAPHSTOIHT needs more observations than GRAPHIHT theoretically, our experiment results demonstrate that GRAPHSTOIHT uses less observations than GRAPHIHT to estimate $\boldsymbol{x}^*$ accurately under the same condition. Such kind of interesting experiment phenomenon is reported in Nguyen et al. (2017) but for sparsity constraint.

Graph logistic regression. Given a dataset $\{\boldsymbol{a}_i, y_i\}_{i=1}^m$ ($\{\boldsymbol{a}_i\}_{i=1}^m$ is a set of i.i.d random vectors), the graph logistic regression is formulated as the following problem

$$\underset{\text{supp}(\boldsymbol{x}^*)\in\mathcal{M}(\mathbb{M})}{\arg\min} \quad F(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}), \tag{13}$$

where each $f_i(\boldsymbol{x})$ is defined as

$$f_i(\boldsymbol{x}) = \frac{n}{m}\sum_{j=1}^{m/n}\Big[\log(1 + \exp(-y_{i_j}\boldsymbol{a}_{i_j}^\top\boldsymbol{x}))\Big] + \frac{\lambda}{2}\|\boldsymbol{x}\|^2. \tag{14}$$

Here λ is the regularization parameter. Again we divide $[m]$ into n blocks, i.e., $B_1, B_2, \ldots, B_n$ and assume each block size b is the same, i.e., $b = m/n$. Problem (13) has many important applications. For example, in its application to breast cancer classification based on gene mirco-array dataset and gene-gene network, each $\boldsymbol{a}_i$ is the i-th sample representing the gene expression records of patient i, and y_i is associated label with value -1 or 1. $y_i = 1$ means metastatic growth on patient i while $y_i = -1$ non-metastatic growth on patient i. Assume that we have more prior information such as

gene-gene network, the goal here is to minimize the objective function $F(\boldsymbol{x})$ meanwhile to find a connected subgraph induced by $\text{supp}(\boldsymbol{x})$ which is highly relevant with breast cancer-related genes. This kind of application is theoretically based on the following corollary.

Corollary 2. *Suppose we have the logistic loss, $F(\boldsymbol{x})$ in (13) and each sample $\boldsymbol{a}_i$ is normalized, i.e., $\|\boldsymbol{a}_i\| = 1$. Then $F(\boldsymbol{x})$ satisfies λ-RSC and each $f_i(\boldsymbol{x})$ in (14) satisfies $(\alpha + (1+\nu)\theta_{max})$-RSS. The condition of getting effective contraction factor of* GRAPHSTOIHT *is as the following*

$$\frac{\lambda}{\lambda + n(1+\nu)\theta_{max}/4m} \geq \frac{243}{250}, \tag{15}$$

with probability $1 - p\exp(-\theta_{max}\nu/4)$, where $\theta_{max} = \lambda_{max}(\sum_{j=1}^{m/n} \mathbb{E}[\boldsymbol{a}_{i_j}\boldsymbol{a}_{i_j}^\mathsf{T}])$ and $\nu \geq 1$. $\mathbb{E}[\boldsymbol{a}_{i_j}\boldsymbol{a}_{i_j}^\mathsf{T}]$ is the expectation with respect to the random variable $\boldsymbol{a}_{i_j}$.

Given a sufficiently large number of observations m and a suitable n, it is still possible to find a bound such that the inequality (15) is valid. In Bahmani et al. (2013), they show when $\mu > 1/(1 + \frac{(1+\nu)\theta_{\max}}{4\lambda})$, it is sufficient for GRASP (Bahmani et al., 2013) to get a linear convergence rate. Our corollary shares the similar spirit.

5. Experiments

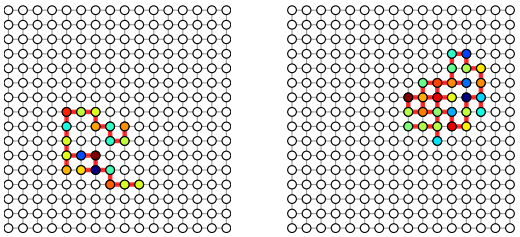
We conduct experiments on both synthetic and real datasets to evaluate the performance of GRAPHSTOIHT[3]. We consider two applications, graph sparse linear regression and breast cancer metastatic classification. All codes are written in Python and C language. All experiments are tested on 56 CPUs of Intel Xeon(R) E5-2680 with 251GB of RAM.

5.1. Graph sparse linear regression

Experimental setup. The graph sparse linear regression is to recover a graph-structured vector $\boldsymbol{x}^* \in \mathbb{R}^p$ by using a Gaussian matrix $\boldsymbol{A} \in \mathbb{R}^{m\times p}$, and the observation vector $\boldsymbol{y}$ where $\boldsymbol{y}$ is measured by $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}^* + \boldsymbol{\epsilon}$ as defined in (11). The entries of the design matrix $\boldsymbol{A}$ are sampled from $\mathcal{N}(0, 1/\sqrt{m})$ independently and nonzero entries of $\boldsymbol{x}^*$ from $\mathcal{N}(0,1)$ independently for the simulation study. $\boldsymbol{\epsilon}$ is potentially a Gaussian noise vector and $\boldsymbol{\epsilon} = \boldsymbol{0}$ the noiseless case. We mainly follow the experimental settings in Nguyen et al. (2017). All results are averaged on 50 trials of trimmed results by excluding the best 5% and the worst 5%. All methods terminate when $\|\boldsymbol{A}\boldsymbol{x}^{t+1} - \boldsymbol{y}\| \leq 10^{-7}$ (corresponding to convergence) or $t/n \geq 500$ (corresponding to the maximum number of epochs allowed). For GRAPHSTOIHT, one epoch contains n iterations. We recall that $\boldsymbol{A}$ has been partitioned into n blocks with block size b.

[3]Implementation details of the head and tail projection are provided in Supplementary material.

We say $\boldsymbol{x}^*$ is successfully recovered if the *estimation error* $\|\boldsymbol{x}^{t+1} - \boldsymbol{x}^*\| \leq 10^{-6}$.



(a) $s = 20$ (b) $s = 28$

Figure 1. Two simulated graphs. Subgraphs (colored nodes and edges) are generated by *random walk*. Each node v_i is associated with x_i. Colored-face nodes have values sampled from $\mathcal{N}(0,1)$ while white-face nodes have values 0.0.

To simulate a graph structure on $\boldsymbol{x}^* \in \mathbb{R}^p$ that mimics realistic subgraph patterns (e.g., malicious activities) in a network (Yu et al., 2016), we fix $p = 256$ and generate its s nonzero entries by using *random walk* (Lovász et al., 1993) on a 16×16 grid graph so that $\|\boldsymbol{x}^*\|_0$ forms a connected subgraph. Each edge has unit weight 1.0. Specifically, the procedure of *random walk* follows three main steps: 1) select an initial node (we choose the center of the grid); 2) move to its neighbor with probability $1/d(v_t)$, where $d(v_t)$ is the degree of node v_t; 3) repeat 2) until s different nodes have been visited. Figure 1 presents two *random walks* of $s = 20$ (on the left) and $s = 28$ (on the right).

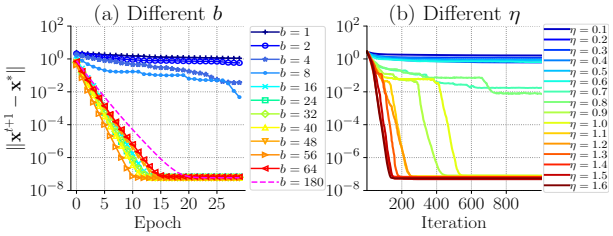


Figure 2. Choice of b and η. The left part illustrates the *estimation error* as a function of epochs for different choices of b. When $b = 180$, it degenerates to GRAPHIHT (the dashed line). The right part shows the *estimation error* as a function of iterations for different choices of η.

Choice of b and η. We first consider how block size b and learning rate η affect the performance of GRAPHSTOIHT. We fix the sparsity $s = 8$ and $n = 180$ and try different b from set $\{1, 2, 4, 8, 16, 24, 32, 40, 48, 56, 64, 180\}$. When $b = 1$, only 1 observation has been used at each iteration; when $b = 180$, all measurements used, corresponding to GRAPHIHT. Results are presented on the left of Figure 2. In order to successfully recover $\boldsymbol{x}^*$ within 30 epochs, the block size should be at least s. A potential explanation is that, since $b < s$, the Hessian matrix of f_{ξ_t}, i.e., $\boldsymbol{A}_{B_{\xi_t}}^\top \boldsymbol{A}_{B_{\xi_t}}$ is not positive definite. Thus it is hard to converge to $\boldsymbol{x}^*$ in a short time. Another interesting finding is that GRAPHSTOIHT converges faster than GRAPHIHT when block size

is suitable, say between 32 and 64. As shown in the Supplementary Material, the run time is dominated by the time of calculating the gradient when p and m are both increasing.

To further explore learning rate η, we use the similar setting ($s = 8$) above except $b = 8, m = 80$ (when $m = 80$, GRAPHIHT successfully recovers $\boldsymbol{x}^*$ with high probability as shown in Figure 4 (a). We consider 16 different η from set $\{0.1, 0.2, \ldots, 1.5, 1.6\}$. The right part of Figure 2 shows that GRAPHSTOIHT converges even when the learning rate is relatively large say $\eta \geq 1.5$. It means we can choose relatively larger learning rate so that the algorithm can achieve optimal solution faster. By Theorem 1, the optimal learning rate η is chosen by $\eta = 1/\beta$, so a potential explanation is that the strong smoothness parameter β is relatively small due to the head projection inequality $\|\nabla_H f_{\xi_t}(\boldsymbol{x}^{t+1}) - \nabla_H f_{\xi_t}(\boldsymbol{x}^*)\| \leq \beta \|\boldsymbol{x}^{t+1} - \boldsymbol{x}^*\|$.

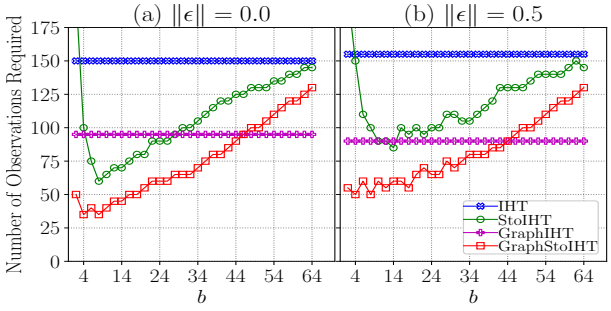


Figure 3. Robustness to noise $\boldsymbol{\epsilon}$. The number of observations required is a function of different block sizes.

Robustness to noise $\boldsymbol{\epsilon}$. To explore the performance under noise setting, we consider two noise conditions: $\|\boldsymbol{\epsilon}\| = 0.0$ (without noise) and $\|\boldsymbol{\epsilon}\| = 0.5$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. When $\|\boldsymbol{\epsilon}\| = 0.5$, the sparse vector is successfully recovered if $\|\boldsymbol{x}^{t+1} - \boldsymbol{x}^*\| \leq 0.5$. We try the block sizes b from the set $\{2, 4, 6, 8, 10, \ldots, 62, 64\}$, and then measure the number of observations m required by the algorithms (so that $\boldsymbol{x}^*$ can be successfully recovered with probability 1.0). The minimum number of observations is required such that the error $\|\boldsymbol{x}^{t+1} - \boldsymbol{x}^*\| \leq 10^{-6}$ for $\|\boldsymbol{\epsilon}\| = 0$ and $\|\boldsymbol{x}^{t+1} - \boldsymbol{x}^*\| \leq 0.5$ for $\|\boldsymbol{\epsilon}\| = 0.5$ in all trials. The results are reported in Figure 3. We can see both GRAPHSTOIHT and STOIHT are robust to noise. In particular, when block size is between 4 and 8, the number of observations required is the least for our method. Compared with the noiseless (the left), the performance of noise case (the right) degrades gracefully.

Results from synthetic dataset. We explore the performance of GRAPHSTOIHT on the *probability of recovery*, which is defined as the total number of successful trials divided by the total number of trimmed trials. Recall that $\boldsymbol{x}^*$ is successfully recovered if the *estimation error* $\|\boldsymbol{x}^{t+1} - \boldsymbol{x}^*\| \leq 10^{-6}$. We compare GRAPHSTOIHT with three baseline methods, i.e., Iterative Hard Thresholding (IHT) (Blumensath & Davies, 2009), Stochastic Iterative Hard Thresholding (STOIHT) (Nguyen et al., 2017), and GRAPHIHT (Hegde et al., 2016). We consider four different sparsity levels, i.e., $s \in \{8, 20, 28, 36\}$. For each trial, $\boldsymbol{x}^*$ is generated by *random walk*. To be consistent with the setting in Nguyen et al. (2017), the block size is set by $b = \min(s, m)$ with the number of observations m chosen from $\{5, 10, \ldots, 245, 250\}$. All of the four methods, including ours, use a constant learning rate $\eta = 1$. Our method uses less observations to successfully recover $\boldsymbol{x}^*$ due to the randomness (compared with IHT and GRAPHIHT) and the graph-structured projection (compared with IHT and STOIHT) as shown in Figure 4. It indicates that GRAPHSTOIHT outperforms the other three baselines in terms of probability of recovery. It also shows that SGD-based methods are more stable than batch methods with respect to small perturbation of data as recently shown by the works of Hardt et al. (2016) and Charles & Papailiopoulos (2018).

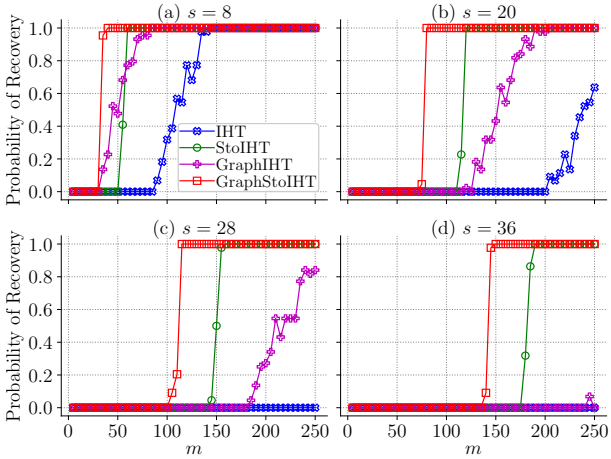


Figure 4. Probability of recovery on synthetic dataset. The probability of recovery is a function of number of observations m.

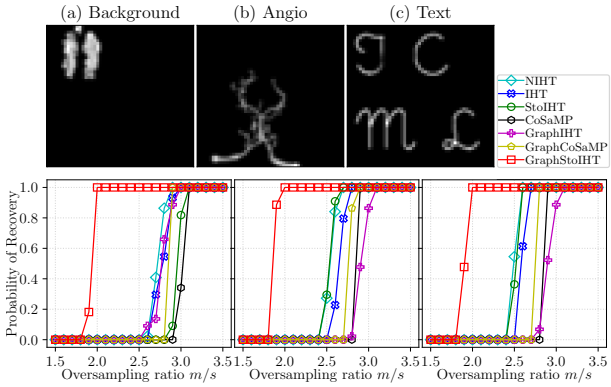


Figure 5. Probability of recovery on three 50×50 resized real images: (a) Background, (b) Angio, and (c) Text (Hegde et al., 2015b). The probability of recovery is a function of the oversampling ratio m/s.

Results from three real-world images. In order to further demonstrate the merit of GRAPHSTOIHT, we compare it with another three popular methods: NIHT (Blumen-

Table 2. AUC score ± standard deviation on the breast cancer dataset.

Folding ID	ℓ^1-PATHWAY	ℓ^1/ℓ^2-PATHWAY	ℓ^1-EDGE	ℓ^1/ℓ^2-EDGE	IHT	STOIHT	GRAPHIHT	GRAPHSTOIHT
Folding 00	0.705±0.09	0.715±0.07	0.726±0.07	0.724±0.07	0.726±0.07	**0.731**±0.06	0.716±0.02	0.718±0.03
Folding 01	0.665±0.11	**0.718**±0.03	0.688±0.04	0.703±0.07	0.680±0.07	0.683±0.07	0.710±0.06	0.704±0.05
Folding 02	0.640±0.07	**0.729**±0.06	0.714±0.04	0.717±0.04	0.716±0.06	0.723±0.06	0.724±0.07	0.720±0.08
Folding 03	0.691±0.05	0.705±0.05	0.720±0.05	**0.722**±0.04	0.699±0.05	0.703±0.05	0.687±0.04	0.687±0.04
Folding 04	0.680±0.05	0.690±0.05	0.705±0.06	**0.721**±0.05	0.694±0.05	0.695±0.06	0.687±0.05	0.688±0.03
Folding 05	0.648±0.07	0.694±0.04	0.658±0.06	0.683±0.07	0.671±0.07	0.671±0.07	0.675±0.07	**0.712**±0.05
Folding 06	0.682±0.04	0.733±0.04	0.701±0.05	0.701±0.06	0.712±0.07	0.733±0.06	**0.742**±0.06	0.741±0.06
Folding 07	0.674±0.04	0.682±0.07	0.695±0.04	0.704±0.03	0.711±0.08	0.704±0.07	**0.725**±0.07	0.714±0.08
Folding 08	0.686±0.06	0.705±0.06	0.696±0.07	0.691±0.07	0.724±0.07	0.720±0.07	0.703±0.03	**0.729**±0.03
Folding 09	0.671±0.07	0.690±0.07	0.660±0.05	0.687±0.03	**0.712**±0.06	0.712±0.06	0.703±0.06	0.704±0.06
Folding 10	0.693±0.09	**0.735**±0.09	0.706±0.10	0.718±0.06	0.701±0.08	0.717±0.09	0.710±0.08	0.707±0.07
Folding 11	0.669±0.04	0.697±0.07	0.711±0.05	0.704±0.05	0.707±0.05	0.706±0.04	**0.733**±0.06	0.733±0.06
Folding 12	0.670±0.05	**0.716**±0.06	0.703±0.05	0.701±0.03	0.714±0.06	0.715±0.07	0.711±0.07	0.711±0.08
Folding 13	0.678±0.07	0.688±0.04	0.703±0.04	0.697±0.04	0.699±0.06	0.701±0.06	0.703±0.04	**0.715**±0.05
Folding 14	0.653±0.02	0.700±0.02	0.703±0.03	0.692±0.03	0.695±0.04	0.699±0.04	**0.721**±0.06	0.721±0.06
Folding 15	0.663±0.07	0.682±0.07	0.697±0.08	0.687±0.07	0.724±0.07	0.712±0.06	**0.725**±0.07	0.716±0.07
Folding 16	0.687±0.07	0.720±0.04	0.697±0.06	**0.729**±0.06	0.721±0.05	0.723±0.05	0.719±0.04	0.715±0.04
Folding 17	0.684±0.05	0.703±0.05	0.677±0.07	0.716±0.05	0.712±0.04	0.712±0.04	**0.730**±0.03	0.720±0.04
Folding 18	0.660±0.06	0.714±0.06	0.675±0.08	0.685±0.08	0.713±0.06	0.706±0.06	0.735±0.05	**0.735**±0.05
Folding 19	0.694±0.02	0.692±0.04	**0.727**±0.05	0.713±0.04	0.719±0.02	0.703±0.03	0.725±0.05	0.715±0.05
Averaged	0.675±0.06	0.705±0.06	0.698±0.06	0.705±0.06	0.707±0.06	0.708±0.06	0.714±0.06	**0.715**±0.06

sath & Davies, 2010), COSAMP (Needell & Tropp, 2009), GRAPHCOSAMP (Hegde et al., 2015b). We test all of them on three 50×50 resized real images: Background, Angio, and Text provided in Hegde et al. (2015b), where the first two images have one connected component while Text has four. NIHT and COSAMP have sparsity s as input parameter. GRAPHSTOIHT shares the same head and tail projection as GRAPHCOSAMP. The learning rates η of IHT, STOIHT, GRAPHIHT and GRAPHSTOIHT are tuned from the set $\{0.2, 0.4, 0.6, 0.8\}$, and the block sizes b of STOIHT and GRAPHSTOIHT are tuned from the set $\{m/5, m/10\}$. We tune b and η on an additional validation dataset with 100 observations. To clarify, the design matrix $\boldsymbol{A}$ used here is Gaussian matrix, different from the Fourier Matrix used in Hegde et al. (2015b). Our method outperforms the others consistently.

5.2. Graph sparse logistic regression

To further test our method on real-world dataset, we apply it to the breast cancer dataset in Van De Vijver et al. (2002), which contains 295 training samples including 78 positives (metastatic) and 217 negatives (non-metastatic). Each $\boldsymbol{a}_i$ in (13) is the training sample of patient i with dimension $p = 8,141$ (genes). Label $y_i = 1$ if patient i has metastatic growth; otherwise -1. We use the Protein-Protein Interaction (PPI) network in Jacob et al. (2009)[4]. There are 637 pathways in this PPI network. We restrict our analysis on 3,243 genes (nodes) which form a connected graph with 19,938 edges. Due to lack of edge weights, all weights have been set to 1.0. We fold the dataset uniformly into 5 subfolds as done by Jenatton et al. (2011) to make comparison later. All related parameters are tuned by 5-fold-cross-validation on each training dataset. More experiment details are available in the Supplementary Material. We repeat the folding strategy 20 times, and Table 2 reports AUC scores with standard deviation.

Metastasis classification. We compare our algorithm with the three aforementioned non-convex based methods and four ℓ^1/ℓ^2 mixed norm-based algorithms, ℓ^1-PATHWAY, ℓ^1/ℓ^2-PATHWAY, ℓ^1-EDGE, and ℓ^1/ℓ^2-EDGE[5]. ℓ^1-PATHWAY and ℓ^1/ℓ^2-PATHWAY use pathways as groups while ℓ^1-EDGE and ℓ^1/ℓ^2-EDGE use edges as groups. On average, GRAPHSTOIHT achieves 0.715 AUC score, the highest among the eight methods.

Gene identification. We also investigate different numbers of breast cancer-related genes identified by each method. Out of 25 breast cancer-related genes, GRAPHSTOIHT finds 24% of them, more than GRAPHIHT (20%), STOIHT (16%) and IHT (8%). The mixed norm-based methods ℓ^1-PATHWAY, ℓ^1/ℓ^2-PATHWAY, ℓ^1-EDGE, and ℓ^1/ℓ^2-EDGE find 16%, 8%, 12%, and 12% respectively. Our method outperforms other methods in finding cancer-related genes.

6. Conclusion and Future Work

In this paper, we have proposed GRAPHSTOIHT, a stochastic gradient-based method for solving graph-structured sparsity constraint problems. We proved that it enjoys a linear convergence property. Experimental evaluation shows our method consistently outperforms other algorithms for both graph sparse linear regression and graph logistic regression on real-world datasets. In future, it would be interesting to see if one can apply the variance reduction techniques such as SAGA (Defazio et al., 2014) and SVRG (Johnson & Zhang, 2013) to GRAPHSTOIHT.

[4]This network was originally proposed by Chuang et al. (2007).

[5]The code of these four methods is sourced from `http://cbio.ensmp.fr/~ljacob/documents/overlasso-package.tgz`.

Acknowledgement

The work of Yiming Ying is supported by the National Science Foundation (NSF) under Grant No #1816227. The work of Baojian Zhou and Feng Chen is supported by the NSF under Grant No #1815696 and #1750911.

References

Agarwal, A., Negahban, S., Wainwright, M. J., et al. Fast global convergence of gradient methods for high-dimensional statistical recovery. *The Annals of Statistics*, 40(5):2452–2482, 2012.

Aksoylar, C., Orecchia, L., and Saligrama, V. Connected subgraph detection with mirror descent on sdps. In *International Conference on Machine Learning*, pp. 51–59, 2017.

Arias-Castro, E., Candès, E. J., and Durand, A. Detection of an anomalous cluster in a network. *The Annals of Statistics*, pp. 278–304, 2011.

Bach, F. and Moulines, E. Non-strongly-convex smooth stochastic approximation with convergence rate o (1/n). In *Advances in neural information processing systems*, pp. 773–781, 2013.

Bach, F., Jenatton, R., Mairal, J., Obozinski, G., et al. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012a.

Bach, F., Jenatton, R., Mairal, J., Obozinski, G., et al. Structured sparsity through convex optimization. *Statistical Science*, 27(4):450–468, 2012b.

Bahmani, S., Raj, B., and Boufounos, P. T. Greedy sparsity-constrained optimization. *Journal of Machine Learning Research*, 14(Mar):807–841, 2013.

Baraniuk, R. G., Cevher, V., Duarte, M. F., and Hegde, C. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010.

Blumensath, T. and Davies, M. E. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009.

Blumensath, T. and Davies, M. E. Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE Journal of selected topics in signal processing*, 4 (2):298–309, 2010.

Bousquet, O. and Bottou, L. The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pp. 161–168, 2008.

Candes, E. J. and Tao, T. Decoding by linear programming. *IEEE transactions on information theory*, 51(12):4203–4215, 2005.

Charles, Z. and Papailiopoulos, D. Stability and generalization of learning algorithms that converge to global optima. In *International Conference on Machine Learning*, pp. 744–753, 2018.

Chen, S. S., Donoho, D. L., and Saunders, M. A. Atomic decomposition by basis pursuit. *SIAM review*, 43(1): 129–159, 2001.

Chuang, H.-Y., Lee, E., Liu, Y.-T., Lee, D., and Ideker, T. Network-based classification of breast cancer metastasis. *Molecular systems biology*, 3(1):140, 2007.

Defazio, A., Bach, F., and Lacoste-Julien, S. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pp. 1646–1654, 2014.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Elenberg, E. R., Khanna, R., Dimakis, A. G., Negahban, S., et al. Restricted strong convexity implies weak submodularity. *The Annals of Statistics*, 46(6B):3539–3568, 2018.

Hardt, M., Recht, B., and Singer, Y. Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pp. 1225–1234, 2016.

Hegde, C., Indyk, P., and Schmidt, L. A fast approximation algorithm for tree-sparse recovery. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pp. 1842–1846. IEEE, 2014.

Hegde, C., Indyk, P., and Schmidt, L. Approximation algorithms for model-based compressive sensing. *IEEE Transactions on Information Theory*, 61(9):5129–5147, 2015a.

Hegde, C., Indyk, P., and Schmidt, L. A nearly-linear time framework for graph-structured sparsity. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 928–937, 2015b.

Hegde, C., Indyk, P., and Schmidt, L. Fast recovery from a union of subspaces. In *Advances in Neural Information Processing Systems*, pp. 4394–4402, 2016.

Jacob, L., Obozinski, G., and Vert, J.-P. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pp. 433–440. ACM, 2009.

Jain, P., Tewari, A., and Kar, P. On iterative hard thresholding methods for high-dimensional m-estimation. In *Advances in Neural Information Processing Systems*, pp. 685–693, 2014.

Jenatton, R., Audibert, J.-Y., and Bach, F. Structured variable selection with sparsity-inducing norms. *Journal of Machine Learning Research*, 12(Oct):2777–2824, 2011.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. How to escape saddle points efficiently. *arXiv preprint arXiv:1703.00887*, 2017.

Johnson, R. and Zhang, T. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Li, X. and Orabona, F. On the convergence of stochastic gradient descent with adaptive stepsizes. *arXiv preprint arXiv:1805.08114*, 2018.

Liu, B., Yuan, X.-T., Wang, L., Liu, Q., and Metaxas, D. N. Dual iterative hard thresholding: From non-convex sparse minimization to non-smooth concave maximization. In *International Conference on Machine Learning*, pp. 2179–2187, 2017.

Lovász, L. et al. Random walks on graphs: A survey. *Combinatorics*, 2(1):1–46, 1993.

Morales, J., Micchelli, C. A., and Pontil, M. A family of penalty functions for structured sparsity. In *Advances in Neural Information Processing Systems*, pp. 1612–1623, 2010.

Murata, T. and Suzuki, T. Sample efficient stochastic gradient iterative hard thresholding method for stochastic sparse linear regression with limited attribute observation. In *Advances in Neural Information Processing Systems*, pp. 5313–5322, 2018.

Needell, D. and Tropp, J. A. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and computational harmonic analysis*, 26(3):301–321, 2009.

Nguyen, N., Needell, D., and Woolf, T. Linear convergence of stochastic iterative greedy algorithms with sparse constraints. *IEEE Transactions on Information Theory*, 63 (11):6869–6895, 2017.

Obozinski, G. and Bach, F. A unified perspective on convex structured sparsity: Hierarchical, symmetric, submodular norms and beyond. *HAL*, 2016.

Qian, J., Saligrama, V., and Chen, Y. Connected sub-graph detection. In *Artificial Intelligence and Statistics*, pp. 796–804, 2014.

Rakhlin, A., Shamir, O., and Sridharan, K. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 449–456, 2012.

Rosasco, L., Villa, S., and Vũ, B. C. Convergence of stochastic proximal gradient algorithm. *arXiv preprint arXiv:1403.5074*, 2014.

Rozenshtein, P., Anagnostopoulos, A., Gionis, A., and Tatti, N. Event detection in activity networks. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1176–1185. ACM, 2014.

Shamir, O. and Zhang, T. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*, pp. 71–79, 2013.

Shen, J. and Li, P. A tight bound of hard thresholding. *The Journal of Machine Learning Research*, 18(1):7650–7691, 2017.

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

Turlach, B. A., Venables, W. N., and Wright, S. J. Simultaneous variable selection. *Technometrics*, 47(3):349–363, 2005.

Van De Vijver, M. J., He, Y. D., Van't Veer, L. J., Dai, H., Hart, A. A., Voskuil, D. W., Schreiber, G. J., Peterse, J. L., Roberts, C., Marton, M. J., et al. A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine*, 347(25):1999–2009, 2002.

Van Ngai, H. and Penot, J.-P. Paraconvex functions and paraconvex sets. *Studia Mathematica*, 184:1–29, 2008.

Yang, T. and Lin, Q. Rsg: Beating subgradient method without smoothness and strong convexity. *arXiv preprint arXiv:1512.03107*, 2015.

Ying, Y. and Zhou, D.-X. Online regularized classification algorithms. *IEEE Transactions on Information Theory*, 52(11):4775–4788, 2006.

Ying, Y. and Zhou, D.-X. Unregularized online learning algorithms with general loss functions. *Applied and Computational Harmonic Analysis*, 42(2):224–244, 2017.

Yu, R., Qiu, H., Wen, Z., Lin, C., and Liu, Y. A survey on social media anomaly detection. *ACM SIGKDD Explorations Newsletter*, 18(1):1–14, 2016.

Yuan, M. and Lin, Y. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68 (1):49–67, 2006.

Yuan, X.-T., Li, P., and Zhang, T. Gradient hard thresholding pursuit. *Journal of Machine Learning Research*, 18(166): 1–43, 2018.

Zhou, P., Yuan, X., and Feng, J. Efficient stochastic gradient hard thresholding. In *Advances in Neural Information Processing Systems*, pp. 1985–1994, 2018a.

Zhou, P., Yuan, X., and Feng, J. New insight into hybrid stochastic gradient descent: Beyond with-replacement sampling and convexity. In *Advances in Neural Information Processing Systems*, pp. 1242–1251, 2018b.