GIST: General Iterative Shrinkage and Thresholding for Non-convex Sparse Learning

Version 2.0

Pinghua Gong\(^1\), Changshui Zhang\(^1\)
Zhaosong Lu\(^2\), Jianhua Huang\(^3\), Jieping Ye\(^4\)

\(^1\)State Key Lab on Intelligent Technology and Systems
Tsinghua National Lab for Information Science and Technology
Department of Automation, Tsinghua Univ., Beijing 100084, China

\(^2\)Department of Mathematics, Simon Fraser University
Burnaby, BC, V5A 1S6, Canada

\(^3\)Department of Statistics, Texas A&M University, TX 77843, USA

\(^4\)Computer Science & Engineering
Center for Evolutionary Medicine and Informatics
The Biodesign Institute, Arizona State University
Tempe, AZ 85287, USA

\{gph08@mails, zcs@mail\}.tsinghua.edu.cn
zhaosong@sfu.ca, jianhua@stat.tamu.edu, jieping.ye@asu.edu

http://www.public.asu.edu/~jye02/Software/GIST

June 4, 2014
1 Introduction

Learning sparse representations has very important applications in real-world problems. In the last decade, $\ell_1$-norm based sparse learning [10, 3], by solving a convex optimization problem, has been extensively studied and successfully applied to many areas including signal & image processing [1, 13], computer vision [12], biomedical informatics [9] and so on. However, recent theoretical investigations have shown that $\ell_1$-norm based sparse learning achieves suboptimal performance in many cases [2, 15, 16]. To this end, many non-convex regularized sparse learning formulations have been proposed and shown their superiority over their convex counterparts in several sparse learning settings. In these non-convex sparse learning formulations, many non-convex regularizers (penalties) are employed, which include $\ell_q$-norm ($0 < q < 1$) [5], Smoothly Clipped Absolute Deviation (SCAD) [4], Log-Sum Penalty (LSP) [2], Minimax Concave Penalty (MCP) [14], Geman Penalty (GP) [6, 11] and Capped-$\ell_1$ penalty [15, 16, 7].

Although non-convex regularized sparse learning has some advantages over the convex ones, the main challenge is how to efficiently solve the corresponding non-convex optimization problem. In this package, we provide an efficient implementation called General Iterative Shrinkage and Theresholding (GIST) to solve non-convex optimization problems.

2 Optimization Problem and Algorithm

Our package provides implementations for the following non-convex sparse vector optimization problem:

$$\text{Sparse Vector : } \min_{w \in \mathbb{R}^d} \{ f(w) = l(w) + r(w) \},$$

where the loss function $l(w)$ and regularizer $r(w)$ implemented in our package are listed in Table 1 and Table 2, respectively.

We solve Eq. (1) by generating a sequence $\{w^{(k)}\}$ via:

$$w^{(k+1)} = \arg \min_w l(w^{(k)}) + \langle \nabla l(w^{(k)}), w - w^{(k)} \rangle + \frac{t^{(k)}}{2} \| w - w^{(k)} \|^2 + r(w),$$

which has a closed-form solution for all the regularizers listed in Table 2 [8]. The detailed procedure of the GIST algorithm is presented in Algorithm 1. It seems that the GIST algorithm is similar to SpaRSA algorithm [13]. The main difference is that the GIST algorithm can handle both non-convex and convex regularized optimization problems, while SpaRSA algorithm is proposed based on the convex regularization. Please refer to the literature [8] for more technical details.

The proposed GIST algorithm can also solve more extensive non-convex optimization problems, e.g., non-convex low rank matrix problems:

$$\text{Low Rank Matrix : } \min_{W \in \mathbb{R}^{d \times m}} \{ f(W) = l(W) + r(W) \},$$

where the loss function $l(W)$ and regularizer $r(W)$ implemented in our package are listed in Table 3 and Table 4, respectively. The key of the implementation is how to efficiently solve the
Table 1: (Sparse Vector) Loss functions \( l(w) \) implemented in our GIST package. \( X = [x_1^T; \cdots; x_n^T] \in \mathbb{R}^{n \times d} \) is a data matrix and \( y = [y_1, \cdots, y_n]^T \in \mathbb{R}^n \) is a target vector.

<table>
<thead>
<tr>
<th>Name</th>
<th>( l(w) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic loss</td>
<td>( \frac{1}{n} \sum_{i=1}^{n} \log (1 + \exp(-y_i x_i^T w)) )</td>
</tr>
<tr>
<td>L2 SVM loss</td>
<td>( \frac{1}{2n} \sum_{i=1}^{n} \max(0, 1 - y_i x_i^T w)^2 )</td>
</tr>
<tr>
<td>Least Square loss</td>
<td>( \frac{1}{2n} |Xw - y|^2 )</td>
</tr>
</tbody>
</table>

Table 2: (Sparse Vector) Regularizers (penalties) \( r(w) \) implemented in our GIST package. \( \lambda > 0 \) is the regularization parameter; \( r(w) = \sum_i r_i(w_i), [x]_+ = \max(0, x) \).

<table>
<thead>
<tr>
<th>Name</th>
<th>( r_i(w_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \ell_1 )</td>
<td>( \lambda</td>
</tr>
<tr>
<td>LSP</td>
<td>( \lambda \log(1 +</td>
</tr>
<tr>
<td>SCAD</td>
<td>( \lambda \int_0^{[w_i]} \min\left(1, \frac{\theta</td>
</tr>
</tbody>
</table>
|              | \( = \left\{ \begin{array}{ll} \lambda |w_i|, & \text{if } |w_i| \leq \lambda, \\
|              | \quad \frac{-w_i^2 + 2\theta \lambda |w_i| - \lambda^2}{2(\theta - 1)} & \text{if } \lambda < |w_i| \leq \theta \lambda, \\
|              | \quad (\theta + 1) \lambda^2 / 2, & \text{if } |w_i| > \theta \lambda. \\
| MCP          | \( \lambda \int_0^{[w_i]} \left[1 - \frac{x}{\theta \lambda} \right]_+ dx \) \( (\theta > 0) \) |
|              | \( = \left\{ \begin{array}{ll} \lambda |w_i| - w_i^2 / (2\theta), & \text{if } |w_i| \leq \theta \lambda, \\
|              | \quad \theta \lambda^2 / 2, & \text{if } |w_i| > \theta \lambda. \\
| Capped \( \ell_1 \) | \( \lambda \min(|w_i|, \theta) \) \( (\theta > 0) \)                              |

The proximal operator problem in Eq. (2) when \( r(W) \) is a non-convex low rank regularizer. For all regularizers listed in Table 4, the proximal operator problem has a closed form solution. Please refer to the full version of [8] for more details.

3 How to Use the GIST Package

3.1 Package Installation

The GIST package is currently implemented by Matlab (some functions are implemented by C). Before you use the package, make sure that the Matlab software is correctly installed (You may also need a C compiler to mex C files in Matlab). After that, please follow the following steps to install the GIST package.

1. Download the GIST package online\(^1\) and unzip it to a folder.
2. Run install.m in Matlab.

\(^1\)http://www.public.asu.edu/~jye02/Software/GIST
Algorithm 1: GIST: General Iterative Shrinkage and Thresholding

Input: \( w^{(0)} \in \mathbb{R}^d \)

1. Initialize \( \eta > 1, t_{\text{min}}, t_{\text{max}} \), where \( 0 < t_{\text{min}} < t_{\text{max}} \);
2. for \( k = 1, 2, \ldots, \text{maxiter} \) do
   3. \( t^{(k)} \in [t_{\text{min}}, t_{\text{max}}] \);
   4. repeat
      5. \( w^{(k+1)} \leftarrow \arg \min_w l(w^{(k)}) + \langle \nabla l(w^{(k)}), w - w^{(k)} \rangle + \frac{r^{(k)}}{2} \|w - w^{(k)}\|^2 + r(w) \);
      6. \( t^{(k)} \leftarrow \eta t^{(k)} \);
   7. until some line search criterion is satisfied;
   8. if some stopping criterion is satisfied then
      9. \( w^* = w^{(k)} ; \text{ iter} = k \);
     10. break;
   11. end
   12. end

Output: \( w^*, \text{ iter} \)

Table 3: (Low Rank Matrix) Loss functions \( l(W) \) implemented in our GIST package. \( X_i = [x_{i1}^T; \cdots; x_{in_i}^T] \in \mathbb{R}^{n_i \times d} \) is a data matrix of the \( i \)-th task; \( y_i = [y_{i1}; \cdots; y_{in_i}]^T \in \mathbb{R}^{n_i} \) is a target vector of the \( i \)-th task; \( W = [w_1; \cdots; w_m] \in \mathbb{R}^{d \times m} \) is a weight matrix; \( P_\Omega(W)(i, j) = w_{ij} \), if \((i, j) \in \Omega\), and 0 otherwise.

<table>
<thead>
<tr>
<th>Name</th>
<th>( l(W) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Multi-Task Learning loss</td>
<td>( \sum_{i=1}^m \sum_{j=1}^{n_i} \log (1 + \exp(-y_{ij}x_{ij}^Tw_i)) )</td>
</tr>
<tr>
<td>L2 SVM Multi-Task Learning loss</td>
<td>( \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^{n_i} \max(0, 1 - y_{ij}x_{ij}^Tw_i)^2 )</td>
</tr>
<tr>
<td>Least Square Multi-Task Learning loss</td>
<td>( \frac{1}{2} \sum_{i=1}^m |X_iw_i - y_i|^2 )</td>
</tr>
<tr>
<td>Multivariate Linear Regression loss</td>
<td>( \frac{1}{2} |XW - Y|_F^2, X \in \mathbb{R}^{n \times d}, Y \in \mathbb{R}^{n \times m} )</td>
</tr>
<tr>
<td>Matrix Completion loss</td>
<td>( \frac{1}{2} |P_\Omega(W) - P_\Omega(X)|_F^2 )</td>
</tr>
</tbody>
</table>

3.2 Package Structure

- GIST: includes all functions implemented in this package, which has three subfolders:
  - CFile: includes C files
  - Function: includes main functions (both sparse vector and low rank matrix cases)
  - Utility: includes some auxiliary functions
- Examples: includes some examples (both sparse vector and low rank matrix cases) to show how to use functions implemented in this package.
- Data: includes a data set used in examples.
- Manual: includes a manual on how to use this package.
Table 4: (Low Rank Matrix) Regularizers (penalties) \( r(W) \) implemented in our GIST package. \( \lambda > 0 \) is the regularization parameter; 
\( r(W) = \sum_i r_i(\sigma_i), [x]_+ = \max(0, x) \), where \( \sigma_i \) is the 
\( i \)-th largest singular value of \( W \) (note that \( \sigma_i \geq 0 \)).

<table>
<thead>
<tr>
<th>Name</th>
<th>( r_i(\sigma_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trace Norm</td>
<td>( \lambda \sigma_i )</td>
</tr>
<tr>
<td>LSP trace</td>
<td>( \lambda \log(1 + \sigma_i/\theta) ) (( \theta &gt; 0 ))</td>
</tr>
</tbody>
</table>
| SCAD trace         | \( \int_0^{\sigma_i} \min \left( 1, \frac{[\theta \lambda - x]_+}{(\theta - 1)\lambda} \right) dx \) (\( \theta > 2 \))
|                    | \( = \begin{cases} 
\lambda \sigma_i, & \text{if } \sigma_i \leq \lambda, \\
-\frac{\sigma_i^2 + 2\theta \lambda \sigma_i - \lambda^2}{2(\theta - 1)} & \text{if } \lambda < \sigma_i \leq \theta \lambda, \\
(\theta + 1)\lambda^2/2, & \text{if } \sigma_i > \theta \lambda. 
\end{cases} \) |
| MCP trace          | \( \int_0^{\sigma_i} [1 - \frac{x_+}{x}]_+ dx \) (\( \theta > 0 \))
|                    | \( = \begin{cases} 
\lambda \sigma_i - \sigma_i^2/(2\theta), & \text{if } \sigma_i \leq \theta \lambda, \\
\theta \lambda^2/2, & \text{if } \sigma_i > \theta \lambda. 
\end{cases} \) |
| Capped \( \ell_1 \) trace | \( \lambda \min(\sigma_i, \theta) \) (\( \theta > 0 \)) | 3.3 Package Interface
3.3.1 Sparse Vector

All functions for the sparse vector case have the following interface:

\[
[w, \text{fun}, \text{time}, \text{iter}] = \text{gistLossName}(X, y, \lambda, \theta, \text{varargin})
\]

LossName is one of the three loss functions in Table 1:

- Logistic: Logistic loss
- Least: Least Square loss
- L2SVM: L2 SVM loss

Input (Sparse Vector)

- X: data matrix with each row as a sample
- y: target vector
- lambda: regularization parameter
- theta: thresholding parameter
- varargin: optional parameters which must be passed in pair, e.g., 'parameterName', parameterValue, 'parameterName', parameterValue, ·····
  - 'regtype': nonconvex regularization type
    1: CapL1 (default)
    2: LSP
    3: SCAD
    4: MCP
- 'stopcriterion': stopping criterion
  1: relative difference of objective functions is less than tol (default)
  0: relative difference of iterative weights is less than tol
- 'startingpoint': starting point (default: zero vector)
- 'tolerance': stopping tolerance (default: 1e-5)
- 'maxiteration': number of maximum iteration (default: 1000)
- 'tinitialization': initialization of t (default: 1)
- 'tmin': tmin parameter (default: 1e-20)
- 'tmax': tmax parameter (default: 1e-20)
- 'eta': eta factor (default: 2)
- 'sigma': parameter in the line search (default: 1e-5)
- 'nonmonotone': nonmonotone steps in the line search (default: 5)
- 'stopnum': number of satisfying stopping criterion (default: 3)
- 'maxinneriter': number of maximum inner iteration (line search) (default: 20)

\textbf{Output (Sparse Vector)}

- w: output weight vector
- fun: a vector including all function values at each iteration
- time: a vector including all CPU times at each iteration
- iter: the number of iterative steps

\textbf{Remark 1} If you want to solve the $\ell_1$-regularized sparse learning problem using GIST package, please set 'regtype' as 1 (Capped L1) and set the theta parameter as $+\infty$ (or a very large number).

3.3.2 Low Rank Matrix

All functions (except gistMLR and gistMatComp functions\textsuperscript{2}) for the low rank matrix case have the following interface:

$$[W, \text{fun}, \text{time}, \text{iter}] = \text{gistLossName}(X, y, \text{samplesize}, \lambda, \theta, \text{varargin})$$

\textbf{LossName} is one of the three loss functions in Table 3:

- LogisticMTL: Logistic Multi-Task Learning loss
- LeastMTL: Least Squaure Multi-Task Learning loss

\textsuperscript{2}$$[W, \text{fun}, \text{time}, \text{iter}] = \text{gistMLR}(X, Y, \lambda, \theta, \text{varargin}),$$ where $X \in \mathbb{R}^{n \times d}$ is a data matrix (design matrix) and $Y \in \mathbb{R}^{n \times m}$ is a target matrix; the loss function is the Multivariate Linear Regression loss in Table 3. All other input and output parameters are the same.

$$[W, \text{fun}, \text{time}, \text{iter}] = \text{gistMatComp}(X, \lambda, \theta, \text{varargin}),$$ where $X \in \mathbb{R}^{n \times d}$ is an observed matrix (X is a sparse matrix in which missing entries are filled with zeros); $W \in \mathbb{R}^{n \times d}$ is a recovered matrix; the loss function is the Matrix Completion loss in Table 3. All other input and output parameters are the same.
• L2SVMMTL: L2 SVM Multi-Task Learning loss

**Input (Low Rank Matrix)**
- X: \( \sum_i n_i \times d \) data matrix which is stacked by data matrices of all tasks
- y: \( \sum_i n_i \times 1 \) response vector which is stacked by responses of all tasks
- samplesize: \( m \times 1 \) vector; the \( i \)-th entry is the sample size of the \( i \)-th task
- lambda: regularization parameter
- theta: thresholding parameter
- varargin: **optional parameters** which must be passed in pair, e.g., 'parameterName', parameterValue, 'parameterName', parameterValue, ······
  - 'regtype': nonconvex regularization type
    1: CapL1 trace (default)
    2: LSP trace
    3: SCAD trace
    4: MCP trace
  - 'stopcriterion': stopping criterion
    1: relative difference of objective functions is less than tol (default)
    0: relative difference of iterative weights is less than tol
  - 'startingpoint': starting point (default: zero matrix)
  - 'tolerance': stopping tolerance (default: 1e-5)
  - 'maxiteration': number of maximum iteration (default: 1000)
  - 'tinitialization': initialization of t (default: 1)
  - 'tmin': tmin parameter (default: 1e-20)
  - 'tmax': tmax parameter (default: 1e-20)
  - 'eta': eta factor (default: 2)
  - 'sigma': parameter in the line search (default: 1e-5)
  - 'nonmonotone': nonmonotone steps in the line search (default: 5)
  - 'stopnum': number of satisfying stopping criterion (default: 3)
  - 'maxinneriter': number of maximum inner iteration (line search) (default: 20)

**Output (Low Rank Matrix)**
- W: output weight matrix
- fun: a vector including all function values at each iteration
- time: a vector including all CPU times at each iteration
- iter: the number of iterative steps

**Remark 2** If you want to solve the trace norm regularized low rank learning problem using GIST package, please set 'regtype' as 1 (Capped L1 trace) and set the theta parameter as \(+\infty\) (or a very large number).
4 Examples

To illustrate how to use the functions included in the GIST package, we provide some examples in the folder "Examples".

5 Citation

In citing GIST in your papers, please use the following references:


If you use Latex, you can enter the following bibtex entries:

@MANUAL{gong2013gist,
title= {GIST: General Iterative Shrinkage and Thresholding for Non-convex Sparse Learning},
author= {Gong, P. and Zhang, C. and Lu, Z. and Huang, J. and Ye, J.},
orGANIZATION= {Tsinghua University},
year= {2013},
url= {http://www.public.asu.edu/~jye02/Software/GIST}
}

@INPROCEEDINGS{gong2013general,
title= {A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems},
author= {Gong, P. and Zhang, C. and Lu, Z. and Huang, J. and Ye, J.},
booktitle= {The 30th International Conference on Machine Learning (ICML)},
pages= {37-45}
year= {2013}
}

6 Acknowledgement

This software project is supported partly by 973 Program (2013CB329503), NSFC (Grant No. 91120301, 61075004, 61021063), NIH (R01 LM010730) and NSF (IIS-0953662, CCF-1025177).
References


