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

Version 1.0

[1]Pinghua Gong, [1]Changshui Zhang
[2]Zhaosong Lu, [3]Jianhua Huang, [4]Jieping Ye

[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`

March 14, 2013

1 Introduction

Learning sparse representations has very important applications in real-world problems. In the last decade, ℓ_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 ℓ_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 ℓ_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-ℓ_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 Thereshholding (GIST) to solve non-convex optimization problems.

2 Optimization Problem and Algorithm

Our package provides implementations for the following optimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \left\{ f(\mathbf{w}) = l(\mathbf{w}) + r(\mathbf{w}) \right\}, \tag{1}$$

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

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

$$\begin{aligned} \mathbf{w}^{(k+1)} = &\arg\min_{\mathbf{w}} \; l(\mathbf{w}^{(k)}) + \langle \nabla l(\mathbf{w}^{(k)}), \mathbf{w} - \mathbf{w}^{(k)} \rangle \\ &+ \frac{t^{(k)}}{2} \|\mathbf{w} - \mathbf{w}^{(k)}\|^2 + r(\mathbf{w}), \end{aligned} \tag{2}$$

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

Table 1: Loss functions $l(\mathbf{w})$ implemented in our GIST package. $X = [\mathbf{x}_1^T; \cdots; \mathbf{x}_n^T] \in \mathbb{R}^{n\times d}$ is a data matrix and $\mathbf{y} = [y_1, \cdots, y_n]^T \in \mathbb{R}^n$ is a target vector.

Name	$l(\mathbf{w})$
Logistic loss	$\frac{1}{n}\sum_{i=1}^n \log\left(1+\exp(-y_i\mathbf{x}_i^T\mathbf{w})\right)$
L2 SVM loss	$\frac{1}{2n}\sum_{i=1}^n \max\left(0, 1-y_i\mathbf{x}_i^T\mathbf{w}\right)^2$
Least Square loss	$\frac{1}{2n}\|X\mathbf{w}-\mathbf{y}\|^2$

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.

Algorithm 1: GIST: General Iterative Shrinkage and Thresholding

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

Initialize $\eta > 1, t_{\min}, t_{\max}$, where $0 < t_{\min} < t_{\max}$;
for $k = 1, 2, \cdots, \text{maxiter}$ **do**
  $t^{(k)} \in [t_{\min}, t_{\max}]$;
  repeat
    $$\mathbf{w}^{(k+1)} \leftarrow \arg\min_{\mathbf{w}} \; l(\mathbf{w}^{(k)}) + \langle \nabla l(\mathbf{w}^{(k)}), \mathbf{w}-\mathbf{w}^{(k)}\rangle + \frac{t^{(k)}}{2}\|\mathbf{w}-\mathbf{w}^{(k)}\|^2 + r(\mathbf{w});$$
    $t^{(k)} \leftarrow \eta t^{(k)}$;
  until *some line search criterion is satisfied*;
  if *some stopping criterion is satisfied* **then**
    $\mathbf{w}^\star = \mathbf{w}^{(k)}$; $iter = k$;
    break;
  end
end

Output: $\mathbf{w}^\star$, *iter*

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

Name	$r_i(w_i)$
ℓ_1	$\lambda\|w_i\|$
LSP	$\lambda \log(1 + \|w_i\|/\theta)\ (\theta > 0)$
SCAD	$\lambda \int_0^{\|w_i\|} \min\left(1, \frac{[\theta\lambda - x]_+}{(\theta-1)\lambda}\right) dx\ (\theta > 2)$ $= \begin{cases} \lambda\|w_i\|, & \text{if } \|w_i\| \le \lambda, \\ \frac{-w_i^2 + 2\theta\lambda\|w_i\| - \lambda^2}{2(\theta-1)}, & \text{if } \lambda < \|w_i\| \le \theta\lambda, \\ (\theta+1)\lambda^2/2, & \text{if } \|w_i\| > \theta\lambda. \end{cases}$
MCP	$\lambda \int_0^{\|w_i\|} \left[1 - \frac{x}{\theta\lambda}\right]_+ dx\ (\theta > 0)$ $= \begin{cases} \lambda\|w_i\| - w_i^2/(2\theta), & \text{if } \|w_i\| \le \theta\lambda, \\ \theta\lambda^2/2, & \text{if } \|w_i\| > \theta\lambda. \end{cases}$
Capped ℓ_1	$\lambda \min(\|w_i\|, \theta)\ (\theta > 0)$

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.

3.2 Package Structure

- GIST: includes all functions implemented in this package.
- examples: includes some examples to show how to use this package.
- data: includes a data set used in examples.
- manual: includes a manual on how to use this package.

[1] `http://www.public.asu.edu/~jye02/Software/GIST`

3.3 Package Interface

All functions included in the current package have the following interface:

[w,fun,time,iter] = gist**LossName**(X, y, lambda, theta, varargin)

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

- Logistic: logistic loss
- Least: Least Sqaure loss
- L2SVM: L2 SVM loss

3.3.1 Input

- X: data matrix with each row as a sample
- y: target vector
- lambda: regularization parameter
- theta: theresholding 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)

3.3.2 Output

- 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

Remark 1 *If you want to solve the ℓ_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).*

4 Examples

To illustrate how to use the functions included in the GIST package, we provide a simple example as follows:

```
% Before you run this example, make sure that you have run install.m
% to add the path and mex C files

clear;
clc;
close all;

% load data
Data = load ('..\data\classic_binary.mat');

y = Data.L';
X = Data.X';

clear Data
```

```
% statistics of the data
[n,d] = size(X);

% input parameters
lambda = 1e-3*abs(randn);
theta = 1e-2*lambda*abs(randn);

% optional parameter settings

regtype = 1; % nonconvex regularization type (default: 1 [capped L1])
w0 = randn(d,1); % starting point (default: zero vector)
stopcriterion = 0; % stopping criterion (default: 1)
maxiter = 1000; % number of maximum iteration (default: 1000)
tol = 1e-5; % stopping tolerance (default: 1e-5)
M = 5; % nonmonotone steps (default: 5)
t = 1; % initialization of t (default: 1)
tmin = 1e-20; % tmin parameter (default: 1e-20)
tmax = 1e20; % tmax parameter (default: 1e20)
sigma = 1e-5; % parameter in the line search (default: 1e-5)
eta = 2; % eta factor (default: 2)
stopnum = 3; % number of satisfying stopping criterion (default: 3)
maxinneriter = 20; % number of maximum inner iteration (line search) (default: 20)


% call the function
[w,fun,time,iter] = gistLogistic(X,y,lambda,theta,...
                                  'maxiteration',maxiter,...
                                  'regtype',regtype,...
                                  'stopcriterion', stopcriterion,...
                                  'tolerance',tol,...
                                  'startingpoint',w0,...
                                  'nonmonotone',M,...
                                  'tinitialization',t,...
                                  'tmin',tmin,...
                                  'tmax',tmax,...
                                  'sigma',sigma,...
                                  'eta',eta,...
                                  'stopnum',stopnum,...
                                  'maxinneriter',maxinneriter);


% plot
figure
semilogy(time(1:iter+1),fun(1:iter+1),'r-','LineWidth', 2)
xlabel('CPU time (seconds)')
ylabel('Objective function value (log scaled)')
legend('GIST-Logistic')
```

Citation

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

- P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye. GIST: General Iterative Shrinkage and Thresholding for Non-convex Sparse Learning. Tsinghua University, 2013. `http://www.public.asu.edu/~jye02/Software/GIST`
- P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye. A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. ICML 2013.

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= {ICML},
year= {2013}
}
```

5 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

[1] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[2] E. Candes, M. Wakin, and S. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

[3] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[4] J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.

[5] S. Foucart and M. Lai. Sparsest solutions of underdetermined linear systems via ℓ_q-minimization for $0 < q \leq 1$. *Applied and Computational Harmonic Analysis*, 26(3):395–407, 2009.

[6] D. Geman and C. Yang. Nonlinear image recovery with half-quadratic regularization. *IEEE Transactions on Image Processing*, 4(7):932–946, 1995.

[7] P. Gong, J. Ye, and C. Zhang. Multi-stage multi-task feature learning. In *NIPS*, pages 1997–2005, 2012.

[8] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. In *ICML*, 2013.

[9] S. Shevade and S. Keerthi. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246–2253, 2003.

[10] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[11] J. Trzasko and A. Manduca. Relaxed conditions for sparse signal recovery with general concave priors. *IEEE Transactions on Signal Processing*, 57(11):4347–4354, 2009.

[12] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–227, 2008.

[13] S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

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

[15] T. Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11:1081–1107, 2010.

[16] T. Zhang. Multi-stage convex relaxation for feature selection. *Bernoulli*, 2012.