

An R Package **flare** for High Dimensional Linear Regression and Precision Matrix Estimation

Xingguo Li^{*†} Tuo Zhao^{*‡} Xiaoming Yuan[§] Han Liu[¶]

Abstract

This paper describes an R package named **flare**, which implements a family of new high dimensional regression methods (LAD lasso, SQRT lasso, L_q lasso and Dantzig selector) and their extensions to sparse precision matrix estimation (TIGER and CLIME). The proposed solver is based on the alternating direction method of multipliers (ADMM), which is further combined with the linearization and coordinate descent algorithm. The package **flare** is coded in C and has a friendly user interface. The memory usage is optimized by using the sparse matrix output. The experiments show that **flare** is efficient and can scale up to large problems.

1 Introduction

As a popular sparse linear regression method for high dimensional data analysis, lasso has been extensively studied by machine learning and statistics communities [9]. It adopts the L_1 -penalized least square formulation to estimate and select non-zero parameters simultaneously. It further generated a wide range of research interests, and motivated many variants using non-smooth loss functions such as L_1 loss and L_2 loss functions. However, these non-smooth loss functions pose a great challenge to computation. To the best of our knowledge, no efficient solver has been developed so far, while several software packages such **glmnet** has been developed to solve large scale lasso problem [5].

In this paper, we describe a newly developed R package named **flare** (tuning optimized regression under sparsity constraints), which implements a family of new linear regression methods including LAD lasso [10], SQRT lasso [1], L_q lasso and Dantzig selector [4]. By using the column by column regression scheme, we further extends these regression methods to sparse precision matrix estimation problems including CLIME [3] and TIGER [7]. The proposed solver is based on the alternating direction method of multipliers (ADMM) combined with the linearization and efficient coordinate descent algorithm. The global convergence result of ADMM has been established in [6]. The numerical simulations show that **flare** is very efficient and can scale up to large problems.

^{*}Xingguo Li and Tuo Zhao contributed equally to this work;

[†]Department of Mathematics and Statistics, University of Minnesota Duluth;

[‡]Department of Computer Science, Johns Hopkins University;

[§]Department of Mathematics, Hong Kong Baptist University;

[¶]Department of Operations Research and Financial Engineering, Princeton University.

2 Notation

Before we proceed with the technical background, we first start with some notations. Let $\mathbf{A} = [\mathbf{A}_{jk}] \in \mathbb{R}^{d \times d}$, and $\mathbf{v} = (v_1, \dots, v_d) \in \mathbb{R}^d$. $\Lambda_{\min}(\mathbf{A})$ and $\Lambda_{\max}(\mathbf{A})$ denote the smallest and largest eigenvalues of \mathbf{A} . We define vector norms:

$$\|\mathbf{v}\|_1 = \sum_j |v_j|, \quad \|\mathbf{v}\|_2^2 = \sum_j v_j^2, \quad \|\mathbf{v}\|_\infty = \max_j |v_j|.$$

We also define the following winterization, univariate and group soft-thresholding operators denoted by \mathcal{W} , \mathcal{S} and \mathcal{G} respectively

$$\begin{aligned} \mathcal{W}(\mathbf{v}, \lambda) &= [\text{sign}(v_j) \cdot \min\{|v_j|, \lambda\}]_{j=1}^d, \\ \mathcal{S}(\mathbf{v}, \lambda) &= [\text{sign}(v_j) \cdot \max\{|v_j| - \lambda, 0\}]_{j=1}^d, \\ \mathcal{G}(\mathbf{v}, \lambda) &= \frac{\mathbf{v}}{\|\mathbf{v}\|_2} \cdot \max\{\|\mathbf{v}\|_2 - \lambda, 0\}. \end{aligned}$$

3 Algorithm

Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \mathbb{R}^n$ be the design matrix and response vector respectively, we aim to solve the following optimization problem,

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{argmin}} L(\mathbf{r}, \lambda) + \|\boldsymbol{\beta}\|_1 \quad \text{s.t. } \mathbf{b} - \mathbf{A}\boldsymbol{\beta} = \mathbf{r}. \quad (1)$$

where $\lambda > 0$ is the regularization parameter. The corresponding $L(\mathbf{r}, \lambda)$, \mathbf{A} and \mathbf{b} for different regression methods are shown in Table 1. As can be seen, LAD lasso and SQRT lasso are equivalent to L_q lasso with $q = 1$ and $q = 2$ respectively. All methods above can be efficiently solved by the

Table 1: Regression methods in **flare**.

Method	Loss function	\mathbf{A}	\mathbf{b}	Existing solver
LAD lasso	$L(\mathbf{r}, \lambda) = \frac{1}{n\lambda} \ \mathbf{r}\ _1$	\mathbf{X}	\mathbf{y}	L.P.
SQRT lasso	$L(\mathbf{r}, \lambda) = \frac{1}{\sqrt{n}\lambda} \ \mathbf{r}\ _2$	\mathbf{X}	\mathbf{y}	S.O.C.P.
L_q lasso	$L(\mathbf{r}, \lambda) = \frac{1}{\sqrt[q]{n}\lambda} \ \mathbf{r}\ _q$	\mathbf{X}	\mathbf{y}	None
Dantzig selector	$L(\mathbf{r}, \lambda) = \begin{cases} \infty & \text{if } \ \mathbf{r}\ _\infty > \lambda \\ 0 & \text{otherwise} \end{cases}$	$\mathbf{X}^T \mathbf{X}$	$\mathbf{X}^T \mathbf{y}$	L.P.

following updating scheme

$$\mathbf{r}^{t+1} = \underset{\mathbf{r}}{\text{argmin}} \frac{1}{2} \|\mathbf{u}^t / \rho + \mathbf{b} - \mathbf{A}\boldsymbol{\beta}^t - \mathbf{r}\|_2^2 + \frac{1}{\rho} L(\mathbf{r}, \lambda), \quad (2)$$

$$\boldsymbol{\beta}^{t+1} = \underset{\boldsymbol{\beta}}{\text{argmin}} \frac{1}{2} \|\mathbf{u}^t / \rho - \mathbf{r}^{t+1} + \mathbf{b} - \mathbf{A}\boldsymbol{\beta}\|_2^2 + \frac{1}{\rho} \|\boldsymbol{\beta}\|_1, \quad (3)$$

$$\mathbf{u}^{t+1} = \mathbf{u}^t + \rho(\mathbf{b} - \mathbf{r}^{t+1} - \mathbf{A}\boldsymbol{\beta}^{t+1}), \quad (4)$$

where \mathbf{u} is the Lagrange multiplier, and $\rho > 0$ is the penalty parameter. Existing ADMM based algorithms usually choose a fixed ρ , while in the package “flare”, we dynamically adjust ρ to well balance the timing performance and precision. For LAD lasso, SQRT lasso, and Dantzig selector, (2) has closed form solutions as follows,

$$\text{LAD lasso} \quad : \mathbf{r}^{t+1} = \mathcal{S}(\mathbf{u}^t/\rho + \mathbf{b} - \mathbf{A}\beta^t, 1/\rho), \quad (5)$$

$$\text{SQRT lasso} \quad : \mathbf{r}^{t+1} = \mathcal{G}(\mathbf{u}^t/\rho + \mathbf{b} - \mathbf{A}\beta^t, 1/\rho), \quad (6)$$

$$\text{Dantzig selector} \quad : \mathbf{r}^{t+1} = \mathcal{W}(\mathbf{u}^t/\rho + \mathbf{b} - \mathbf{A}\beta^t, \lambda). \quad (7)$$

For L_q lasso with $1 < q < 2$, we can solve it by the bi-section based root finding algorithm [8]. (3) is a standard L_1 penalized least square problem. For simplicity, existing ADMM based algorithms usually adopt the linearization at $\beta = \beta^t$ as follows and solve (3) approximately.

$$\beta^{t+1} = \underset{\beta}{\operatorname{argmin}} \frac{1}{2} \|\beta - \beta^t + \mathbf{A}^T(\mathbf{A}\beta^t - \mathbf{u}^t + \mathbf{r}^{t+1} - \mathbf{b})/\gamma\|_2^2 + \frac{1}{\rho} \|\beta\|_1, \quad (8)$$

where $\gamma \geq \Lambda_{\max}(\mathbf{A}^T \mathbf{A})$. (8) has a closed form solution

$$\beta^{t+1} = \mathcal{S}\left(\beta^t - \mathbf{A}^T(\mathbf{A}\beta^t - \mathbf{u}^t + \mathbf{r}^{t+1} - \mathbf{b})/\gamma, \frac{1}{\rho}\right). \quad (9)$$

However, such a linearization step is too conservative, and usually makes the algorithm convergence slowly in practice. To overcome this drawback, we propose an acceleration scheme to further boost the empirical performance:

- (I) We first solve (3) exactly by the efficient coordinate descent algorithm combined with active set and the covariance update tricks suggested by [5];
- (II) After a few iterations, when the obtained solution is close to the optimum, we switch to the linearization and solve (3) approximately.

Such a hybrid procedure is motivated by a very common phenomenon in convex optimization [2]. At the early stage of iterations, when the solution is far from the optimum, we can gain progress by taking an aggressive step (solving (3) exactly). After a few iterations, when the solution is close to the optimum, we need to take a conservative step (the linearization) to avoid oscillating around the optimum.

4 Examples

We illustrate the user interface by two examples. The first one is the eye disease dataset in our package.

```
> library(flare); data(eyedata) # Load the dataset
> out1 = flare.slim(x,y,method="lq",q=1.5,nlambda=40,lambda.min.ratio=0.25)
> out2 = flare.slim(x,y,method="dantzig",nlambda=20,lambda.min.ratio=0.9)
> plot(out1); plot(out2) # Plot solution paths
```

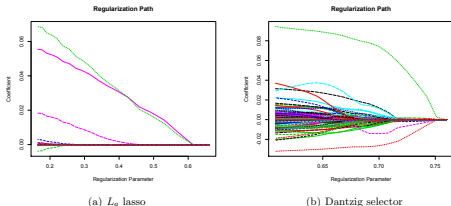


Figure 1: Solution paths obtained by the package **flare**

The program automatically generates a sequence of 40 regularization parameters and estimates the corresponding solution paths of L_q lasso and Dantzig selector. We further plot two solution paths in Figure 1, and we see that L_q lasso has a stable and sparse path, while the solutions of Dantzig selector may dramatically change along the path.

Our second example is the simulated dataset using the data generator in our package.

```
> # Generate data with hub structure
> L = flare.tiger.generator(n=200,d=200,graph="hub",g=10)
> out1 = flare.tiger(L$data,method="clime",nlmbda=10,lambda.min.ratio=0.5)
> # Model selection using cross validation.
> out1.opt = flare.tiger.select(out1,criterion="cv")
> out2 = flare.tiger(L$data,lambda = 2*sqrt(log(200)/400))
> # Visualize obtained grpahs
> plot(L); plot(out1.opt); plot(out2)
```

For CLIME, the program automatically generates a sequence of 10 regularization parameters, estimates the corresponding graph path, and chooses the optimal regularization parameter by cross validation. For TIGER, we manually choose the regularization to be $2\sqrt{\log(d)/n}$, and we further compare the obtained graphs with the ground truth using the visualization functions in our package, and the resulting figures are shown in Figures 2.

5 Numerical Simulation

All experiments below were carried out on a PC with Intel Core i5 3.3GHz processor, and the convergence threshold of **flare** is chosen to be 10^{-3} . Timings (in seconds) are averaged over 50 replications using a sequence of 5 regularization parameters, and the range of regularization parameters is chosen so that each method produced approximately the same number of non-zero estimates.

We first evaluate the timing performance of our package for linear regression. We set $n = 100$ and vary d from 500 to 4000 as is shown in Table 2. We independently generate each row of

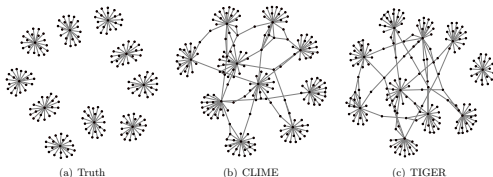


Figure 2: Graphs estimated by the package **flare**

the design matrix from a d -dimensional normal distribution $N(0, \Sigma)$, where $\Sigma_{jk} = 0.5^{|j-k|}$. Then we generate the response vector using $y_i = 3\mathbf{X}_{i1} + 2\mathbf{X}_{i2} + 1.5\mathbf{X}_{i4} + \epsilon_i$, where ϵ_i is independently generated from $N(0, 1)$. As can be seen from Table 2, LAD lasso, SQRT lasso and $L_{1.5}$ lasso all achieve very good timing performance. Dantzig selector is slower than others, but still much faster than general linear program solver.

We then evaluate the timing performance of our package for sparse precision matrix estimation. We set $n = 200$ and vary d from 100 to 800 as is shown in Table 2. We independently generate the data from a d -dimensional normal distribution $N(0, \Sigma)$, where $\Sigma_{jk} = 0.5^{|j-k|}$. The corresponding precision matrix $\Omega = \Sigma^{-1}$ has $\Omega_{jj} = 1.3333$, $\Omega_{jk} = -0.6667$ for all $j, k = 1, \dots, d$ and $|j - k| = 1$, and all other entries are 0. As can be seen from Table 2, TIGER and CLIME both achieve good timing performance, and CLIME is slower than TIGER due to a more difficult formulation.

Table 2: Average timing performance (in seconds) with standard errors in the parentheses on sparse linear regression and sparse precision matrix estimation.

Sparse Linear Regression				
Method	$d = 500$	$d = 1000$	$d = 2000$	$d = 4000$
LAD lasso	0.1243(0.0404)	0.2944(0.0488)	0.6934(0.0561)	2.2351(0.0859)
SQRT lasso	0.0564(0.0018)	0.1213(0.0366)	0.6499(0.0485)	2.1836(0.0556)
$L_{1.5}$ lasso	0.5664(0.0448)	0.6337(0.0043)	1.0176(0.0459)	2.4514(0.0674)
Dantzig selector	0.2407(0.0740)	4.2093(1.0691)	20.519(1.1620)	120.06(2.0016)
Sparse Precision Matrix Estimation				
Method	$d = 100$	$d = 200$	$d = 400$	$d = 800$
TIGER	2.9412(0.5437)	5.8062(0.1269)	19.903(0.2034)	138.49(1.9792)
CLIME	0.4556(0.0436)	2.575(0.0922)	25.487(0.5885)	388.09(8.9558)

6 Conclusion

Our developed package **flare** is complementary to the existing **glmnet** package. It provides the implementation of several new regression methods. We will maintain and support this package in

the future.

References

- [1] A. Belloni, V. Chernozhukov, and L. Wang. Square-root lasso: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98(4):791–806, 2011.
- [2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, the second edition edition, 2009.
- [3] T. Cai, W. Liu, and X. Luo. A constrained ℓ_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106:594–607, 2011.
- [4] E. Candes and T. Tao. The dantzig selector: Statistical estimation when p is much larger than n . *The Annals of Statistics*, 35(6):2313–2351, 2007.
- [5] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.
- [6] B. He and X. Yuan. On non-ergodic convergence rate of douglas-rachford alternating direction method of multipliers. Technical report, Tech. rep, 2012.
- [7] H. Liu and L. Wang. Tiger: A tuning-insensitive approach for optimally estimating gaussian graphical models. Technical report, Massachusetts Institute of Technology, 2012.
- [8] J. Liu and J. Ye. Efficient l_1/l_q norm regularization. Technical report, Arizona State University, 2010.
- [9] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1996.
- [10] Lie Wang. L1 penalized lad estimator for high dimensional linear regression. Technical report, Massachusetts Institute of Technology, 2012.