The `picasso` Package for Nonconvex Regularized M-estimation in High Dimensions in `R`

Xingguo Li* Tuo Zhao† Tong Zhang‡ Han Liu§

Abstract

We describe an `R` package named `picasso`, which implements a unified framework of pathwise coordinate optimization for a variety of sparse learning problems (Sparse Linear Regression, Sparse Logistic Regression and Sparse Column Inverse Operator), combined with distinct active set identification schemes (truncated cyclic, greedy, randomized and proximal gradient selection). Besides, the package provides the choices between convex (ℓ_1 norm) and nonvoncex (MCP and SCAD) regularizations. These methods provide a broad range of options of different sparsity inducing regularizations for most commonly used regression approaches, and various schemes of active set identification allow for the trade-off between statistical consistency and computational efficiency. Moreover, `picasso` has a provable linear convergence to a unique sparse local optimum with optimal statistical properties, which the competing packages (e.g., `ncvreg`) do not have. The package is coded in `C` and can scale up to large problems efficiently with the memory optimized via the sparse matrix output.

1 Introduction

Let $\boldsymbol{\theta}^* = (\theta_1^*, ..., \theta_d^*)^T \in \Omega^*$ be a parameter vector to be estimated, where $\Omega^* \in \mathbb{R}^d$. We are interested in solving a class of regularized sparse learning problems in a generic form:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \quad \mathcal{L}(\boldsymbol{\theta}) + \mathcal{R}_\lambda(\boldsymbol{\theta}), \tag{1}$$

where $\mathcal{L}(\boldsymbol{\theta})$ is the loss function, $\mathcal{R}_\lambda(\boldsymbol{\theta})$ is the regularization term with a regularization parameter λ. The pathwise coordinate optimization combined with the active set identification, also called "active shooting algorithm" (Peng et al., 2009), is one the of the most widely applied solvers for a large variety of sparse learning problems (1) by virtue of its algorithmic simplicity and favorable property of taking advantage of the model sparsity (Friedman et al., 2007; Breheny and Huang, 2011; Shalev-Shwartz and Tewari, 2011). Recent research also justifies the computational and statistical superiority of the empirical performance of the pathwise coordinate descent procedure for a large family of regularized M-estimators, including both convex and nonvoncex regularizations (Zhao and Liu, 2015), which makes it a more attractive algorithm in practice.

*Department of Electrical and Computer Engineering, University of Minnesota Twin Cities;
†Department of Computer Science, Johns Hopkins University;
‡Department of Statistics, Rutgers University;
§Department of Operations Research and Financial Engineering, Princeton University.

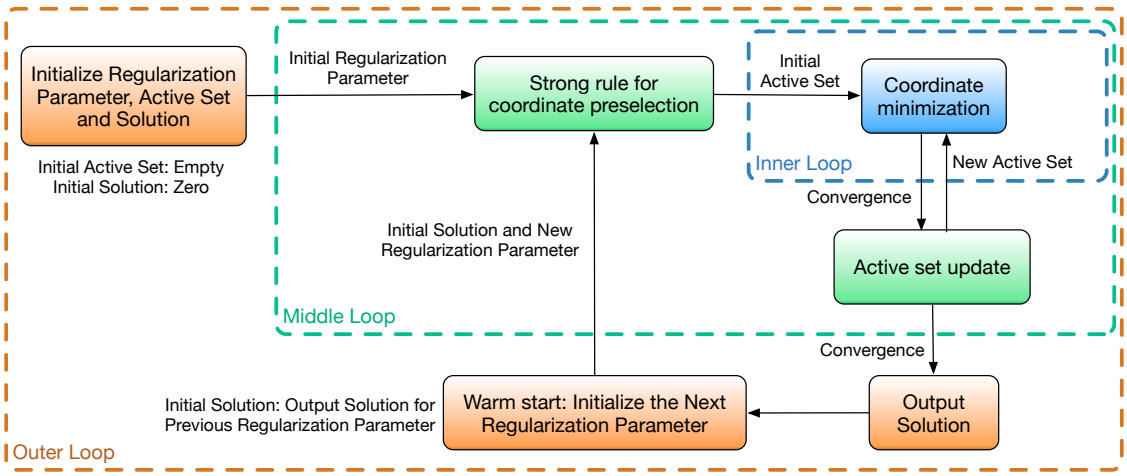


Figure 1: The pathwise coordinate optimization framework with 3 nested loops : (1) Warm start initialization; (2) Active set updating and strong rule for coordinate preselection; (3) Active coordinate minimization.

In this paper, we introduce and describe an `R` package called `picasso`, where a unified framework of pathwise coordinate optimization is implemented for a large class of regularized sparse learning problems with both convex and nonvoncex regularization options. Different types of active set identification schemes are also provided such that users have the freedom to trade off between better statistical performance and computational preference. More specifically, the sparse learning problems we implement include Sparse Linear Regression, Sparse Logistic Regression and Sparse Column Inverse Operator (Tibshirani, 1996; Banerjee et al., 2008; Liu and Luo, 2012), each with ℓ_1 norm, MCP and SCAD regularizations (Fan and Li, 2001; Zhang, 2010). Integrated with four different active identification methods for all approaches described above, including truncated cyclic, greedy, randomized and proximal gradient selection, we provide a wide class of combinations for sparse learning problems based on pathwise coordinate optimization. Unlike existing packages for nonvoncex regularized M-estimation, such as `ncvreg`, the algorithms in `picasso` have provable linear convergence to a unique sparse local optimum with optimal statistical properties (e.g. minimax optimality and oracle properties in Zhao et al. (2014)).

2 Algorithm Design and Implementation

The design of `picasso` is based on the recent development in the generic pathwise coordinate optimization algorithm (Zhao et al., 2014). It integrates the warm start initialization, active set updating strategy, and strong rule for coordinate preselection into the classical coordinate optimization. The overall algorithm contains three structurally nested loops as shown in Figure 1:

(1) Outer loop: The warm start initialization, also referred to as the pathwise optimization scheme, is adopted to optimize the objective function in a multistage way with a sequence of decreasing regularization parameters corresponding to solutions from sparse to dense. For each stage, the algorithm initializes the estimator using the solution from the previous stage.

(2) Middle loop: The algorithm divides all coordinates into active ones (active set) and inactive

ones (inactive set) based on the strong rule (Tibshirani et al., 2012) for coordinate gradient thresholding. Besides, when the inner loop terminates, the algorithm then exploits one of the active set update rules, including truncated cyclic, greedy, randomized and proximal gradient selection, to identify a new active set, which further decreases the objective value and repeats the inner loops. The middle loop terminates when the active set no longer changes.

(3) Inner loop: The coordinate optimization is conducted only on the current active coordinates until convergence, with all inactive coordinates remaining zero. Active coordinates are updated efficiently with the "naive update" that only operates on the non-zero coefficients. Further efficiencies are achieved using the "covariance update" for sparse linear regression (Friedman et al., 2010). The inner loop terminates when the difference of estimates in successive loops is within a predefined numerical precision.

In practice, the warm start initialization, active set updating strategies, and strong rule for coordinate preselection significantly boost the computational performance, making pathwise coordinate optimization one of the most important computational frameworks for solving the sparse learning problems. The package is implemented in `C` with the memory optimized via the sparse matrix output, and called from `R` by a user-friendly interface. The numerical evaluations show that `picasso` is efficient and scales up to large problems.

3 Examples of User Interface

We illustrate the user interface of `picasso` by the following examples on sparse linear regression, sparse logistic regression and sparse undirected graphical model estimation.

3.1 Sparse Linear Regression

Before we proceed with the example, we first introduce some background knowledge. Let $\boldsymbol{\theta} \in \mathbb{R}^d$ denote the regression coefficient vector, $\mathbf{X} \in \mathbb{R}^{d \times n}$ denote the design matrix, and $\boldsymbol{y} \in \mathbb{R}^n$ denote the response vector. We solve the following regularized minimization problem

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{1}{2n} ||\boldsymbol{y} - \mathbf{X}\boldsymbol{\theta}||_2^2 + \mathcal{R}_\lambda(\boldsymbol{\theta}), \tag{2}$$

where $\mathcal{R}_\lambda(\boldsymbol{\theta}) = \sum_{j=1}^d r_\lambda(\theta_j)$ is the regularization function. There are three options for $\mathcal{R}_\lambda(\boldsymbol{\theta})$, including:

(1) The ℓ_1 regularization, where

$$r_\lambda(\theta_j) = |\theta_j|;$$

(2) The SCAD regularization, where

$$r_\lambda(\theta_j) = \begin{cases} \theta_j, & \text{if } |\theta_j| \leq \lambda, \\ \frac{\gamma\lambda\theta_j - 0.5(\theta_j^2 + \lambda^2)}{\lambda(\gamma - 1)}, & \text{if } \lambda < |\theta_j| \leq \gamma\lambda, \\ \frac{\lambda(\gamma^2 - 1)}{2(\gamma - 1)}, & \text{if } |\theta_j| > \gamma\lambda; \end{cases}$$

(3) The MCP regularization, where

$$r_\lambda(\theta_j) = \begin{cases} \theta_j - \frac{\theta_j^2}{2\lambda\gamma}, & \text{if } |\theta_j| \leq \gamma\lambda, \\ \frac{\gamma\lambda}{2}, & \text{if } |\theta_j| > \gamma\lambda. \end{cases}$$

The illustrative examples of the above three regularization functions are provided in Figure 2. The ℓ_1 regularization is convex and computationally tractable, but introduces large estimation bias. The nonconvex SCAD and MCP regularizations reduce the estimation bias, but are more computationally challenging. See more technical details in Zhang (2010); Fan and Li (2001).

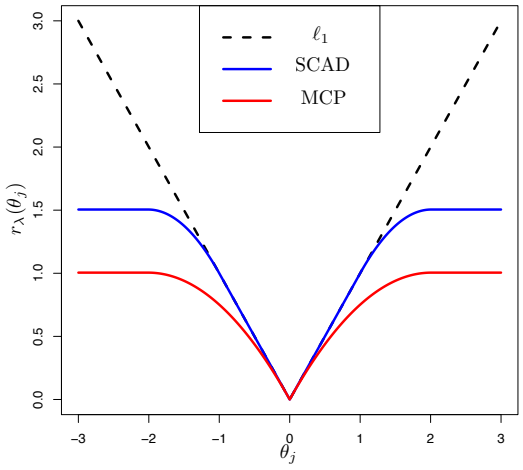


Figure 2: The illustrative examples of the ℓ_1, SCAD, and MCP regularizations. We choose $\lambda = 1$ and $\beta = 2.01$ for both SCAD and MCP.

Existing software packages (e.g., `ncvreg`) adopt a heuristic pathwise coordinate descent algorithms to solve (2). But there is no theoretical guarantee on the obtained estimators due to the nonconvexity. In contrast, `picasso` adopt the pathwise calibrated sparse shooting algorithm (PICASSO), which guarantees linear convergence to a sparse local optimum with good statistical properties. See more technical details in Zhao et al. (2014).

We then proceed with a concrete example of using `picasso` to solve sparse linear regression problems using different regularization functions.

```
> # Load the library
> library(picasso)
> # parameter settings
> n = 200                 # data size
> d = 2000                # data dimension
> cor.X = 0.5             # covariance between predictors in a random resign
> S0 = matrix(cor.X,d,d) + (1-cor.X)*diag(d) # covariance matrix of random design
> R = chol(S0)
> X = scale(matrix(rnorm(n*d),n,d)%*%R*sqrt(n-1)*sqrt(n))/sqrt(n-1)*sqrt(n)
>     # generate the design matrix satisfying the column normalization condition
> w = c(2,0,3,-2, rep(0,d-4))    # generate regression coefficient vector
> Y = X%*%w + rnorm(n)           # response vector
```

Using the above R script, we load the picasso package, and generate a simulated dataset with 200 samples and 2000 variables. The true coefficient vector is $\boldsymbol{\theta}^* = [2, 0, 3, -2, 0, ..., 0]^T$, and the random noise is sampled from a standard n-dimensional normal distribution $N(0, \mathbf{I})$.

```
> nlambda = 20           # number of regularization parameter
> lambda.ratio = 0.02    # minimum ratio of regularization paramter
> # fitting the linear model with Lasso
> out.l1 = picasso(X, Y, nlambda = nlambda,lambda.min.ratio = lambda.ratio,
+                  family="gaussian", method = "l1", alg="greedy", opt="naive",
+                  max.act.in=3)
> # fitting the linear model with MCP
> out.mcp = picasso(X, Y, nlambda = nlambda,lambda.min.ratio = lambda.ratio,
+                   family="gaussian", method = "mcp", alg="proximal", opt="cov",
+                   df=100, max.act.in=3)
> # fitting the linear model with SCAD
> out.scad = picasso(X, Y, nlambda = nlambda,lambda.min.ratio = lambda.ratio,
+                    method = "scad")
> # plot the regularization path
> par(mfrow=c(1,3))
> plot(out.l1)
> plot(out.mcp)
> plot(out.scad)
```

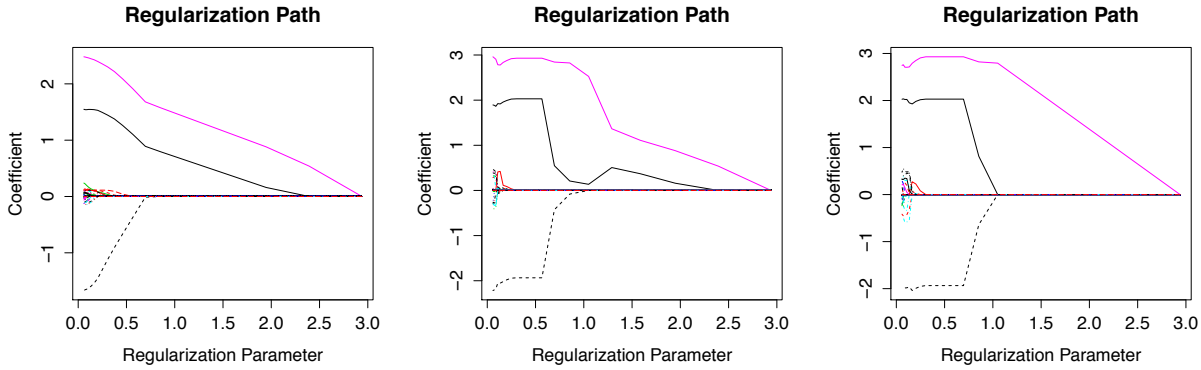


Figure 3: Regularization Path for sparse linear regression. The regularization functions are ℓ_1, SCAD, and MCP respectively from left to right.

Using the above R script, we fit sparse linear models using the ℓ_1, MCP, and SCAD regularizations. This yields three solution paths corresponding to a sequence of 20 regularization parameters (in Figure 3). For Lasso, wo choose the greedy search scheme for updating the active set in the middle loop. We choose max.act.in=3, which allows at most three coordinates to be updated each time in the active set update, to further reduce the iterations of the middle loop. For MCP regularization, we choose the proximal gradient search scheme for updating the active set and the option of "covariance update" for the sparse update in the inner loop with the maximal degree of

freedom (nonzero coefficients in the solution) to be 100. For SCAD regularization, we use default values for unspecified options.

3.2 Sparse Logistic Regression

Before we proceed with the example, we first introduce some background knowledge. Let $\boldsymbol{\theta} \in \mathbb{R}^d$ denote the regression coefficient vector, $\mathbf{X} \in \mathbb{R}^{d \times n}$ denote the design matrix, and $\boldsymbol{y} \in \mathbb{R}^n$ denote the response vector. We solve the following regularized minimization problem

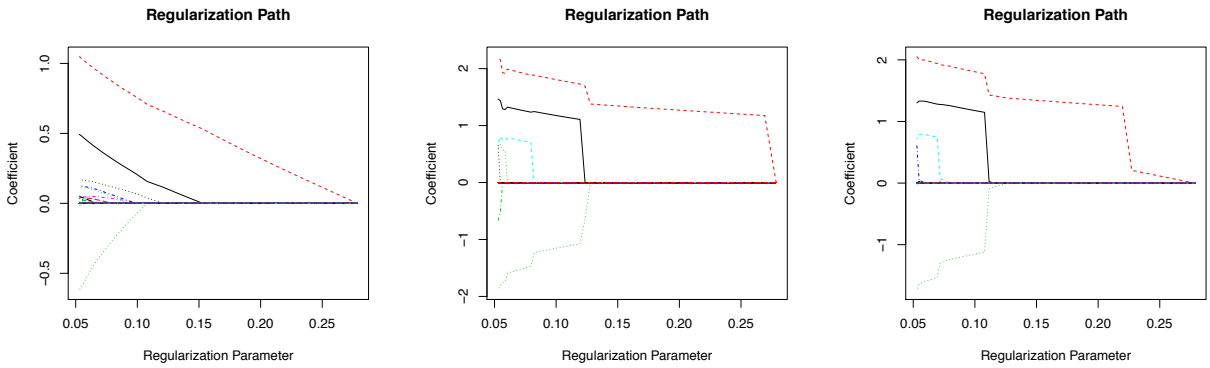
$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(\log \left[1 + \exp(\mathbf{X}_{i*}^T \boldsymbol{\theta}) \right] - y_i \mathbf{X}_{i*}^T \boldsymbol{\theta} \right) + \mathcal{R}_\lambda(\boldsymbol{\theta}), \tag{3}$$

where $\mathbf{X}_{i*} = [\mathbf{X}_{i1}, ...\mathbf{X}_{id}]^T$ denotes the i-th row of $\mathbf{X}$.

We then proceed with a concrete example of using `picasso` to solve sparse linear regression problem. The generation of simulated dataset is identical to the code for sparse linear regression, except that the observation model Y is generated as

```
> p = exp(X%*%w)/(1+exp(X%*%w))
> Y = rbinom(n,rep(1,n),p)
```

The model fitting for sparse logistic regression is identical to the code for sparse linear regression, except that we choose `family = "binomial"`. We provide the solution paths for sparse logistic regression in Figure 4.



(a) ℓ_1 (b) MCP (c) SCAD

Figure 4: Regularization Path for sparse linear regression. The regularization functions are ℓ_1, SCAD, and MCP respectively from left to right.

3.3 Sparse Column Inverse Operator

Before we proceed with the example, we first introduce some background knowledge. Let $\boldsymbol{X} \in \mathbb{R}^d$ be a random vector, which follows a d-variate Gaussian distribution with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$. The graph structure of the Gaussian distribution is encoded by the sparsity pattern of the inverse covariance matrix, also called precision matrix, $\boldsymbol{\Theta} = \boldsymbol{\Sigma}^{-1}$. Suppose we have n i.i.d. random

sample $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(n)}\}$ from the distribution of $\boldsymbol{X}$. Our goal is to recovery the underlying graph structure by estimating $\boldsymbol{\Theta}$. In `picasso`, we adopt the Sparse Column Inverse Operator (SCIO) Liu and Luo (2012) for this task. More specifically, we solve the following collection of regularized minimization problems

$$\min_{\boldsymbol{\Theta}_{*j} \in \mathbb{R}^d} \frac{1}{2} \boldsymbol{\Theta}_{*j}^T \mathbf{S} \boldsymbol{\Theta}_j - \mathbf{I}_{*j}^T \boldsymbol{\Theta}_{*j} + \lambda ||\boldsymbol{\Theta}_{*j}||_1 \quad \text{for all } j = 1, 2, \ldots, d, \tag{4}$$

where $\boldsymbol{\Theta}_{*j}$ and $\mathbf{I}_{*j}$ denote the j-th columns of $\boldsymbol{\Theta}$ and $\mathbf{I}$ respectively.

We then proceed with a concrete example of using `picasso` to estimate the undirected graphical model using different procedures.

```
> # Load the library
> library(picasso)
> # parameter settings
> n = 200                # data size
> d = 100                # data dimension
> # generate a chain graph
> D = scio.generator(n=n,d=d,graph="band",seed=seed,g=1)
> plot(D)                # plot the generated data
```

Using the above `R` script, we generates a simulated dataset with 200 samples and 100 variables (the number of parameters to be estimated is $100 * (100 - 1)/2 = 4950$). The true undirected graphical model has a chain structure, with the $(i, i+1)$, $(i+1, i)$-th entries are non-zero and all the other off diagonal entries of the precision matrix are zero. This results in the sparsity level as $99/4950 = 0.02$. We provide the plot of the adjacency matrix, (empirical) covariance matrix and the undirected graph pattern in Figure 5 for better visual illustration.

```
> data = D$data                  # extract the data
> nlambda = 50                   # number of regularization parameter
> lmin.ratio = 0.2               # minimum ratio of regularization paramter
> family="graph"                 # family parameter for graphical model
> # fitting the graphical model with SCIO
> out.scio = picasso(data, nlambda = nlambda, lambda.min.ratio = lmin.ratio,
+                    family=family)
> # plot the sparsity level information and 3 typical sparse graphs
> plot(out.scio)
```

Using the above `R` script, we estimate the undirected graphical model via SCIO. This yields a solution path corresponding to sequences of 50 regularization parameters. We provide the plots of sparsity level curves and 3 typical sparse graphs from the paths for all methods in Figure 6.

4 Numerical Simulation

We compare `picasso` with `R` package `ncvreg`, the most popular one for non-convex regularized sparse regression, to demonstrate the superior efficiency of our package. Another popular package `sparseset` is not considered here, since `sparsenet` implements a different coordinate minimization

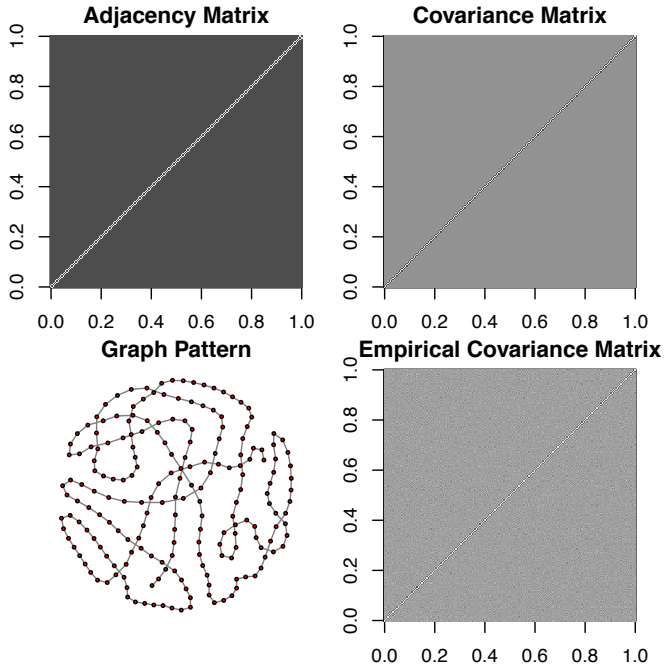


Figure 5: Plot of data generation from multivariate normal distributions with `"chain"` structure, including adjacency matrix, (empirical) covariance matrix, and the graph pattern.

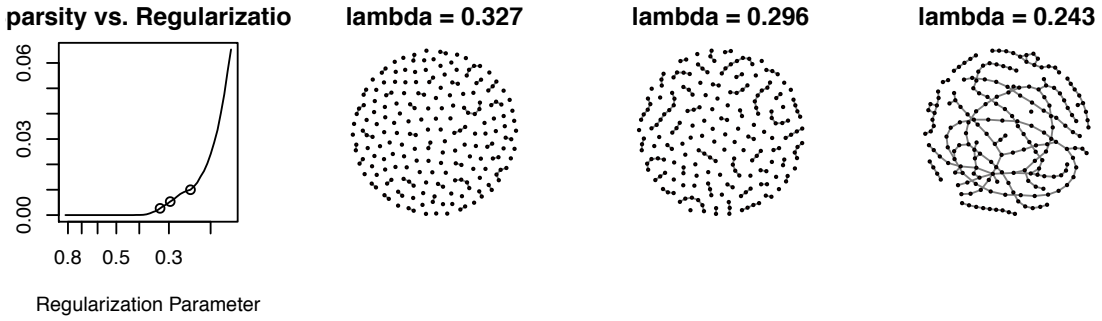


Figure 6: Plot sparsity level information and 3 typical sparse graphs from the graph path using SCIO.

algorithm based on calibrated MCP (Mazumder et al., 2011). All experiments are evaluated on a PC with Intel Core i5 3.2GHz processor. Timings of the CPU execution are recored in seconds and averaged over 100 replications on a sequence of 50 regularization parameters with approximately the same estimation errors. The convergence threshold are chosen to be 10^{-5} for all experiments. For `picasso`, we choose the greedy search scheme and set `max.act.in=5` throughout all experiments.

We first compare the timing performance and the statistical performance for sparse linear regression under well-conditioned scenarios. We choose the (n, d) pairs as $(500, 5000)$ and $(1000, 10000)$ respectively, where n is the number of observation in the response vector $\mathbf{Y} \in \mathbb{R}^n$ and d is the

Table 1: Average timing performance (in seconds) and optimal estimation errors with standard errors in the parentheses on sparse linear regression.

Sparse Linear Regression (Well-Conditioned)					
Method	Package	$n = 500, d = 5000$		$n = 1000, d = 10000$	
		Time	Est. Err.	Time	Est. Err.
ℓ_1 norm	`picasso`	0.5013(0.1404)	0.3924(0.0662)	1.4040(0.2358)	0.2677(0.0346)
	`ncvreg`	42.521(7.7725)	0.3924(0.0667)	138.44(24.122)	0.2670(0.0345)
MCP	`picasso`	0.4957(0.1809)	0.0773(0.0499)	1.3815(0.2018)	0.0586(0.0306)
	`ncvreg`	22.290(2.7846)	0.0775(0.0499)	94.746(18.329)	0.0592(0.0308)
SCAD	`picasso`	0.4942(0.0875)	0.0766(0.0505)	1.4384(0.1883)	0.0587(0.0306)
	`ncvreg`	38.476(7.0584)	0.0769(0.0505)	139.59(25.226)	0.0591(0.0309)
Sparse Linear Regression (Ill-Conditioned)					
Method	Package	$n = 50, d = 5000$		$n = 50, d = 10000$	
		Time	Est. Err.	Time	Est. Err.
MCP	`picasso`	0.1480(0.0098)	0.4629(0.2840)	0.2181(0.0310)	0.4904(0.3232)
	`ncvreg`	0.0908(0.0053)	1.5069(0.9596)	0.1646(0.0087)	1.7827(0.8856)

dimension of the parameter vector $\boldsymbol{\beta} \in \mathbb{R}^d$. We also set `opt="naive"`. For the design matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, we generate each row independently from a d-dimensional normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}_{ij} = 0.5$ for $i \neq j$ and $\Sigma_{ii} = 1$. Then we have $\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, where $\boldsymbol{\beta}$ has all 0 entries except $\boldsymbol{\beta}_{150} = 2$, $\boldsymbol{\beta}_{380} = 3$, $\boldsymbol{\beta}_{690} = -1.5$ and $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ has independent $\mathcal{N}(0, 1)$ entries. From the summary in Table 1, we see that while achieving almost identical optimal estimation errors $\|\boldsymbol{\beta} - \widehat{\boldsymbol{\beta}}\|_2$, `picasso` uniformly outperforms `ncvreg` under all settings, where `picasso` is approximately $50 \sim 100$ times faster.

We then compare the timing performance and the statistical performance for sparse linear regression under ill-conditioned scenarios. We choose the (n, d) pairs as $(50, 5000)$ and $(50, 10000)$ respectively. The generations of $\mathbf{X}$, $\boldsymbol{\beta}$ and $\boldsymbol{\varepsilon}$ are identical to the settings above, except that $\boldsymbol{\Sigma}_{ij} = 0.75$ for $i \neq j$. Due to the choices that values of d are much larger than n, and a larger value is chosen for $\boldsymbol{\Sigma}_{ij}$ for $i \neq j$, the problems considered here are much more challenging than the problems in the well-conditioned scenarios. We see from Table 1 that though `picasso` is slightly slower than `ncvreg`, its statistical performance is much better than `ncvreg`.

We also compare the timing performance for sparse logistic regression. The choices of (n, d) pairs are $(500, 2000)$, $(1000, 2000)$, $(500, 5000)$ and $(1000, 5000)$. The generations of $\mathbf{X}$ and $\boldsymbol{\beta}$ follow from the settings for sparse linear regression under well-conditioned scenarios. Then the response vector $\mathbf{Y}$ has independent Bernoulli$\left(\frac{exp(\mathbf{X}_{i*}\boldsymbol{\beta})}{1+exp(\mathbf{X}_{i*}\boldsymbol{\beta})}\right)$ entries. We see from Table 2 that `picasso` outperforms `ncvreg` under all settings, and scales better for increasing values of n and d.

We want to make a final comment that further speedups may be achieved for sparse linear regression with less correlated settings of the design matrix. For example, when the rows of $\mathbf{X}$ are generated independently from some multivariate normal distribution with $\boldsymbol{\Sigma}_{ij} = a^{|i-j|}$ for some constant $a \in (0, 1)$, then we may achieve > 100 times of acceleration than `ncvreg` by setting `opt="cov"` and `df` to be small compared with $\min\{n, d\}$.

Table 2: Average timing performance (in seconds) with standard errors in the parentheses on sparse logistic regression.

Sparse Logistic Regression					
Method	Package	$d = 2000$		$d = 5000$	
		$n = 500$	$n = 1000$	$n = 500$	$n = 1000$
ℓ_1 norm	`picasso`	0.2127(0.0089)	0.3918(0.0252)	0.4583(0.0321)	0.8054(0.0246)
	`ncvreg`	1.2464(0.7255)	5.7377(1.5040)	2.2527(0.7114)	10.096(2.4513)
MCP	`picasso`	0.3820(0.0892)	0.4860(0.0282)	0.6197(0.0543)	0.9942(0.0710)
	`ncvreg`	0.6639(0.4253)	2.5244(0.9032)	0.8451(0.2590)	2.8319(0.8218)
SCAD	`picasso`	0.3383(0.0553)	0.4995(0.0575)	0.6188(0.0555)	0.9323(0.0711)
	`ncvreg`	0.7226(0.1639)	3.9026(0.9745)	1.5180(0.5561)	7.1200(0.8744)

5 Discussion and Conclusion

In general, the `picasso` package demonstrates significantly improved computational efficiency and statistical consistency than competing packages (e.g., `ncvreg`) on non-convex regularized sparse regression problems. Especially, `picasso` provides much broader choices of regression and regularization families, which guarantees linear convergence to a unique sparse local optimum with optimal statistical properties. Further options for trading off statistical consistency and computational efficiency are provided using different active set identification schemes. Overall, the `picasso` package has the potential to serve as a powerful toolbox for high dimensional non-convex sparse learning problems. We will continue to maintain and support this package.

References

Banerjee, O., El Ghaoui, L. and d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data. *The Journal of Machine Learning Research* **9** 485–516.

Breheny, P. and Huang, J. (2011). Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The Annals of Applied Statistics* **5** 232–253.

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

Friedman, J., Hastie, T., Höfling, H. and Tibshirani, R. (2007). Pathwise coordinate optimization. *The Annals of Applied Statistics* **1** 302–332.

Friedman, J., Hastie, T. and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software* **33** 1–13.

Liu, W. and Luo, X. (2012). High-dimensional sparse precision matrix estimation via sparse column inverse operator. *arXiv preprint arXiv:1203.3896* .

Mazumder, R., Friedman, J. and Hastie, T. (2011). Sparsenet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association* **106**.

Peng, J., Wang, P., Zhou, N. and Zhu, J. (2009). Partial correlation estimation by joint sparse regression models. *Journal of the American Statistical Association* **104** 735–746.

Shalev-Shwartz, S. and Tewari, A. (2011). Stochastic methods for ℓ_1-regularized loss minimization. *The Journal of Machine Learning Research* **12** 1865–1892.

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

Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J. and Tibshirani, R. (2012). Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **74** 245–266.

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

Zhao, T. and Liu, H. (2015). Pathwise calibrated active shooting algorithm with application to semiparametric graph estimation. *Journal of Computational and Graphical Statistics* .

Zhao, T., Liu, H. and Zhang, T. (2014). A general theory of pathwise coordinate optimization. *arXiv preprint arXiv:1412.7477* .