The picasso Package for High Dimensional Regularized Sparse Learning in R

X. Li, J. Ge, T. Zhang, M. Wang, H. Liu, and T. Zhao*

Abstract

We introduce 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 Poisson Regression), combined with efficient active set selection strategies. Besides, the package allows users to choose different sparsity-inducing regularizers, including the convex ℓ_1, nonvoncex MCP and SCAD regularizers. The package is coded in C and can scale up to large problems efficiently with the memory optimized using sparse matrix output.

1 Introduction

The pathwise coordinate optimization combined is undoubtedly one the of the most popular solvers for a large variety of sparse learning problems. It takes advantage of the solution sparsity through a simple but elegant algorithmic structure, and significantly boosts the computational performance in practice(Friedman et al., 2007). Some recent advances in (Zhao et al., 2014; Ge et al., 2016) establishes theoretical guarantees to further justify its computational and statistical superiority for both convex and nonvoncex sparse learning, which makes it even more attractive to practitioners.

Here we introduce an R package called picasso, which implements a unified toolkit of pathwise coordinate optimization for a large class of convex and nonconvex regularized sparse learning approaches. Efficient active set selection strategies are provided to guarantee superior statistical and computational preference. Specifically, we implement sparse linear regression, sparse logistic regression, and sparse Poisson regression (Tibshirani, 1996). The options for regularizers include the ℓ_1, MCP, and SCAD regularizers (Fan and Li, 2001; Zhang, 2010). Unlike existing

*Xingguo Li is affiliated with Department of Electrical and Computer Engineering at University of Minnesota; Tong Zhang is affiliated with Tencent AI Lab; Jason Ge, Mengdi Wang, and Han Liu are affiliated with Department of Operations Research and Financial Engineering at Princeton University; Tuo Zhao is affiliated with School of Industrial and Systems Engineering at Georgia Institute of Technology, Atlanta, GA 30332; Emails: lixx1661@umn.edu, jiange@princeton.edu, tourzhao@gatech.edu; Xingguo Li and Jason Ge contributed equally; Tuo Zhao is the corresponding author.

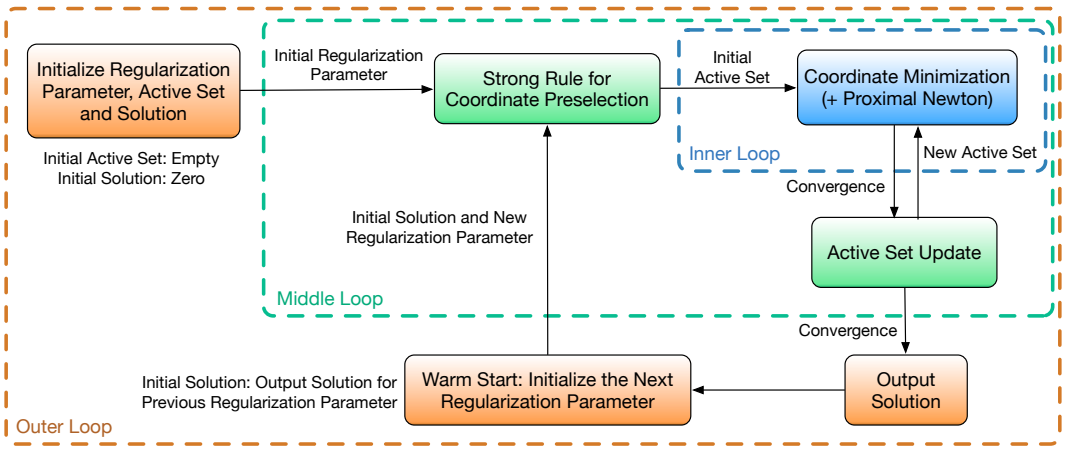


Figure 1: The pathwise coordinate optimization framework with 3 nested loops : (1) Warm start initialization; (2) Active set updating and strong rule for coordinate preselection; (3) Active coordinate minimization.

packages implementing heuristic optimization algorithms such as `ncvreg`, our implemented algorithm `picasso` have strong theoretical guarantees that it attains a global linear convergence to a unique sparse local optimum with optimal statistical properties (e.g. minimax optimality and oracle properties). See more technical details in Zhao et al. (2014); Ge et al. (2016).

2 Algorithm Design and Implementation

The algorithm implemented in `picasso` is mostly based on the generic pathwise coordinate optimization framework proposed by Zhao et al. (2014); Ge et al. (2016), which integrates the warm start initialization, active set updating strategy, and strong rule for coordinate preselection into the classical coordinate optimization. The 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 applied to minimize the objective function in a multistage manner using a sequence of decreasing regularization parameters, which yields a sequence of solutions from sparse to dense. At each stage, the algorithm uses the solution from the previous stage as initialization.

(2) Middle loop: The algorithm first divides all coordinates into active ones (active set) and inactive ones (inactive set) by a so-called strong rule based on coordinate gradient thresholding (Tibshirani et al., 2012). Then the algorithm calls an inner loop to optimize the objective, and update the active set based on efficient active set updating strategies. Such a routine is repeated until the active set no longer changes

(3) Inner loop: The algorithm conducts coordinate optimization (for sparse linear regression) or proximal Newton optimization combined with coordinate optimization (for sparse logistic regression and Possion regression) only over active coordinates until convergence, with all inactive coordinates staying zero values. The active coordinates are updated efficiently using an efficient "naive update" rule that only operates on the non-zero coefficients. Further efficiencies are achieved using the "covariance update" rule. See more details in (Friedman et al., 2010). The inner loop terminates when the successive descent is within a predefined numerical precision.

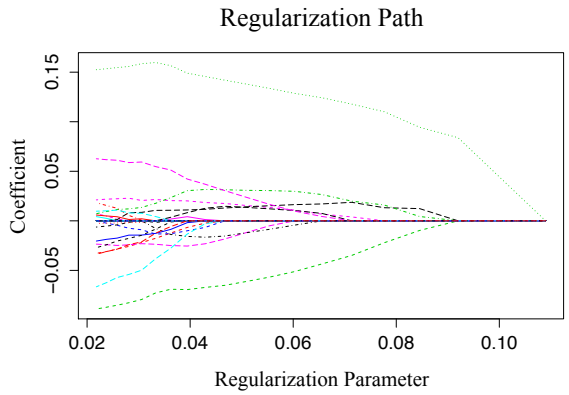
The warm start initialization, active set updating strategies, and strong rule for coordinate preselection significantly boost the computational performance, making pathwise coordinate optimization one of the most important computational frameworks for sparse learning. The package is implemented in `C`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

3 Examples of User Interface

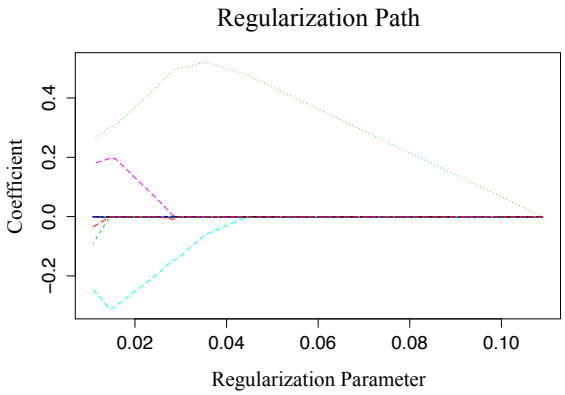
We illustrate the user interface by analyzing the eye disease data set in `picasso`.

```
> # Load the data set
> library(picasso); data(eyedata)
> # Lasso
> out1 = picasso(x,y,method="l1",opt="naive",nlambda=20,
+                lambda.min.ratio=0.2)
> # MCP regularization
> out2 = picasso(x,y,method="mcp", gamma = 1.25, prec=1e-4)
> # Plot solution paths
> plot(out1); plot(out2)
```

The program automatically generates a sequence of regularization parameters and estimate the corresponding solution paths based on the ℓ_1 and MCP regularizers respectively. For the ℓ_1 regularizer, the number of regularization parameters as 20, and the minimum regularization parameter as `0.2*lambda.max`. Here `lambda.max` is the smallest regularization parameter yielding an all zero solution (automatically calculated by the package). For the MCP regularizer, we set the concavity parameter as $\gamma = 1.25$, and the pre-defined accuracy as 10^{-4}. Here `nlambda` and `lambda.min.ratio` are omitted, and therefore set by the default values (`nlambda=100` and `lambda.min.ratio=0.01`). We further plot two solution paths in Figure 3.



(a) ℓ_1



(b) MCP

4 Numerical Simulation

To demonstrate the superior efficiency of our package, we compare `picasso` with a popular `R` package `ncvreg` for nonconvex regularized sparse regression, and with the most popular `R` package `glmnet` for convex regularized sparse regression. 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 100 regularization parameters with approximately the same estimation errors and sparsities. The convergence threshold are chosen to be 10^{-7} for 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 $y \in \mathbb{R}^n$ and d is the dimension of the parameter vector $\theta \in \mathbb{R}^d$. We also set `opt="naive"`. For the design matrix $X \in \mathbb{R}^{n \times d}$, we generate each row independently from a d-dimensional normal distribution $N(\mathbf{0}, \Sigma)$, where $\Sigma_{ij} = 0.5$ for $i \neq j$ and $\Sigma_{ii} = 1$. Then we have $y = X\theta + \varepsilon$, where θ has all 0 entries except randomly selected 1% entries have independent $N(0, 25)$ entries and $\varepsilon \in \mathbb{R}^n$ has independent $N(0, 1)$ entries. From the summary in Table 1, we see that while achieving almost identical optimal estimation errors $\|\theta - \widehat{\theta}\|_2$, `picasso` is as fast as `glmnet`, both of which uniformly outperform `ncvreg` under all settings with approximately $50 \sim 100$ times speedups.

We then compare the timing performance and the statistical performance for sparse linear regression under ill-conditioned scenarios. We choose the (n, d) pairs, the generations of X, θ and ε identical to the settings above, except that $\Sigma_{ij} = 0.75$ for $i \neq j$. Due to a larger value is chosen for Σ_{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 `picasso` not only outperforms `ncvreg` uniformly with approximately $50 \sim 100$ times speedups as in the well-conditioned setting, but also outperforms `glmnet` with approximately 2 times speedups for ℓ_1 norm regularized method. This is the most efficient `R` function for the sparse linear regression so far to the best of our knowledge.

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	0.2647(0.0092)	0.3454(0.0594)	1.0746(0.0715)	0.2549(0.0405)
	glmnet	0.2799(0.0105)	0.3481(0.0600)	1.0776(0.0638)	0.2563(0.0408)
	ncvreg	20.698(1.9521)	0.3479(0.0600)	97.540(7.0725)	0.2555(0.0405)
MCP	picasso	0.2546(0.0084)	0.0743(0.0376)	1.0085(0.0696)	0.0444(0.0235)
	ncvreg	5.4379(0.7299)	0.0747(0.0370)	45.262(2.5917)	0.0445(0.0222)
SCAD	picasso	0.2526(0.0088)	0.0786(0.0357)	1.0072(0.0654)	0.0471(0.0229)
	ncvreg	17.865(1.0678)	0.0880(0.0374)	89.713(9.8081)	0.0522(0.0255)
Sparse Linear Regression (Ill-Conditioned)					
ℓ_1 norm	picasso	0.2985(0.0205)	1.5168(0.2103)	1.1905(0.0374)	1.4641(0.0737)
	glmnet	0.5055(0.1411)	1.5152(0.1992)	2.2316(0.2811)	1.4737(0.0631)
	ncvreg	26.171(2.1503)	1.5048(0.1739)	85.140(5.9043)	1.4622(0.0520)
MCP	picasso	0.4614(0.0296)	0.5089(0.0770)	2.1639(0.0820)	0.5367(0.0742)
	ncvreg	20.897(2.8878)	0.5168(0.0819)	111.78(12.255)	0.5345(0.0731)
SCAD	picasso	0.4825(0.0622)	0.5117(0.0673)	2.4075(0.2736)	0.5362(0.0766)
	ncvreg	53.025(3.4142)	0.5070(0.0700)	193.63(15.992)	0.5359(0.0712)

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 X and θ follow from the settings for sparse linear regression under well-conditioned scenarios. Then the response vector y has independent Bernoulli$\left(\frac{\exp(X_{i*}^\top \theta)}{1+\exp(X_{i*}^\top \theta)}\right)$ entries. We see from Table 2 that picasso outperforms ncvreg under all settings, and scales better for increasing values of n and d.

Here we make a final comment that picasso performs stably for various choices of n, d and tuning parameters compared with ncvreg. Especially when the tuning parameters are relatively small (corresponding to denser estimators), ncvreg may converge very slow or fail to converge, which we did not show here. To avoid such scenario, we choose the sequence of tunning parameters under the criteria that ncvreg attains the optimal performance in terms of the parameter estimation, while the estimators are not too dense so that it fails to converge.

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

Sparse Logistic Regression (Timing Performance)					
Method	Package	$d = 2000$		$d = 5000$	
		$n = 500$	$n = 1000$	$n = 500$	$n = 1000$
ℓ_1 norm	picasso	0.3339(0.0466)	0.5705(0.0121)	0.6135(0.1195)	1.1130(0.0281)
	glmnet	0.2765(0.0212)	0.7116(0.0579)	0.4495(0.0318)	1.0922(0.1404)
	ncvreg	42.970(8.8731)	187.15(25.265)	75.181(12.831)	316.47(46.937)
MCP	picasso	0.8110(0.0268)	1.3986(0.0565)	1.6125(0.0678)	3.1815(0.2510)
	ncvreg	1.8135(0.7233)	7.5910(0.5854)	3.1759(0.4533)	9.3205(0.4815)
SCAD	picasso	0.8241(0.0141)	1.6325(0.0728)	1.9053(0.0749)	3.0137(0.3252)
	ncvreg	2.5635(0.1225)	11.558(1.5506)	4.9186(0.6390)	21.932(1.4928)
Sparse Logistic Regression (Estimation Error)					
ℓ_1 norm	picasso	0.7661(0.0842)	1.1498(0.1014)	1.8152(0.0942)	1.1930(0.0281)
	glmnet	0.7607(0.0757)	1.1461(0.1003)	1.8133(0.0991)	1.1886(0.0245)
	ncvreg	0.7591(0.0736)	1.1574(0.1051)	1.7914(0.1023)	1.1985(0.0204)
MCP	picasso	0.1062(0.0244)	0.2145(0.0249)	0.6469(0.0343)	0.1524(0.0060)
	ncvreg	0.0944(0.1335)	0.2120(0.0350)	0.6546(0.0596)	0.1487(0.0128)
SCAD	picasso	0.1173(0.0225)	0.2129(0.0104)	0.7062(0.1048)	0.1624(0.0060)
	ncvreg	0.1189(0.0182)	0.2154(0.0193)	0.7117(0.0957)	0.1516(0.0154)

5 Conclusion

The picasso package demonstrates significantly improved computational and statistical performance over existing packages such as ncvreg for nonconvex regularized sparse learning. Besides, picasso also shows improvement over the popular packages such as glmnet for sparse linear regression under ill-conditioned settings. Moreover, the algorithm implemented in picasso, which guarantees a global linear convergence to a unique sparse local optimum with optimal statistical properties. Overall, the picasso package has the potential to serve as a powerful toolbox for high dimensional sparse learning. We will continue to maintain and support this package.

References

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.

Ge, J., Wang, M., Liu, H., Hong, M. and Zhao, T. (2016). Homotopy active set proximal newton algorithm for sparse learning. Tech. rep., Georgia Tech.

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., Liu, H. and Zhang, T. (2014). Pathwise coordinate optimization for nonconvex sparse learning: Algorithm and theory. *arXiv preprint arXiv:1412.7477* .