A Unified Algorithm for the Non-Convex Penalized Estimation: The ncpen Package

by Dongshin Kim, Sangin Lee[*] *and Sunghoon Kwon*[†]

Abstract Various R packages have been developed for the non-convex penalized estimation but they can only be applied to the smoothly clipped absolute deviation (SCAD) or minimax concave penalty (MCP). We develop an R package, entitled **ncpen**, for the non-convex penalized estimation in order to make data analysts to experience other non-convex penalties. The package **ncpen** implements a unified algorithm based on the convex concave procedure and modified local quadratic approximation algorithm, which can be applied to a broader range of non-convex penalties, including the SCAD and MCP as special examples. Many user-friendly functionalities such as generalized information criteria, cross-validation and ridge regularization are provided also.

1 Introduction

The penalized estimation has been one of the most important statistical techniques for high dimensional data analysis, and many penalties have been developed such as the least absolute shrinkage and selection operator (LASSO) (Tibshirani, 1996), smoothly clipped absolute deviation (SCAD) (Fan and Li, 2001), and minimax concave penalty (MCP) (Zhang, 2010). In the context of R, many authors released fast and stable R packages for obtaining the whole solution path of the penalized estimator for the generalized linear model (GLM). For example, **lars** (Efron et al., 2004), **glmpath** (Park and Hastie, 2007) and **glmnet** (Friedman et al., 2007) implement the LASSO. Packages such as **plus** (Zhang, 2010), **sparsenet** (Mazumder et al., 2011), **cvplogit** (Jiang and Huang, 2014) and **ncvreg** (Breheny and Huang, 2011) implement the SCAD and MCP. Among them, **glmnet** and **ncvreg** are very fast, stable, and well-organized, presenting various user-friendly functionalities such as the cross-validation and ℓ_2-stabilization (Zou and Hastie, 2005; Huang et al., 2016b).

The non-convex penalized estimation has been studied by many researchers (Fan and Li, 2001; Kim et al., 2008; Huang et al., 2008; Zou and Li, 2008; Zhang and Zhang, 2012; Kwon and Kim, 2012; Friedman, 2012). However, there is still a lack in research on the algorithms that exactly implement the non-convex penalized estimators for the non-convexity of the objective function. One nice approach is using the coordinate descent (CD) algorithm (Tseng, 2001; Breheny and Huang, 2011). The CD algorithm fits quite well for some quadratic non-convex penalties such as the SCAD and MC (Mazumder et al., 2011; Breheny and Huang, 2011; Jiang and Huang, 2014) since each coordinate update in the CD algorithm becomes an easy convex optimization problem with a closed form solution. This is the main reason for the preference of the CD algorithm implemented in many R packages such as **sparsenet** and **ncvreg**. However, coordinate updates in the CD algorithm require extra univariate optimizations for other non-convex penalties such as the log and bridge penalties (Zou and Li, 2008; Huang et al., 2008; Friedman, 2012), which severely lowers the convergence speed. Another subtle point is that the CD algorithm requires standardization of the input variables and need to enlarge the concave scale parameter in the penalty (Breheny and Huang, 2011) to obtain the local convergence, which may cause to lose an advantage of non-convex penalized estimation (Kim and Kwon, 2012) and give much different variable selection performance (Lee, 2015).

In this paper, we develop an R package **ncpen** for the non-convex penalized estimation based on the convex-concave procedure (CCCP) or difference-convex (DC) algorithm (Kim et al., 2008; Shen et al., 2012) and the modified local quadratic approximation algorithm (MLQA) (Lee et al., 2016). The main contribution of the package **ncpen** is that it encompasses most of existing non-convex penalties, including the truncated ℓ_1 (Shen et al., 2013), log (Zou and Li, 2008; Friedman, 2012), bridge (Huang et al., 2008), moderately clipped LASSO (Kwon et al., 2015), sparse ridge (Huang et al., 2016a; Kwon et al., 2013) penalties as well as the SCAD and MCP and covers a broader range of regression models: multinomial and Cox models as well as the GLM. Further, **ncpen** provides two unique options: the investigation of initial dependent solution paths and non-standardization of input variables, which allow the users more flexibility.

The rest of the paper is organized as follows. In the next section, we describe the algorithm implemented in **ncpen** with major steps and details. Afterwards, the main functions and various options in **ncpen** are presented with numerical illustrations. Finally, the paper concludes with remarks.

[*] Co-first author: D. Kim and S. Lee equally contributed to this work

[†] Corresponding author

2 An algorithm for the non-convex penalized estimation

We consider the problem of minimizing

$$Q_\lambda(\boldsymbol{\beta}) = L(\boldsymbol{\beta}) + \sum_{j=1}^{p} J_\lambda\left(|\beta_j|\right), \tag{1}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ is a p-dimensional parameter vector of interest, L is a convex loss function and J_λ is a non-convex penalty with tuning parameter $\lambda > 0$. We first introduce the CCCP-MLQA algorithm for minimizing Q_λ when λ is fixed, and then explain how to construct the whole solution path over a decreasing sequence of λs by using the algorithm.

A class of non-convex penalties

We consider a class of non-convex penalties that satisfy $J_\lambda(|t|) = \int_0^{|t|} \nabla J_\lambda(s)\, ds, t \in \mathbb{R}$ for some non-decreasing function ∇J_λ and

$$D_\lambda(t) = J_\lambda(|t|) - \kappa_\lambda |t| \tag{2}$$

is concave function, where $\kappa_\lambda = \lim_{t \to 0+} \nabla J_\lambda(t)$. The class includes most of existing non-convex penalties: SCAD (Fan and Li, 2001),

$$\nabla J_\lambda(t) = \lambda I[0 < t < \lambda] + \{(\tau\lambda - t)/(\tau - 1)\} I[\lambda \le t < \tau\lambda]$$

for $\tau > 2$, MCP (Zhang, 2010),

$$\nabla J_\lambda(t) = (\lambda - t/\tau) I[0 < t < \tau\lambda]$$

for $\tau > 1$, truncated ℓ_1-penalty (Shen et al., 2013),

$$\nabla J_\lambda(t) = \lambda I[0 < t < \tau]$$

for $\tau > 0$, moderately clipped LASSO (Kwon et al., 2015),

$$\nabla J_\lambda(t) = (\lambda - t/\tau)[0 < t < \tau(\lambda - \gamma)] + \gamma[t \ge \tau(\lambda - \gamma)]$$

for $\tau > 1$ and $0 \le \gamma \le \lambda$, sparse ridge (Kwon et al., 2013),

$$\nabla J_\lambda(t) = (\lambda - t/\tau) I[0 < t < \tau\lambda/(\tau\gamma + 1)] + \gamma t I[t \ge \tau\lambda/(\tau\gamma + 1)]$$

for $\tau > 1$ and $\gamma \ge 0$, modified log (Zou and Hastie, 2005).

$$\nabla J_\lambda(t) = (\lambda/\tau)[0 < t < \tau] + (\lambda/t)[t \ge \tau]$$

for $\tau > 0$, and modified bridge (Huang et al., 2008)

$$\nabla J_\lambda(t) = \left(\lambda/2\sqrt{\tau}\right)[0 < t < \tau] + (\lambda/2\sqrt{t})[t \ge \tau]$$

for $\tau > 0$.

The moderately clipped LASSO and sparse ridge are simple smooth interpolations between the MCP (near the origin) and the LASSO and ridge, respectively. The log and bridge penalties are modified to be linear over $t \in (0, \tau]$ so that they have finite right derivative at the origin. See the plot for graphical comparison of the penalties introduced here.

CCCP-MLQA algorithm

The CCCP-MLQA algorithm iteratively conducts two main steps: CCCP (Yuille and Rangarajan, 2003) and MLQA (Lee et al., 2016) steps. The CCCP step decomposes the penalty J_λ as in (2) and then minimizes the tight convex upper bound obtained from a linear approximation of D_λ. The MLQA step first minimizes a quadratic approximation of the loss L and then modifies the solution to keep descent property.

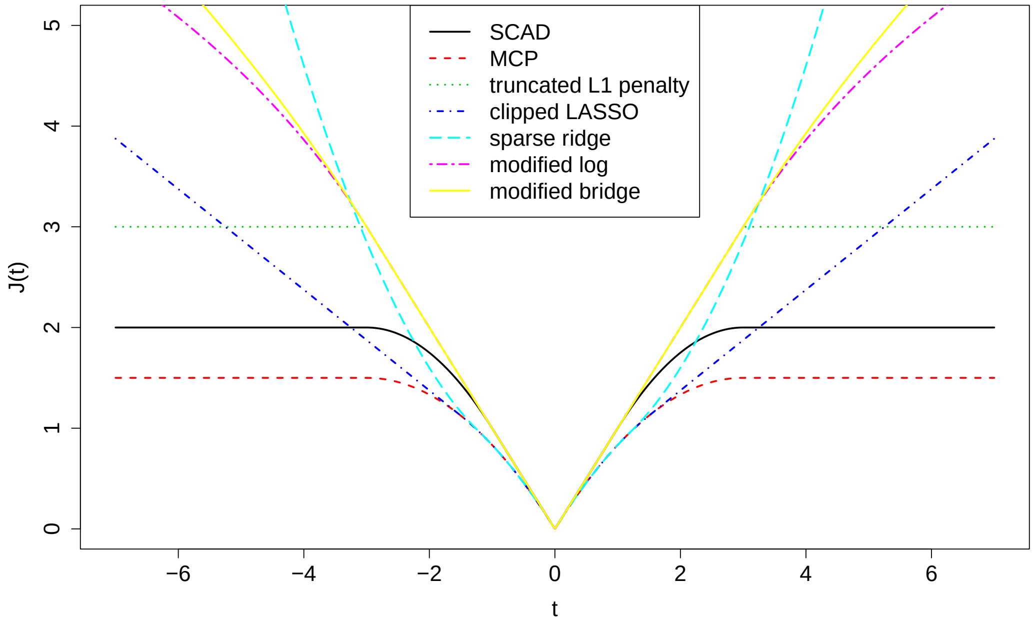

Figure 1: Plot of various penalties with $\lambda = 1, \tau = 3$ and $\gamma = 0.5$.

Concave-convex procedure

The objective function Q_λ in (1) can be rewritten by using the decomposition in (2) as

$$Q_\lambda(\boldsymbol{\beta}) = L(\boldsymbol{\beta}) + \sum_{j=1}^{p} D_\lambda(\beta_j) + \kappa_\lambda \sum_{j=1}^{p} |\beta_j| \tag{3}$$

so that $Q_\lambda(\boldsymbol{\beta})$ becomes a sum of convex, $L(\boldsymbol{\beta}) + \kappa_\lambda \sum_{j=1}^{p} |\beta_j|$, and concave, $\sum_{j=1}^{p} D_\lambda(\beta_j)$, functions. Hence the tight convex upper bound of $Q_\lambda(\boldsymbol{\beta})$ (Yuille and Rangarajan, 2003) becomes

$$U_\lambda(\boldsymbol{\beta}; \tilde{\boldsymbol{\beta}}) = L(\boldsymbol{\beta}) + \sum_{j=1}^{p} \partial D_\lambda(\tilde{\beta}_j)\beta_j + \kappa_\lambda \sum_{j=1}^{p} |\beta_j|, \tag{4}$$

where $\tilde{\boldsymbol{\beta}} = (\tilde{\beta}_1, \ldots, \tilde{\beta}_p)^T$ is a given point and $\partial D_\lambda(\tilde{\beta}_j)$ is a subgradient of $D_\lambda(\beta_j)$ at $\beta_j = \tilde{\beta}_j$. Algorithm 1 summarizes the CCCP step for minimizing Q_λ.

Algorithm 1: minimizing $Q_\lambda(\boldsymbol{\beta})$

```
1. Set $\tilde{\boldsymbol{\beta}}$.
2. Update $\tilde{\boldsymbol{\beta}}$ by $\tilde{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} U_\lambda(\boldsymbol{\beta}; \tilde{\boldsymbol{\beta}})$.
3. Repeat the Step 2 until convergence.
```

Modified Local quadratic approximation

Algorithm 1 includes minimizing $U_\lambda(\boldsymbol{\beta}; \tilde{\boldsymbol{\beta}})$ in (4) given a solution $\tilde{\boldsymbol{\beta}}$. An easy way is iteratively minimizing local quadratic approximation (LQA) of L around $\tilde{\boldsymbol{\beta}}$:

$$L(\boldsymbol{\beta}) \approx \tilde{L}(\boldsymbol{\beta}; \tilde{\boldsymbol{\beta}}) = L(\tilde{\boldsymbol{\beta}}) + \nabla L(\tilde{\boldsymbol{\beta}})^T (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}}) + (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}})^T \nabla^2 L(\tilde{\boldsymbol{\beta}}) (\boldsymbol{\beta} - \tilde{\boldsymbol{\beta}}) / 2,$$

where $\nabla L(\boldsymbol{\beta}) = \partial L(\boldsymbol{\beta}) / \partial \boldsymbol{\beta}$ and $\nabla^2 L(\boldsymbol{\beta}) = \partial^2 L(\boldsymbol{\beta}) / \partial \boldsymbol{\beta}^2$. Then $U_\lambda(\boldsymbol{\beta}; \tilde{\boldsymbol{\beta}})$ can be minimized by iteratively minimizing

$$\tilde{U}_\lambda(\boldsymbol{\beta}; \tilde{\boldsymbol{\beta}}) = \tilde{L}(\boldsymbol{\beta}; \tilde{\boldsymbol{\beta}}) + \sum_{j=1}^{p} \partial D_\lambda(|\tilde{\beta}_j|)\beta_j + \kappa_\lambda \sum_{j=1}^{p} |\beta_j|. \tag{5}$$

It is easy to minimize $\tilde{U}_\lambda\left(\boldsymbol{\beta};\tilde{\boldsymbol{\beta}}\right)$ since it is simply a quadratic function and the penalty term is the LASSO. For the algorithm, we use the coordinate descent algorithm introduced by Friedman et al. (2007). Note that the LQA algorithm may not have the descent property. Hence, we incorporate the modification step to ensure the descent property. Let $\tilde{\boldsymbol{\beta}}^a$ be the minimizer of $\tilde{U}_\lambda\left(\boldsymbol{\beta};\tilde{\boldsymbol{\beta}}\right)$. Then we modify the solution $\tilde{\boldsymbol{\beta}}^a$ by $\tilde{\boldsymbol{\beta}}^{\hat{h}}$ whenever it violates the descent property, i.e., $U_\lambda(\tilde{\boldsymbol{\beta}}^a;\tilde{\boldsymbol{\beta}}) > U_\lambda\left(\tilde{\boldsymbol{\beta}};\tilde{\boldsymbol{\beta}}\right)$:

$$\tilde{\boldsymbol{\beta}}^{\hat{h}} = \hat{h}\tilde{\boldsymbol{\beta}}^a + \left(1-\hat{h}\right)\tilde{\boldsymbol{\beta}}, \tag{6}$$

where $\hat{h} = \arg\min_{h>0} U_\lambda\left(h\tilde{\boldsymbol{\beta}}^a + (1-h)\,\tilde{\boldsymbol{\beta}};\tilde{\boldsymbol{\beta}}\right)$. This modification step in (6) guarantees the descent property of the LQA algorithm (Lee et al., 2016).

Algorithm 2: minimizing $U_\lambda\left(\boldsymbol{\beta};\tilde{\boldsymbol{\beta}}\right)$

```
1. Set $\tilde{\boldsymbol{\beta}}$.
2. Find $\tilde{\boldsymbol{\beta}}^a = \arg\min_{\boldsymbol{\beta}} \tilde{U}_\lambda\left(\boldsymbol{\beta};\tilde{\boldsymbol{\beta}}\right)$.
3. Find $\hat{h} = \arg\min_{h>0} U_\lambda\left(h\tilde{\boldsymbol{\beta}}^a + (1-h)\,\tilde{\boldsymbol{\beta}};\tilde{\boldsymbol{\beta}}\right)$.
4. Update $\tilde{\boldsymbol{\beta}}$ by $\hat{h}\tilde{\boldsymbol{\beta}}^a + \left(1-\hat{h}\right)\tilde{\boldsymbol{\beta}}$.
5. Repeat the Step 2–4 until convergence.
```

Efficient path construction over λ

Usually, the computation time of the algorithm rapidly increases as the number of non-zero parameters increases or λ decreases toward zero. To accelerate the algorithm, we incorporate the active-set-control procedure while constructing the solution path over a decreasing sequence of λ.

Assume that λ is given and we have an initial solution $\tilde{\boldsymbol{\beta}}$ which is expected to be very close to the minimizer of $Q_\lambda\left(\boldsymbol{\beta}\right)$. First we check the first order KKT optimality conditions:

$$\partial Q_\lambda\left(\tilde{\boldsymbol{\beta}}\right)/\partial\beta_j = 0, j \in \mathcal{A} \text{ and } |\partial Q_\lambda\left(\tilde{\boldsymbol{\beta}}\right)/\partial\beta_j| \le \kappa_\lambda, j \in \mathcal{N}, \tag{7}$$

where $\mathcal{A} = \{j : \tilde{\beta}_j \neq 0\}$ and $\mathcal{N} = \{j : \tilde{\beta}_j = 0\}$. We stop the algorithm if the conditions are satisfied else update $\mathcal{N}$ and $\tilde{\boldsymbol{\beta}}$ by $\mathcal{N} = \mathcal{N} \setminus \{j_{\max}\}$ and

$$\tilde{\boldsymbol{\beta}} = \arg\min_{\beta_j=0, j\in\mathcal{N}} Q_\lambda\left(\boldsymbol{\beta}\right), \tag{8}$$

respectively, where $j_{\max} = \arg\max_{j\in\mathcal{N}} |\partial Q_\lambda\left(\tilde{\boldsymbol{\beta}}\right)/\partial\beta_j|$. We keep these iterations until the KKT conditions in (7) are satisfied with $\tilde{\boldsymbol{\beta}}$. The key step is (8) which is easy and fast to obtain by using Algorithm 1 and 2 since the objective function only includes the parameters in $\mathcal{A} \cup \{j_{\max}\}$.

Algorithm 3: minimizing $Q_\lambda\left(\boldsymbol{\beta}\right)$

```
1. Set $\tilde{\boldsymbol{\beta}}$.
2. Set $\mathcal{A} = \{j : \tilde{\beta}_j \neq 0\}$ and $\mathcal{N} = \{j : \tilde{\beta}_j = 0\}$.
3. Check whether $\partial Q_\lambda\left(\tilde{\boldsymbol{\beta}}\right)/\partial\beta_j = 0, j \in \mathcal{A}$ and $|\partial Q_\lambda\left(\tilde{\boldsymbol{\beta}}\right)/\partial\beta_j| \le \kappa_\lambda, j \in \mathcal{N}$.
4. Update $\mathcal{N}$ by $\mathcal{N} \setminus \{j_{\max}\}$, where $j_{\max} = \arg\max_{j\in\mathcal{N}} |\partial Q_\lambda\left(\tilde{\boldsymbol{\beta}}\right)/\partial\beta_j|$.
5. Update $\tilde{\boldsymbol{\beta}}$ by $\tilde{\boldsymbol{\beta}} = \arg\min_{\beta_j=0, j\in\mathcal{N}} Q_\lambda\left(\boldsymbol{\beta}\right)$.
6. Repeat the Step 2–5 until the KKT conditions satisfy.
```

Remark 1 *The number of variables that violates the KKT conditions could be large for some high-dimensional cases. In this case, it may be inefficient to add only one variable $j_{\max}$ into $\mathcal{A}$. It would be more efficient to add more variables into $\mathcal{A}$. However, when the number variables added is too large, it also is inefficient. With many experiences, we found that the algorithm would be efficient with 10 variables.*

In practice, we want to approximate the whole solution path or surface of the minimizer $\hat{\boldsymbol{\beta}}^\lambda$ as a function of λ. For the purpose, we first construct a decreasing sequence $\lambda_{\max} = \lambda_0 > \lambda_1 > \cdots > \lambda_{n-1} > \lambda_n = \lambda_{\min}$ and then obtain the corresponding sequence of minimizers $\hat{\boldsymbol{\beta}}^{\lambda_0}, \ldots, \hat{\boldsymbol{\beta}}^{\lambda_n}$. Let $\partial Q_\lambda\left(\mathbf{0}\right)$ be the subdifferential of Q_λ at $\mathbf{0}$. Then we can see that $\mathbf{0} \in \partial Q_\lambda\left(\mathbf{0}\right) = \{\nabla L\left(\mathbf{0}\right) + \boldsymbol{\delta} : \max_j |\delta_j| \le \kappa_\lambda\}$, for any $\kappa_\lambda > \kappa_{\lambda_{\max}} = \max_j |\partial L\left(\mathbf{0}\right)/\partial\beta_j|$, which implies the p-dimensional zero vector is the exact minimizer of $Q_\lambda\left(\boldsymbol{\beta}\right)$ when $\kappa_\lambda \ge \kappa_{\lambda_{\max}}$. Hence, we start from the largest value $\lambda = \lambda_{\max}$ that satisfies $\kappa_{\lambda_{\max}} = \max_j |\partial L\left(\mathbf{0}\right)/\partial\beta_j|$, and then we continue down to $\lambda = \lambda_{\min} = \epsilon\lambda_{\max}$, where ϵ is a predetermined ratio such as $\epsilon = 0.01$. Once we obtain the minimizer $\hat{\boldsymbol{\beta}}^{\lambda_{k-1}}$ then it is easy to find $\hat{\boldsymbol{\beta}}^{\lambda_k}$ by using $\hat{\boldsymbol{\beta}}^{\lambda_{k-1}}$ as an initial solution in Algorithm 3, which is expected to be close to $\hat{\boldsymbol{\beta}}^{\lambda_k}$ for a finely divided λ sequence. This scheme is called the *warm start strategy*, which makes the algorithm more stable and efficient (Friedman et al., 2010).

3 The R package ncpen

In this section, we introduce the main functions with various options and user-friendly functions implemented in the package **ncpen** for the users. Next section will illustrate how the various options in the main function make a difference in data analysis through numerical examples.

The R package **ncpen** contains the main functions: ncpen() for fitting various nonconvex penalized regression models, cv.ncpen() and gic.ncpen() for selecting the optimal model from a sequence of the regularization path based on cross-validation and a generalized information criterion(Wang et al., 2007, 2009; Fan and Tang, 2013; Wang et al., 2013). In addition, the useful functions are also implemented in the package: sam.gen.ncpen() for generating a synthetic data from various models with the correlation structure in (11), plot() for graphical representation, coef() for extracting coefficients from the fitted object, predict() for making predictions from new design matrix. The followings are the basic usage of the main functions:

```
## linear regression with scad penalty
n=100; p=10
sam = sam.gen.ncpen(n=n,p=p,q=5,cf.min=0.5,cf.max=1,corr=0.5,family="gaussian")
x.mat = sam$x.mat; y.vec = sam$y.vec
fit = ncpen(y.vec=y.vec,x.mat=x.mat,family="gaussian",penalty="scad")
coef(fit); plot(fit)

# prediction from a new dataset
test.n=20
newx = matrix(rnorm(test.n*p),nrow=test.n,ncol=p)
predict(fit,new.x.mat=newx)

# selection of the optimal model based on the cross-validation
cv.fit = cv.ncpen(y.vec=y.vec,x.mat=x.mat,family="gaussian",penalty="scad")
coef(cv.fit)

# selection of the optimal model using the generalized information criterion
fit = ncpen(y.vec=y.vec,x.mat=x.mat,family="gaussian",penalty="scad")
gic.ncpen(fit)
```

The main function ncpen() provides various options which produce different penalized estimators. The other packages for nonconvex penalized regressions also have similar options: ℓ_2-regularization and penalty weights. However, the package **ncpen** provides the two unique options for standaradization and initial value. Below, we briefly describe the options in the main function ncpen().

Ridge regularization

The option alpha in the main functions forces the algorithm to solve the following penalized problem with the ℓ_2-regularization or ridge effect (Zou and Hastie, 2005).

$$Q_\lambda(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\ell_i(\boldsymbol{\beta}) + \alpha\sum_{j=1}^{p}J_\lambda(|\beta_j|) + (1-\alpha)\lambda\sum_{j=1}^{p}\beta_j^2, \tag{9}$$

where $\alpha \in [0,1]$ is the value from the option alpha, which is the mixing parameter between the penalties J_λ and ridge. The objective function in (9) includes the elastic net (Zou and Hastie, 2005) when $J_\lambda(t) = \lambda t$ and Mnet (Huang et al., 2016a) when $J_\lambda(\cdot)$ is the MC penalty. By controlling the option alpha, we can treat the problem with highly correlated variables, and it makes the algorithm more stable also since the minimum eigenvalue of the Hessian marix becomes large up to factor $(1-\alpha)\lambda$ (Zou and Hastie, 2005; Lee and Breheny, 2015; Huang et al., 2016a).

Observation and penalty weights

We can give different weights for each observation and penalty by the options `obs.weight` and `pen.weight`, which provides the minimizer of

$$Q_\lambda(\boldsymbol{\beta}) = \sum_{i=1}^{n} d_i \ell_i(\boldsymbol{\beta}) + \sum_{j=1}^{p} w_j J_\lambda(|\beta_j|), \tag{10}$$

where d_i is the weight for the ith observation and w_j is the penalty weight for the jth variable. For example, controlling observation weights is required for the linear regression model with heteroscedastic error variance. Further, we can compute adaptive versions of penalized estimators by giving different penalty weights as in the adaptive LASSO (Zou, 2006).

Standardization

It is common practice to standardize variables prior to fitting the penalized models, but one may opt not to. Hence, we provide the option `x.standardize` for flexible analysis. The option `x.standardize=TRUE` means that the algorithm solves the original penalized problem in (1), with the standardized (scaled) variables, and then the resulting solution $\hat{\beta}_j$ is converted to the original scale by $\hat{\beta}_j/s_j$, where $s_j = \sum_{i=1}^{n} x_{ij}^2/n$. When the penalty J_λ is the LASSO penalty, this procedure is equivalent to solving following penalized problem

$$Q_\lambda^s(\boldsymbol{\beta}) = L(\boldsymbol{\beta}) + \sum_{j=1}^{p} \lambda_j |\beta_j|,$$

where $\lambda_j = \lambda s_j$, which is another adaptive version of the LASSO being different from the adaptive LASSO (Zou, 2006).

Initial value

We introduced the warm start strategy for speed up the algorithm, but the solution path, in fact, depends on the initial solution of the CCCP algorithm because of the non-convexity. The option `local=TRUE` in **ncpen** provides the same initial value specified by the option `local.initial` into each CCCP iterations for whole λ values. The use of the option `local=TRUE` makes the algorithm slower but the performance of the resulting estimator would be often improved as provided a good initial such as the maximum likelihood estimator or LASSO.

4 Numerical illustrations

Elapsed times

We consider the linear and logistic regression models to calculate the total elapsed time for constructing the solution path over 100 λ values:

$$y = \mathbf{x}^T\boldsymbol{\beta} + \varepsilon \text{ and } \mathbf{P}(y=1|\mathbf{x}) = \frac{exp(\mathbf{x}^T\boldsymbol{\beta})}{1+exp(\mathbf{x}^T\boldsymbol{\beta})} \tag{11}$$

where $\mathbf{x} \sim N_p(\mathbf{0}, \Sigma)$ with $\Sigma_{jk} = 0.5^{|j-k|}$, $\beta_j = 1/j$ for $j,k = 1,\cdots,p$ and $\varepsilon \sim N(0,1)$. The averaged elapsed times of **ncpen** in 100 random repetitions are summarized in Table 1 and 2 for various n and p, where the penalties are the SCAD, MCP, truncated ℓ_1 (TLP), moderately clipped LASSO (CLASSO), sparse ridge (SR), modified bridge (MBR) and log (MLOG). For comparison, we try **ncvreg** for the SCAD also. The results show that all methods in **ncpen** are feasible for high-dimensional data.

Standardization effect

We compare the solution paths based on the diabetes samples available from **lars** package (Efron et al., 2004), where the sample size $n = 442$ and the number of covariates $p = 64$, including quadratic and interaction terms. Figure 2 shows four plots where the top two panels draw the solution paths from the LASSO and SCAD with $\tau = 3.7$ given by **ncvreg** and bottom two panels draw those from the SCAD with $\tau = 3.7$ based on **ncpen** with and without standardization of covariates. Two solution paths from **ncvreg** and **ncpen** with standardization are almost the same since **ncvreg** standardizes the covariates by default, which is somewhat different from that of **ncpen** without standardization.

Table 1: Elapsed times for constructing the entire solution path where $p = 500$ and various n

Model	n	**ncvreg**	SCAD	MCP	TLP	CLASSO	SR	MBR	MLOG
Linear regression	200	0.0226	0.1277	0.1971	0.0333	0.0696	0.0618	0.0620	0.0476
	400	0.0329	0.1082	0.2031	0.0662	0.1041	0.1025	0.1160	0.0919
	800	0.0347	0.1008	0.1867	0.0865	0.0993	0.1067	0.1425	0.1197
	1600	0.0665	0.2035	0.3170	0.1717	0.1847	0.1983	0.2669	0.2301
	3200	0.1394	0.4341	0.6173	0.3541	0.3962	0.4161	0.5505	0.4678
	6400	0.2991	0.9853	1.2045	0.7955	0.8788	0.9066	1.2281	1.0148
Logistic regression	200	0.0565	0.0454	0.0400	0.0391	0.0148	0.0160	0.0379	0.0411
	400	0.0787	0.1113	0.0971	0.0747	0.0556	0.0608	0.0969	0.0808
	800	0.0907	0.1570	0.1623	0.1198	0.0777	0.1015	0.1511	0.1298
	1600	0.1682	0.2965	0.3007	0.2294	0.1640	0.2088	0.3002	0.2451
	3200	0.3494	0.6480	0.6258	0.4655	0.3513	0.4423	0.6395	0.5305
	6400	0.7310	1.4144	1.3711	1.0268	0.8389	1.0273	1.4445	1.1827

Table 2: Elapsed times for constructing the entire solution path where $n = 500$ and various p

Model	p	**ncvreg**	SCAD	MCP	TLP	CLASSO	SR	MBR	MLOG
Linear regression	200	0.0150	0.0733	0.2201	0.0433	0.0629	0.0981	0.0909	0.0721
	400	0.0210	0.0664	0.1588	0.0532	0.0617	0.0678	0.0941	0.0813
	800	0.0538	0.1650	0.2172	0.1107	0.1505	0.1457	0.1750	0.1383
	1600	0.0945	0.2703	0.2946	0.1793	0.2253	0.2221	0.2672	0.2045
	3200	0.1769	0.5071	0.5032	0.3379	0.3972	0.3986	0.4801	0.3684
	6400	0.3439	1.0781	1.0228	0.7366	0.8001	0.8207	1.0210	0.7830
Logistic regression	200	0.0590	0.1065	0.1029	0.0750	0.0465	0.0696	0.0978	0.0804
	400	0.0568	0.1054	0.1044	0.0753	0.0453	0.0593	0.0941	0.0809
	800	0.1076	0.1555	0.1349	0.1103	0.0873	0.0934	0.1423	0.1163
	1600	0.1327	0.1944	0.1591	0.1419	0.1122	0.1151	0.1842	0.1460
	3200	0.2073	0.3120	0.2529	0.2382	0.1885	0.1948	0.3055	0.2415
	6400	0.3843	0.5893	0.4792	0.4646	0.3539	0.3576	0.5978	0.4677

Figure 3 shows the solution paths from six penalties with standardization by default in **ncpen**: the MCP, truncated ℓ_1, modified log, bridge, moderately clipped LASSO and sparse ridge.

Ridge regularization effect

There are cases when we need to introduce the ridge penalty for some reasons, and **ncpen** provides a hybrid version of the penalties: $\alpha J_\lambda(|t|) + (1-\alpha)|t|^2$, where α is the mixing parameter between the penalty J_λ and ridge effects. For example, the non-convex penalties often produce parameters that diverge to infinity for the logistic regression because of perfect fitting. Figure 4 shows the effects of ridge penalty where the prostate tumor gene expression data in **spls** are used for illustration. The solution paths using the top 50 variables with high variances are drawn when $\alpha \in \{1, 0.7, 0.3, 0\}$ for the SCAD and modified bridge penalties. The solution paths without ridge effect ($\alpha = 1$) tend to diverge as λ decreases and become stabilized as the ridge effect increases ($\alpha \downarrow$) (Lee and Breheny, 2015).

Initial based solution path

We introduced the warm start strategy for speed up the algorithm but the solution path, in fact, depends on the initial solution because of the non-convexity. For comparison, we use the prostate tumor gene expression data and the results are displayed in Figure 5 and Table 3. In the figure, left panels show the solution paths for the SCAD, MCP and clipped LASSO obtained by the warm start, and the right panels show those obtained by using the LASSO as a global initial for the CCCP algorithm. Figure 5 shows two strategies for initial provide very different solution paths, which may result in different performances of the estimators. We compare the prediction accuracy and selectivity of the estimators by two strategies. The results are obtained by 300 random partitions of data set divided into two parts, training (70%) and test (30%) datasets. For each training data, the optimal tuning parameter values are selected by 10-fold cross-validation, and then we compute the prediction error(the percentage of the misclassified samples) on each test dataset and the number of selected

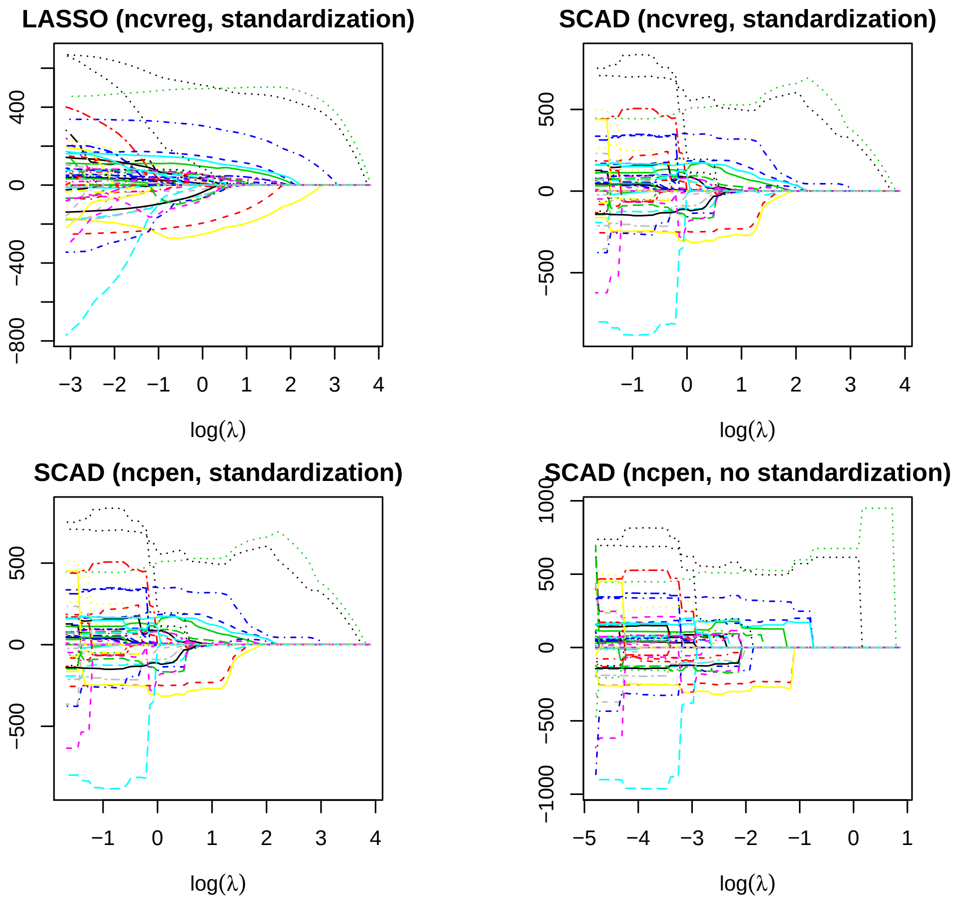

Figure 2: Solution paths from the **ncvreg** and **ncpen** for the LASSO and SCAD.

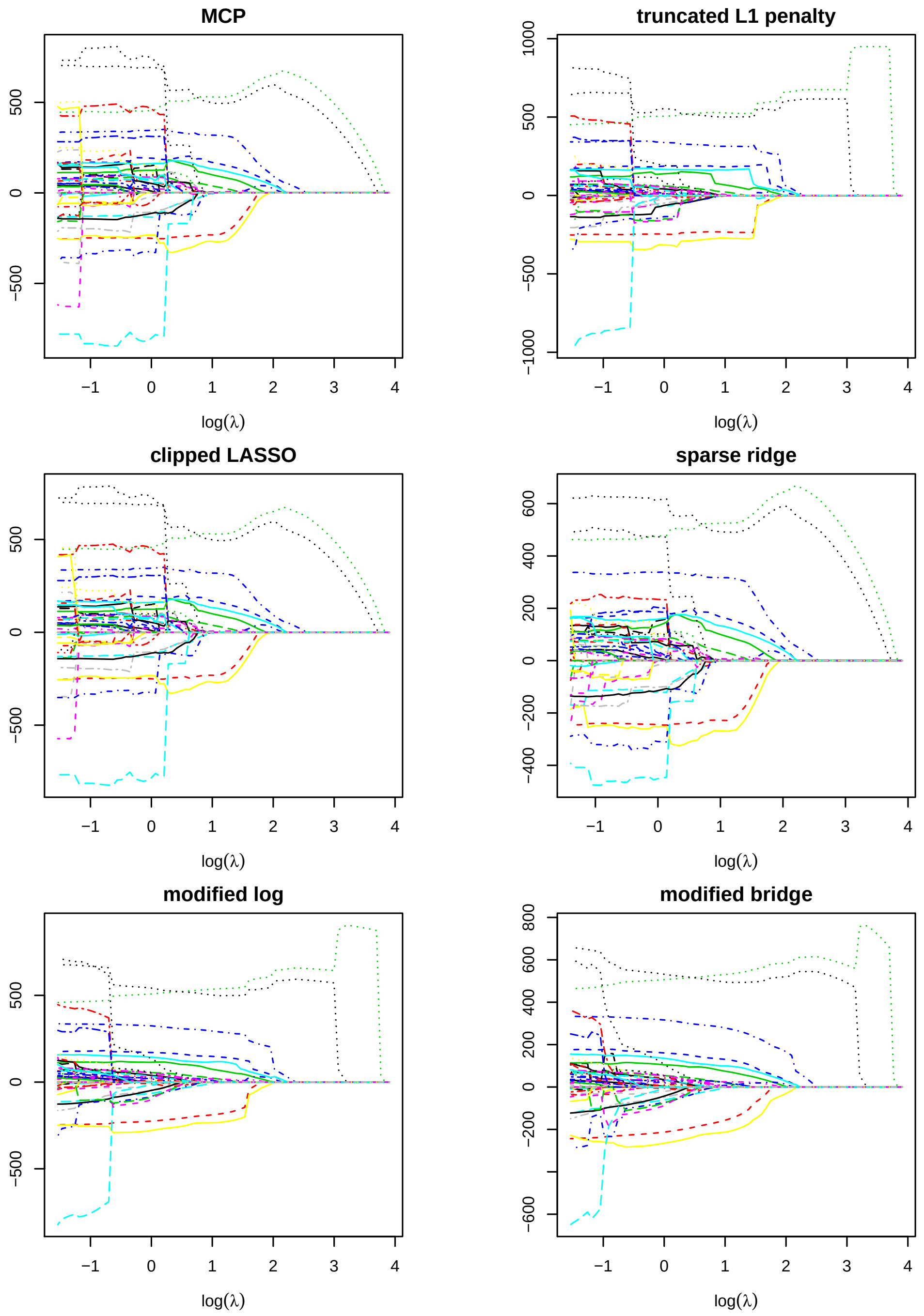

Figure 3: Solution paths from **ncpen** with six non-convex penalties.

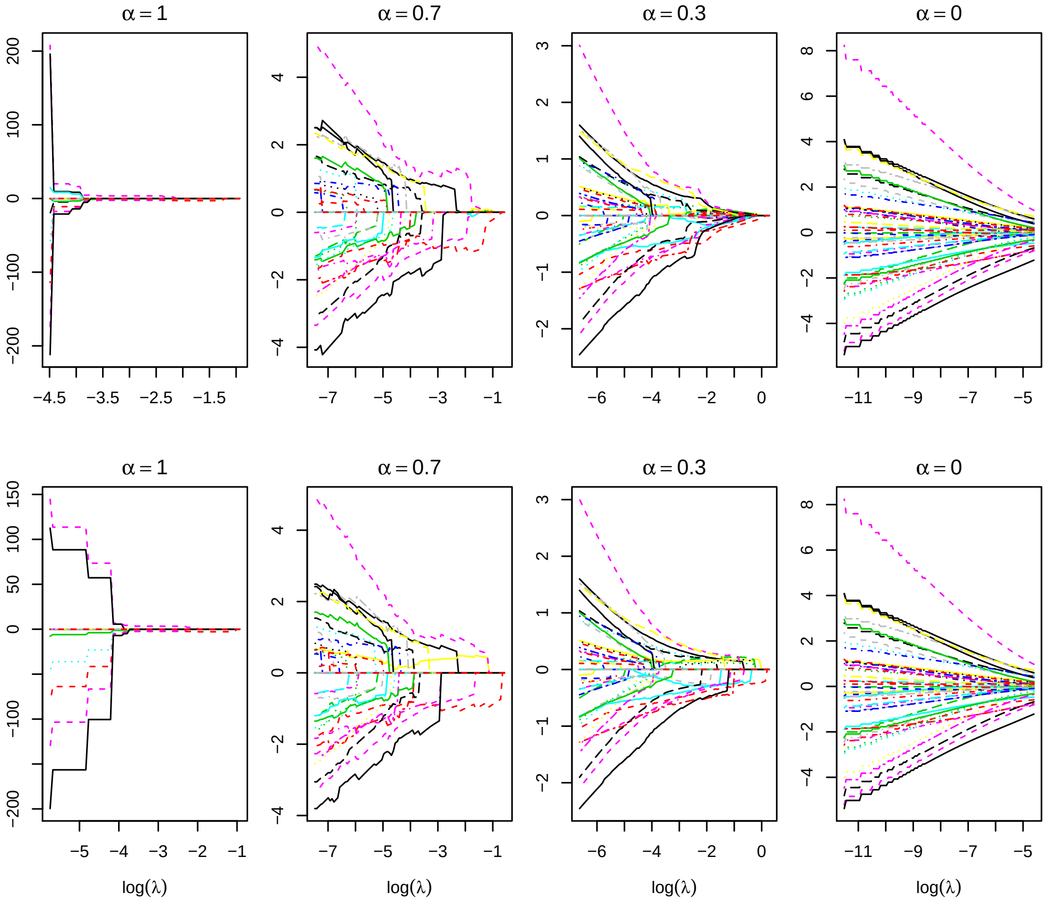

Figure 4: Solution path traces with ridge penalty. Top and bottom panels are drawn from the SCAD and modified bridge penalties, respectively.

nonzero variables on each training dataset. Table 3 shows all methods by the global initial perform better than those by the warm start strategy. In summary, the nonconvex penalized estimation depends on the initial solution, and the non-convex penalized estimator by a good initial would improve its performance.

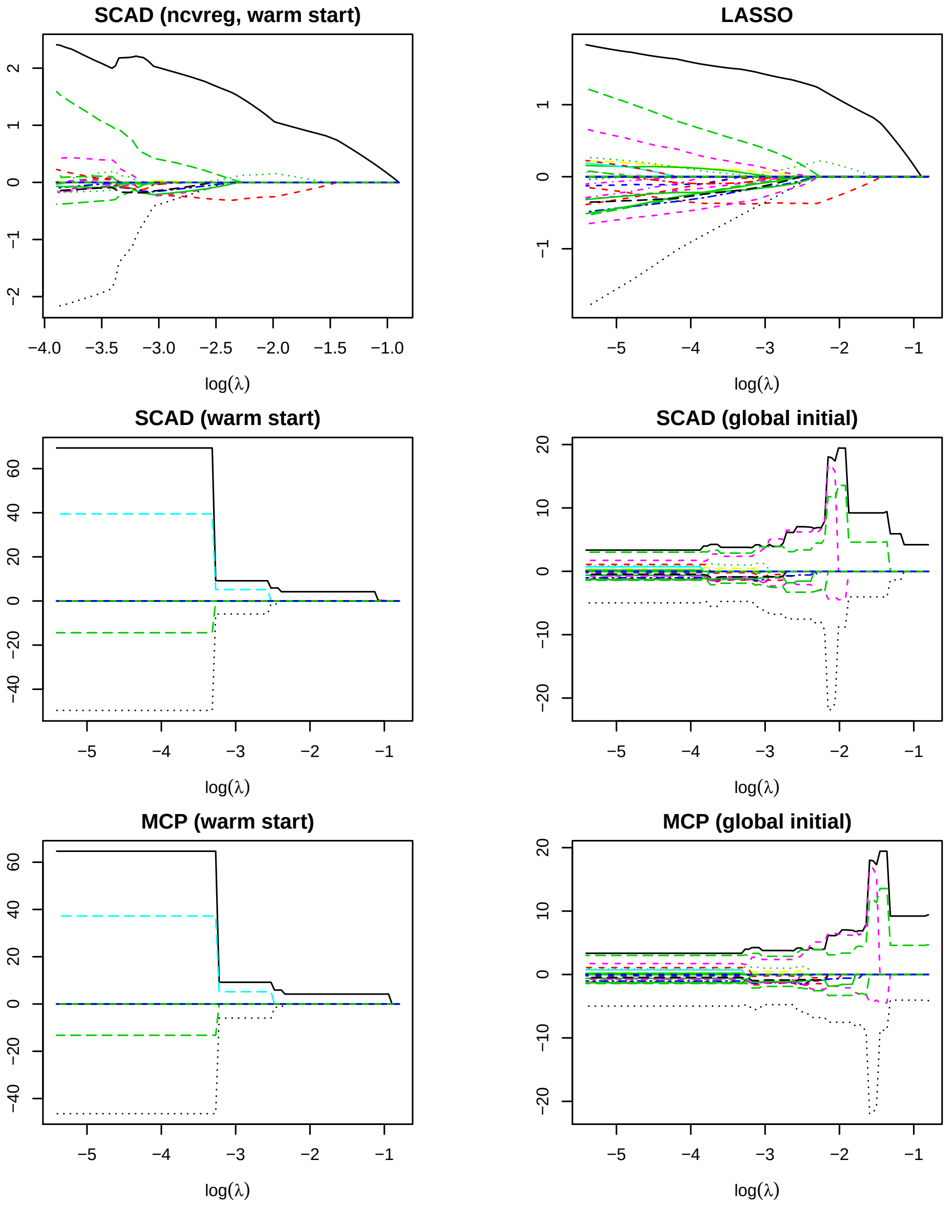

Figure 5: Solution paths of the SCAD and MCP with warm start and global initial solution.

Table 3: Comparison of the warm start and global initial strategies for each method.

	warm start		global initial	
Method	prediction error	# variables	prediction error	# variables
SCAD	9.44 (.2757)	1.19 (.0268)	9.30 (.3091)	7.02 (.3907)
MCP	9.38 (.2774)	1.14 (.0234)	8.84 (.2657)	7.41 (.3771)
TLP	9.44 (.2647)	1.18 (.0254)	7.47 (.2677)	19.01 (.1710)
CLASSO	9.12 (.2749)	4.65 (.1914)	8.20 (.2785)	8.14 (.1542)
SR	9.38 (.2875)	5.15 (.2290)	7.94 (.2677)	15.13 (.2283)
MBR	9.78 (.2621)	1.29 (.0352)	8.21 (.3004)	8.75 (.1712)
MLOG	9.51 (.2627)	1.16 (.0233)	7.66 (.2746)	15.21 (.1816)

5 Concluding remarks

We have developed the R package **ncpen** for estimating generalized linear models with various concave penalties. The unified algorithm implemented in **ncpen** is flexible and efficient. The package also provides various user-friendly functions and user-specific options for different penalized estimators. The package is currently available with a general public license (GPL) from the Comprehensive R Archive Network at https://CRAN.R-project.org/package=ncpen. Our **ncpen** package implements internal optimization algorithms implemented in C++ benefiting from **Rcpp** package (Eddelbuettel et al., 2011).

6 Acknowledgements

This research is supported by the Julian Virtue Professorship 2016-18 Endowment at the Pepperdine Graziadio Business School at Pepperdine University, and the National Research Foundation of Korea (NRF) funded by the Korea government (No. 2020R1I1A3071646 and 2020R1F1A1A01071036).

Bibliography

P. Breheny and J. Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, 5(1):232, 2011. URL https://doi.org/10.1214/10-AOAS388. [p120]

D. Eddelbuettel, R. François, J. Allaire, K. Ushey, Q. Kou, N. Russel, J. Chambers, and D. Bates. Rcpp: Seamless r and c++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL https://doi.org/10.18637/jss.v040.i08. [p131]

B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *Annals of Statistics*, 32(2): 407–499, 2004. URL https://www.jstor.org/stable/3448465. [p120, 125]

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. URL https://doi.org/10.1198/016214501753382273. [p120, 121]

Y. Fan and C. Y. Tang. Tuning parameter selection in high dimensional penalized likelihood. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):531–552, 2013. URL https://doi.org/10.1111/rssb.12001. [p124]

J. Friedman. Fast sparse regression and classification. *International Journal of Forecasting*, 28(3):722–738, 2012. URL https://doi.org/10.1016/j.ijforecast.2012.05.001. [p120]

J. Friedman, T. Hastie, H. Höfling, R. Tibshirani, et al. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007. URL https://doi.org/10.1214/07-AOAS131. [p120, 123]

J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010. URL https://doi.org/10.18637/JSS.V033.I01. [p124]

J. Huang, J. L. Horowitz, and S. Ma. Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *Annals of Statistics*, pages 587–613, 2008. URL https://doi.org/10.1214/009053607000000875. [p120, 121]

J. Huang, P. Breheny, S. Lee, S. Ma, and C. Zhang. The mnet method for variable selection. *Statistica Sinica*, 10, 2016a. URL https://doi.org/10.5705/ss.202014.0011. [p120, 124]

J. Huang, P. Breheny, S. Lee, S. Ma, and C.-H. Zhang. The mnet method for variable selection. *Statistica Sinica*, pages 903–923, 2016b. [p120]

D. Jiang and J. Huang. Majorization minimization by coordinate descent for concave penalized generalized linear models. *Statistics and Computing*, 24(5):871–883, 2014. URL https://doi.org/10.1007/s11222-013-9407-3. [p120]

Y. Kim and S. Kwon. Global optimality of nonconvex penalized estimators. *Biometrika*, 99(2):315–325, 2012. URL https://doi.org/10.1093/biomet/asr084. [p120]

Y. Kim, H. Choi, and H.-S. Oh. Smoothly clipped absolute deviation on high dimensions. *Journal of the American Statistical Association*, 103(484):1665–1673, 2008. URL https://doi.org/10.1198/016214508000001066. [p120]

S. Kwon and Y. Kim. Large sample properties of the scad-penalized maximum likelihood estimation on high dimensions. *Statistica Sinica*, pages 629–653, 2012. URL https://doi.org/10.5705/ss.2010.027. [p120]

S. Kwon, Y. Kim, and H. Choi. Sparse bridge estimation with a diverging number of parameters. *Statistics and Its Interface*, 6(2):231–242, 2013. URL https://doi.org/10.4310/SII.2013.V6.N2.A7. [p120, 121]

S. Kwon, S. Lee, and Y. Kim. Moderately clipped lasso. *Computational Statistics & Data Analysis*, 92:53–67, 2015. URL https://doi.org/10.1016/j.csda.2015.07.001. [p120, 121]

S. Lee. A note on standardization in penalized regressions. *Journal of the Korean Data and Information Science Society*, 26(2):505–516, 2015. URL https://doi.org/10.7465/jkdi.2015.26.2.505. [p120]

S. Lee and P. Breheny. Strong rules for nonconvex penalties and their implications for efficient algorithms in high-dimensional regression. *Journal of Computational and Graphical Statistics*, 24(4):1074–1091, 2015. URL https://doi.org/10.1080/10618600.2014.975231. [p124, 126]

S. Lee, S. Kwon, and Y. Kim. A modified local quadratic approximation algorithm for penalized optimization problems. *Computational Statistics & Data Analysis*, 94:275–286, 2016. URL https://doi.org/10.1016/j.csda.2015.08.019. [p120, 121, 123]

R. Mazumder, J. H. Friedman, and T. Hastie. Sparsenet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495):1125–1138, 2011. URL https://doi.org/10.1198/jasa.2011.tm09738. [p120]

M. Y. Park and T. Hastie. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):659–677, 2007. URL https://doi.org/10.1111/j.1467-9868.2007.00607.x. [p120]

X. Shen, W. Pan, and Y. Zhu. Likelihood-based selection and sharp parameter estimation. *Journal of the American Statistical Association*, 107(497):223–232, 2012. URL https://doi.org/10.1080/01621459.2011.645783. [p120]

X. Shen, W. Pan, Y. Zhu, and H. Zhou. On constrained and regularized high-dimensional regression. *Annals of the Institute of Statistical Mathematics*, 65(5):807–832, 2013. URL https://doi.org/10.1007/s10463-012-0396-3. [p120, 121]

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, pages 267–288, 1996. URL https://doi.org/10.1111/j.2517-6161.1996.tb02080.x. [p120]

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001. URL https://doi.org/10.1023/A:1017501703105. [p120]

H. Wang, R. Li, and C.-L. Tsai. Tuning parameter selectors for the smoothly clipped absolute deviation method. *Biometrika*, 94(3):553–568, 2007. URL https://doi.org/10.1093/biomet/asm053. [p124]

H. Wang, B. Li, and C. Leng. Shrinkage tuning parameter selection with a diverging number of parameters. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):671–683, 2009. URL https://doi.org/10.1111/j.1467-9868.2008.00693.x. [p124]

L. Wang, Y. Kim, and R. Li. Calibrating non-convex penalized regression in ultra-high dimension. *Annals of Statistics*, 41(5):2505, 2013. URL https://doi.org/10.1214/13-AOS1159. [p124]

A. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15:915–936, 2003. URL https://doi.org/10.1162/08997660360581958. [p121, 122]

C. Zhang. Nearly unbiased variable selection under minimax concave penalty. *Ann. Stat.*, 38(2):894–942, 2010. URL https://doi.org/10.1214/09-AOS729. [p120, 121]

C.-H. Zhang and T. Zhang. A general theory of concave regularization for high-dimensional sparse estimation problems. *Statistical Science*, pages 576–593, 2012. URL https://doi.org/10.1214/12-STS399. [p120]

H. Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101 (476):1418–1429, 2006. URL https://doi.org/10.1198/016214506000000735. [p125]

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. URL https://doi.org/10.1111/j.1467-9868.2005.00503.x. [p120, 121, 124]

H. Zou and R. Li. One-step sparse estimates in nonconcave penalized likelihood models. *Annals of Statistics*, 36(4):1509, 2008. URL https://doi.org/10.1214/009053607000000802. [p120]

Dongshin Kim
Pepperdine Graziadio Business School
Pepperdine University
United States of America
dongshin.kim@pepperdine.edu

Sangin Lee
Department of Information and Statistics
Chungnam National University
Korea
sanginlee44@gmail.com

Sunghoon Kwon
Department of Applied Statistics
Konkuk University
Korea
shkwon0522@gmail.com