SAS Statistics Research and Applications
Paper #2022-04

Introducing Folded Concave Penalized Regression: New Variable Selection Methods in the REGSELECT Procedure in SAS® Viya®

Yingwei Wang, SAS Institute Inc., Cary, NC

Variable selection is a fundamental task in high-dimensional modeling and statistical learning. For linear models, traditional approaches such as stepwise regression use sequential procedures, which are computationally intensive and unstable. Alternative selection methods use sparsity-inducing penalized regression techniques to simultaneously select variables and estimate regression coefficients. In this paper, we introduce a class of nonconvex penalized regression methods of linear model selection, which are called folded concave penalization (FCP) methods. FCP estimators have many desirable properties, including sparsity, unbiasedness, and continuity. The corresponding objective functions are high-dimensional, nonlinear, and nonconvex with singularity at the origin. They can be solved by reformulation into problems in quadratic programming (QP) and mixed integer linear programming (MILP). In addition to covering the mathematical properties of the FCP methods, the paper presents practical examples by using the REGSELECT procedure in SAS Viya.

Introduction

With big data becoming ever more prevalent, it is worth noting that a large amount of the information that the data contain is irrelevant for either interpreting or predicting responses, and only a small amount is informative. To address this issue, in the past decade, variable selection methods have seen widespread usage in a diverse range of applications, leading to numerous challenges for statistical theory and implementations.

Let us begin with a typical setup of a linear regression model. If you observe a response vector $\mathbf{y} \in \mathbb{R}^n$ and a design matrix $X \in \mathbb{R}^{n\times p}$ that is constructed from model covariates, a linear regression model assumes the relationship between the response and covariates to be

$$\mathbf{y} = X\beta + \epsilon \tag{1}$$

where $\beta \in \mathbb{R}^p$ is an unknown vector and ϵ is a vector of noise modeled by random error.

The ordinary least squares (OLS) coefficients are the solution of the following optimization problem:

$$\hat{\beta}^{\text{ols}} = \arg\min_{\beta} \|\mathbf{y} - X\beta\|_2^2 = (X^T X)^{-1} X^T \mathbf{y} \tag{2}$$

In this paper, the matrix X^TX is assumed to be invertible unless otherwise specified.

Usually, the OLS estimate $\hat{\beta}^{\text{ols}}$ shown in (2) is dense in the sense that most estimates are nonzero. However, Bickel (2008) pointed out that the main goals of high-dimensional statistical modeling are twofold: (a) to construct as effective a method as possible for predicting a new Y given its X; and (b) to gain insight into the relationships between X and Y for scientific purposes. In high-dimensional statistical modeling, where a lot of information is either irrelevant or redundant, it is often reasonable to assume that β is a sparse vector in which many components are exactly zero or negligibly small; that is, only a few of the predictors contribute to the response. Therefore, the objective of variable selection for linear model (1) is to identify crucial predictors that have nonzero regression coefficients and to give accurate estimates of those coefficients.

In terms of how candidate models are examined and selected, variable selection methods in linear regression are grouped into two categories: sequential selection methods, such as forward selection, backward elimination, and stepwise regression; and penalized regression methods, also known as shrinkage or regularization methods. Penalization techniques have become increasingly popular, because they can perform the variable selection as well as the simultaneous estimation of the coefficients in the selected model.

A penalization technique can be described as follows. Suppose that $P_\lambda(\cdot)$ is a penalty function on the coefficient vector β indexed by a nonnegative penalization parameter λ. A shrinkage method solves the following penalized least squares (PLS) problem:

$$\hat{\beta}^{\text{pls}} = \arg\min_{\beta} \left\{ \|\mathbf{y} - X\beta\|_2^2 + P_\lambda(\beta) \right\} \tag{3}$$

The outcomes of the penalization procedure (3) typically depend on the amount of regularization. With the tuning parameter λ changing from 0 to ∞, penalization procedures often provide a solution path. A significant challenge here is to choose the right amount of regularization—in other words, the proper value of λ. The widely used methods include cross-validation and information criteria on the solution path.

Table 1 lists the most commonly used penalty functions. The ℓ_0, ℓ_1, ℓ_2 norms that are used to construct penalty functions are $\|\beta\|_0 = \sum_{j=1}^p 1\{\beta_j \neq 0\}, \|\beta\|_1 = \sum_{j=1}^p |\beta_j|, \|\beta\|_2 = \sqrt{\sum_{j=1}^p \beta_j^2}$, respectively. In certain penalty functions, there are also extra tuning parameters such as α that additionally control how a specific penalty function behaves. You can find the detailed formulas for smoothly clipped absolute deviation (SCAD) and the minimax concave penalty (MCP) in (4) and (7), respectively.

Several remarks are in order:

- **Sparsity.** Ridge regression, which minimizes a penalized residual sum of squares by using the squared ℓ_2 norm penalty, is used to improve the ordinary least squares estimate through a bias-variance trade-off (Marquardt and Snee 1975). However, like OLS, ridge regression does not usually yield a parsimonious model, because it generally keeps all predictors in the model. Except for the ridge penalty, all other penalized functions shown in Table 1 have the sparsity-inducing property, which means that their solutions can be sparse.

- **Convexity.** Among the sparsity-inducing penalties, the ℓ_1-norm penalties are the most popular; they appear in both the (adaptive) LASSO (Tibshirani 1996; Zou 2006) and (adaptive)

Table 1: Examples of Penalty Functions

Method	Penalty Function
Ridge	$P_{\lambda}^{\text{ridge}}(\beta) = \lambda\|\beta\|_2^2$
Best-subset	$P_{\lambda}^{\text{best-subset}}(\beta) = \lambda\|\beta\|_0$
LASSO Adaptive LASSO	$P_{\lambda}^{\text{lasso}}(\beta) = \lambda\|\beta\|_1$ $P_{\lambda}^{\text{a-lasso}}(\beta) = \lambda\|\mathbf{w}^t\beta\|_1$
Elastic net Adaptive elastic net	$P_{\lambda_1,\lambda_2}^{\text{enet}}(\beta) = \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2$ $P_{\lambda_1,\lambda_2}^{\text{a-enet}}(\beta) = \lambda_1\|\mathbf{w}^t\beta\|_1 + \lambda_2\|\beta\|_2^2$
SCAD MCP	$P_{\lambda,\alpha}^{\text{scad}}(\beta) = \sum_{j=1}^{m} P_{\lambda,\alpha}^{\text{scad}}(\|\beta_j\|)$ $P_{\lambda,\alpha}^{\text{mcp}}(\beta) = \sum_{j=1}^{m} P_{\lambda,\alpha}^{\text{mcp}}(\|\beta_j\|)$

elastic net (Zou and Hastie 2005; Zou and Zhang 2009) methods. The reason is that the convexity of ℓ_1-norm penalties allows direct application of the existing convex optimization techniques with well-established convergence properties. Convex penalties are advantageous from an optimization perspective but could lead to biased results. Using nonconvex penalties, on the other hand, is not ideal for optimization, but it could yield unbiased or nearly unbiased parameter values, especially when some parameters with large absolute values are present.

- **Computational complexity.** Convex optimization problems are relatively easy to solve because their global optimal solutions are efficiently computable. In comparison, for nonconvex problems, often there are multiple local minimizers, and it is hard to find the global minimizer. Best-subset selection with the ℓ_0 norm penalty is notorious for its computational infeasibility. Even though it uses the branch-and-bound algorithm, which efficiently solves them at low dimensions, in general the method has been shown to be NP-hard, which is the worst case among the nonconvex penalty functions.

Many exciting results, including both efficient algorithms and theoretical developments, have been obtained using nonconvex penalized regression. This paper focuses on the folded concave penalized (FCP) regression methods, and it uses several examples to show you how to perform variable selection in PROC REGSELECT.

Folded Concave Penalized Regression

As mentioned earlier, there are many choices for the penalty function $P_{\lambda}(\beta)$ in (3). A natural question is what kind of penalty functions are desirable for variable selection methods in high-dimensional modeling. Fan and Li (2001) proposed that a good penalty function should result in an estimator that has three properties: unbiasedness, sparsity, and continuity. Convex penalties are obviously better than nonconvex ones for practical implementation of optimization. However, they yield biased estimates of the parameters. In contrast, nonconvex penalties are used for regularization in high-dimensional statistical learning algorithms primarily because they yield unbiased or nearly unbiased estimates of the parameters in the model.

A variety of nonconvex penalties have been proposed. In the folded concave penalized sparse linear regression problem, $P_\lambda(\cdot)$ is substantiated by a folded concave penalty (FCP) that satisfies the following conditions:

- $P_\lambda(t)$ is nondecreasing and concave in $t \in \mathbb{R}$ with $P_\lambda(0) = 0$ and $P_\lambda(t) > 0$ if $t > 0$.
- $P_\lambda(t)$ is differentiable at any $t \in \mathbb{R}^+$.
- The first derivative $P'_\lambda(t) = 0$ for any $t \geq \alpha\lambda$.
- $1 \leq P_\lambda(t) \leq \lambda$ for any $t \geq 0$.

The estimators from FCPs achieve the three desirable properties: unbiasedness, sparsity, and continuity. Two of the earliest and most influential FCPs are the smoothly clipped absolute deviation and the minimax concave penalty.

SCAD

In smoothly clipped absolute deviation (SCAD) selection (Fan and Li 2001), the penalty takes the following form:

$$P^{\text{scad}}_{\lambda,\alpha}(\theta) = \begin{cases} \lambda\theta & \text{if } \; 0 \leq \theta \leq \lambda \\ \frac{-1}{2(\alpha-1)}\left(\theta^2 - 2\alpha\lambda\theta + \lambda^2\right) & \text{if } \; \lambda < \theta < \alpha\lambda \\ \frac{\alpha+1}{2}\lambda^2 & \text{if } \; \theta > \alpha\lambda \end{cases} \tag{4}$$

The SCAD estimator $\hat{\beta}^{\text{scad}}$ solves the following minimization problem:

$$\hat{\beta}^{\text{scad}} = \arg\min_{\beta} \left\{ \frac{1}{2}\|\mathbf{y} - \boldsymbol{X}\beta\|_2^2 + n\sum_{j=1}^{m} P^{\text{scad}}_{\lambda,\alpha}(|\beta_j|) \right\}$$

The quadratic program (QP) reformulation of the preceding SCAD problem is given by

$$\begin{array}{ll} \min\limits_{\beta,\mathbf{g},\mathbf{h}\in R^m} & \frac{1}{2}\left[\beta^T X^T X\beta + n(\alpha-1)\mathbf{g}^T\mathbf{g} + 2n\mathbf{g}^T\mathbf{h}\right] - y^T X\beta - n\alpha\lambda\mathbf{1}^T\mathbf{g} \\ \text{subject to} & -\mathbf{h} \leq \beta \leq \mathbf{h} \\ & \mathbf{0} \leq \mathbf{g} \leq \lambda 1 \end{array} \tag{5}$$

where $\mathbf{0} = (0, \ldots, 0)^T$ and $\mathbf{1} = (1, \ldots, 1)^T$.

Liu, Yao, and Li (2016) show that the preceding QP (5) is equivalent to the following mixed integer linear program (MILP):

$$
\begin{aligned}
\min_{\beta,\mathbf{g},\mathbf{h},\{\gamma_k\}_{k=1}^4,\{\mathbf{z}_k\}_{k=1}^4\in R^m} \quad & -\frac{1}{2}y^T X\beta - \frac{1}{2}n\alpha\lambda\mathbf{1}^T\mathbf{g} - \frac{1}{2}\lambda\mathbf{1}^T\gamma_4 \\
\text{subject to} \quad & X^T X\beta - X^T y + \gamma_1 - \gamma_2 = \mathbf{0} \\
& n\mathbf{g} - \gamma_1 - \gamma_2 = \mathbf{0} \\
& n(\alpha-1)\mathbf{g} + n\mathbf{h} - \gamma_3 + \gamma_4 - n\alpha\lambda 1 = \mathbf{0} \\
& \mathbf{0} \le \gamma_1 \le \mathcal{M}\mathbf{z}_1, \quad \mathbf{0} \le \mathbf{h} - \beta \le \mathcal{M}(1-\mathbf{z}_1) \\
& \mathbf{0} \le \gamma_2 \le \mathcal{M}\mathbf{z}_2, \quad \mathbf{0} \le \mathbf{h} + \beta \le \mathcal{M}(1-\mathbf{z}_2) \\
& \mathbf{0} \le \gamma_3 \le \mathcal{M}\mathbf{z}_3, \quad \mathbf{0} \le \mathbf{g} \le \mathcal{M}(1-\mathbf{z}_3) \\
& \mathbf{0} \le \gamma_4 \le \mathcal{M}\mathbf{z}_4, \quad \mathbf{0} \le \lambda 1 - \mathbf{g} \le \mathcal{M}(1-\mathbf{z}_4) \\
& \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4 \in \{0,1\}^m
\end{aligned} \tag{6}
$$

where $\mathcal{M} > 0$ is a properly large constant.

MCP

In minimax concave penalty (MCP) selection (Zhang 2010), the penalty takes the following form:

$$
P_{\lambda,\alpha}^{\text{mcp}}(\theta) = \begin{cases} \lambda\theta - \frac{1}{2\alpha}\theta^2 & \text{if} \quad 0 \le \theta \le \alpha\lambda \\ \frac{\alpha}{2}\lambda^2 & \text{if} \quad \theta > \alpha\lambda \end{cases} \tag{7}
$$

The MCP estimator $\hat{\beta}^{\text{mcp}}$ solves the following minimization problem:

$$
\hat{\beta}^{\text{mcp}} = \arg\min_{\beta} \left\{ \frac{1}{2}\|\mathbf{y} - \boldsymbol{X}\beta\|_2^2 + n\sum_{j=1}^{m} P_{\lambda,\alpha}^{\text{mcp}}(|\beta_j|) \right\}
$$

The QP reformulation of the preceding MCP problem is given by

$$
\begin{aligned}
\min_{\beta,\mathbf{g},\mathbf{h}\in R^m} \quad & \frac{1}{2}\left[\beta^T X^T X\beta + \frac{n}{\alpha}\mathbf{g}^T\mathbf{g} - \frac{2n}{\alpha}\mathbf{g}^T\mathbf{h}\right] - \mathbf{y}^T X\beta + n\lambda\mathbf{1}^T\mathbf{h} \\
\text{subject to} \quad & -\mathbf{h} \le \beta \le \mathbf{h} \\
& \mathbf{0} \le \mathbf{g} \le \alpha\lambda\mathbf{1}
\end{aligned} \tag{8}
$$

Liu, Yao, and Li (2016) show that the preceding QP (8) is equivalent to the following MILP:

$$
\begin{aligned}
\min_{\beta,\mathbf{g},\mathbf{h},\{\gamma_k\}_{k=1}^4,\{\mathbf{z}_k\}_{k=1}^4\in R^m} \quad & -\frac{1}{2}\mathbf{y}^T X\beta + \frac{1}{2}n\lambda\mathbf{1}^T\mathbf{h} - \frac{1}{2}\alpha\lambda\mathbf{1}^T\gamma_4 \\
\text{subject to} \quad & X^T X\beta - X^T\mathbf{y} + \gamma_1 - \gamma_2 = \mathbf{0} \\
& \frac{n}{\alpha}\mathbf{g} + \gamma_1 + \gamma_2 - n\lambda\mathbf{1} = \mathbf{0} \\
& \frac{n}{\alpha}\mathbf{g} - \frac{n}{\alpha}\mathbf{h} - \gamma_3 + \gamma_4 = \mathbf{0} \\
& \mathbf{0} \le \gamma_1 \le \mathcal{M}\mathbf{z}_1, \quad \mathbf{0} \le \mathbf{h} - \beta \le \mathcal{M}(\mathbf{1} - \mathbf{z}_1) \\
& \mathbf{0} \le \gamma_2 \le \mathcal{M}\mathbf{z}_2, \quad \mathbf{0} \le \mathbf{h} + \beta \le \mathcal{M}(\mathbf{1} - \mathbf{z}_2) \\
& \mathbf{0} \le \gamma_3 \le \mathcal{M}\mathbf{z}_3, \quad \mathbf{0} \le \mathbf{g} \le \mathcal{M}(\mathbf{1} - \mathbf{z}_3) \\
& \mathbf{0} \le \gamma_4 \le \mathcal{M}\mathbf{z}_4, \quad \mathbf{0} \le \alpha\lambda\mathbf{1} - \mathbf{g} \le \mathcal{M}(\mathbf{1} - \mathbf{z}_4) \\
& \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4 \in \{0,1\}^m
\end{aligned} \tag{9}
$$

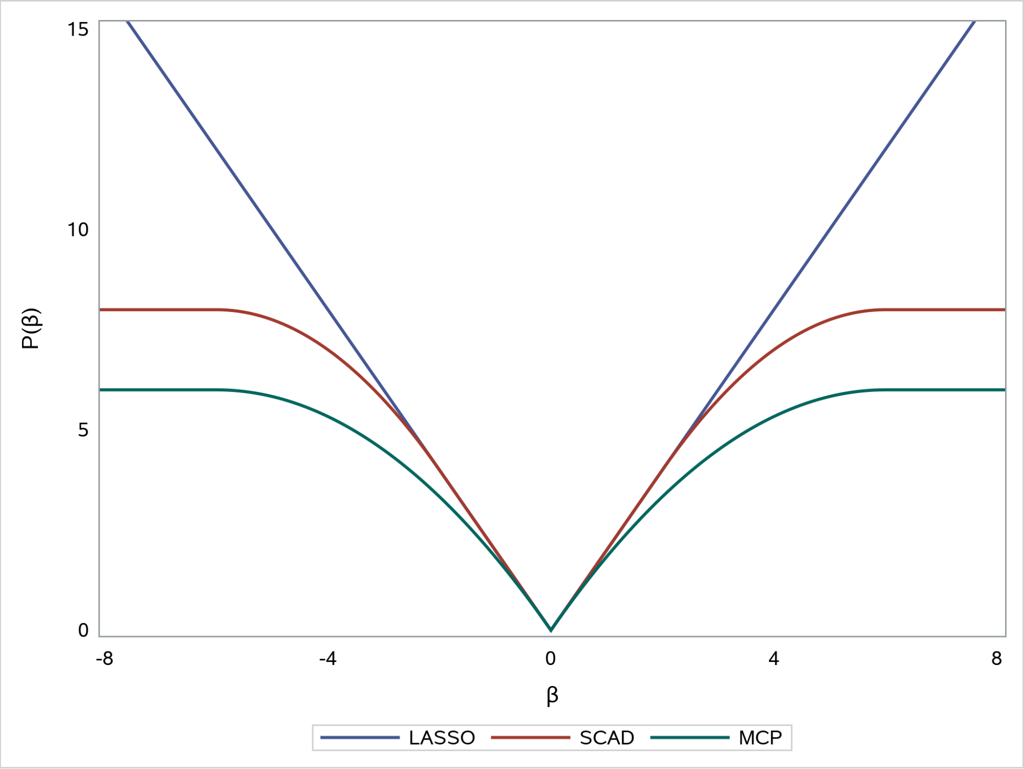
where $\mathcal{M} > 0$ is a properly large constant.

Comparison with LASSO

In the LASSO method, there is only one tuning parameter, λ, whereas the SCAD and MCP penalties have one more tuning parameter, α. LASSO selection tends to overshrink the retained variables. In SCAD and MCP selection, the idea is to let λ and α jointly control the penalty by first suppressing insignificant variables as LASSO does and then tapering off to achieve bias reduction.

Figure 1 plots the LASSO, SCAD, and MCP penalties in a one-dimensional case. You can see that near the origin, the three penalties are almost the same. When the absolute value of β becomes larger, the corresponding folded concave penalty value grows much more slowly than the LASSO penalty value. This illustrates that SCAD and MCP estimates can correct the bias in LASSO estimates that comes from the unboundedness of the ℓ_1 penalty.

Figure 1: Penalty Functions: LASSO, SCAD, and MCP



Variable Selection Procedures in SAS/STAT® and SAS Viya

Both SAS/STAT and SAS Viya provide a rich set of tools for performing variable selection via sequential and penalized methods. Table 2 summarizes the variable selection methods that are supported by the SAS/STAT and SAS Viya procedures.

You can find many useful references in the proceedings of previous SAS® Global Forum conferences that discuss how to perform variable selection by using the procedures shown in Table 3.

Table 2: Variable Selection Methods in SAS/STAT and SAS Viya Procedures

Method	**REG**	**GLMSELECT**	**HPREG**	**REGSELECT**
Forward/Backward/Stepwise	Yes	Yes	Yes	Yes
Best subset	Yes	No	No	Yes
LASSO	No	Yes	Yes	Yes
Adaptive LASSO	No	Yes	Yes	Yes
Elastic net	No	Yes	No	Yes
Adaptive elastic net	No	No	No	Yes
FCP (SCAD/MCP)	No	No	No	Yes

Table 3: References about Variable Selection Methods

Reference	Topic and Procedures
Cohen and Rodriguez (2013)	High-performance statistical modeling (HPGENSELECT, HPLMIXED, HPLOGISTIC, HPNLMOD, HPREG, HPSPLIT)
Günes (2015)	Penalized regression methods for linear models (GLMSELECT)
Rodriguez (2016)	Statistical model building for large, complex data (GLMSELECT, HPGENSELECT, QUANTSELECT, GAMPL, HPSPLIT)
Rodriguez and Cai (2018)	Regression model building for large, complex data (REGSELECT, LOGSELECT, GENSELECT, QTRSELECT, GAMMOD, PHSELECT)
Wang (2020)	Variable selection and convex penalized regression (REGSELECT)

Examples

We use the Sashelp.Baseball data set in the following examples. This data set contains salary and performance information about Major League Baseball players who played at least one game in both the 1986 and 1987 seasons. The salaries are from the 1987 season (Time Inc. 1987), and the performance measures are from the 1986 season (Collier Books 1987). You can load the Sashelp.Baseball data set into your CAS session by using your CAS engine libref named mycas with the following DATA step, after the observations are randomly partitioned into training, validation, and testing groups. Another indicator is added to the data set for further investigation of model selection stability.

```
data work.baseball;
   set sashelp.baseball;
   length Drop $16;
   length Role $16;
   call streaminit(258);
   x = 100*rand('UNIFORM');
   if x<50 then Role = 'TRAIN';
   else if x<80 then Role = 'VAL';
   else Role = 'TEST';
   if 47.5<x<50 then Drop = 'Yes';
   else if 78.5<x<80 then Drop = 'Yes';
   else if x>99 then Drop = 'Yes';
   else Drop = 'No';
   drop x;
run;
data mycas.baseball;
   set work.baseball;
run;
```

Suppose you want to investigate whether you can model the players' salaries from the 1987 season by using performance measures from the 1986 season. You will see how to use the LASSO, adaptive LASSO, and SCAD methods in the REGSELECT procedure to achieve this goal.

LASSO and Adaptive LASSO Selection

You can use the following statements to perform LASSO selection:

```
proc regselect data = mycas.baseball;
   partition roleVar=Role(train='TRAIN' validate='VAL' test='TEST');
   class Div;
   model logSalary = Div nAtBat nHits nHome nRuns nRBI nBB
                      yrMajor crAtBat crHits crHome crRuns crRbi
                      crBB nOuts nAssts nError;
   selection  method = lasso(choose=validate);
run;
```

The PARTITION statement assigns observations to training and validation roles on the basis of the values of the input variable Role. The CHOOSE=VALIDATE option in the SELECTION statement selects the model that yields the smallest average square error (ASE) value for the validation data.

Figure 2 shows that 141 observations are used for model training, 74 observations are used for model validation, and 48 observations are used for model testing. The model that is selected by LASSO is shown in Figure 3.

Figure 2: Basic Information about Baseball Data

Number of Observations Read	322
Number of Observations Used	263
Number of Observations Used for Training	141
Number of Observations Used for Validation	74
Number of Observations Used for Testing	48

Class Level Information		
Class	**Levels**	**Values**
Div	4	AE AW NE NW

Dimensions	
Number of Effects	21
Number of Parameters	21

Figure 3: LASSO Selection of Baseball Data

Model Selected by LASSO

Root MSE	0.57735
R-Square	0.63013
Adj R-Sq	0.60772
AIC	-3.20607
AICC	-1.51377
SBC	-119.66724
ASE (Train)	0.31205
ASE (Validate)	0.30512
ASE (Test)	0.42414

Parameter Estimates		
Parameter	**DF**	**Estimate**
Intercept	1	4.276611
Div_AW	1	-0.058768
nHits	1	0.004108
nRuns	1	0.006835
nBB	1	0.004367
YrMajor	1	0.055307
CrHits	1	0.000012158
CrRuns	1	0.000635
nError	1	-0.001413

The adaptive LASSO selection method assigns weights to each of the parameters in the ℓ_1 penalty. The adaptive LASSO yields consistent estimates of the parameters while retaining the attractive convexity property of the LASSO. By simply adding the ADAPTIVE keyword for the LASSO method in the SELECTION statement, you can use the following statements to perform adaptive LASSO selection:

```
proc regselect data = mycas.baseball;
   partition roleVar=Role(train='TRAIN' validate='VAL' test='TEST');
   class Div;
   model logSalary = Div nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection  method = lasso(adaptive choose=validate);
run;
```

Figure 4: Adaptive LASSO Selection of Baseball Data

Model Selected by Adaptive LASSO

Root MSE	0.60691
R-Square	0.58199
Adj R-Sq	0.56651
AIC	8.04579
AICC	8.88789
SBC	-117.26165
ASE (Train)	0.35267
ASE (Validate)	0.31008
ASE (Test)	0.42103

Parameter Estimates		
Parameter	**DF**	**Estimate**
Intercept	1	4.433354
nHits	1	0.007615
nBB	1	0.001192
CrAtBat	1	0.000095089
CrHits	1	0.000295
CrRuns	1	0.000272

By comparing the results from LASSO selection shown in Figure 3 to the results from adaptive LASSO selection shown in Figure 4, you can see that the model selected by adaptive LASSO has fewer parameters, comparable validation, and a slightly smaller testing ASE.

Furthermore, a model selection method is considered to be unstable if a slight change in the data set leads to a dramatic change in the results. To measure the instability of the selected model, you can randomly remove a small portion (such as 5%) of observations from the data set (by using the DROP= option) and use the remaining data to reselect a new model. You can use the following statements to perform LASSO and adaptive LASSO selection on the reduced data set:

```
proc regselect data = mycas.baseball(where=(Drop='No'));
   partition roleVar=Role(train='TRAIN' validate='VAL' test='TEST');
   class Div;
   model logSalary =  Div nAtBat nHits nHome nRuns nRBI nBB
                        yrMajor crAtBat crHits crHome crRuns crRbi
                        crBB nOuts nAssts nError;
   selection  method = lasso(choose=validate);
run;
```

```
proc regselect data = mycas.baseball(where=(Drop='No'));
   partition roleVar=Role(train='TRAIN' validate='VAL' test='TEST');
   class Div;
   model logSalary =  Div nAtBat nHits nHome nRuns nRBI nBB
                        yrMajor crAtBat crHits crHome crRuns crRbi
                        crBB nOuts nAssts nError;
   selection  method = lasso(adaptive choose=validate);
run;
```

Figure 5 shows that the number of observations that are used for model training, validation, and testing is reduced by 5%.

If you compare Figure 3 to Figure 6, you can see that after randomly removing 5% of the data, in LASSO selection, four predictors (nHits, nRuns, nBB, CrRuns) remain, four previously selected predictors (Div_AW, YrMajor, CrHits, nError) are missing, and two new predictors (CrAtBat, CrRbi) appear. In contrast, if you compare Figure 4 to Figure 7, you can see that the predictors that are selected by the adaptive LASSO method are exactly the same after you randomly remove 5% of the data. This indicates that adaptive LASSO selection is more stable than LASSO selection, partly because the adaptive LASSO tends to select fewer predictors.

Figure 5: Basic Information about Baseball Data with 5% Decrease

Number of Observations Read	302
Number of Observations Used	250
Number of Observations Used for Training	134
Number of Observations Used for Validation	70
Number of Observations Used for Testing	46

Figure 6: LASSO Selection of Baseball Data with 5% Decrease

Model Selected by LASSO

Root MSE	0.67015
R-Square	0.49704
Adj R-Sq	0.47328
AIC	35.54310
AICC	36.69510
SBC	-80.17202
ASE (Train)	0.42564
ASE (Validate)	0.35778
ASE (Test)	0.43986

Parameter Estimates		
Parameter	DF	Estimate
Intercept	1	5.016750
nHits	1	0.002219
nRuns	1	0.004415
nBB	1	0.000441
CrAtBat	1	0.000010348
CrRuns	1	0.000912
CrRbi	1	0.000046024

Figure 7: Adaptive LASSO Selection of Baseball Data with 5% Decrease

Model Selected by Adaptive LASSO

Root MSE	0.61216
R-Square	0.57702
Adj R-Sq	0.56050
AIC	10.33537
AICC	11.22425
SBC	-108.27760
ASE (Train)	0.35795
ASE (Validate)	0.30527
ASE (Test)	0.44227

Parameter Estimates		
Parameter	DF	Estimate
Intercept	1	4.433433
nHits	1	0.008195
nBB	1	0.000366
CrAtBat	1	0.000171
CrHits	1	0.000018715
CrRuns	1	0.000243

SCAD and MCP Selection

You can use the following statements to perform SCAD selection:

```
proc regselect data = mycas.baseball;
   partition roleVar=Role(train='TRAIN' validate='VAL' test='TEST');
   class Div;
   model logSalary = Div nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection  method = scad(choose=validate);
run;
```

Figure 8 shows the basic information about SCAD selection. By default, the MILP solver is used, which means that PROC REGSELECT is solving the mixed integer linear programming problem (6) to perform estimation and selection. You can also specify SOLVER=NLP, which means that the procedure is solving the quadratic programming problem (5). In addition, the searching values of the α series are $\{2.7, 3.7, 4.7, 5.7\}$, and the searching values of the λ series are 10 logarithmically spaced points between the minimum and maximum values.

Figure 9 shows the parameter estimates and the fit statistics of the models that are selected by SCAD. You can see that SCAD outperforms LASSO in the sense that the selected model is sparser and the training, validation, and test ASE values from SCAD are smaller.

Figure 10 shows that the minimal validation ASE is 0.3004 and the corresponding tuning parameter values are $\alpha = 3.74$ and $\lambda = 0.0497985571$.

Figure 8: SCAD Selection Information

Selection Information of SCAD

Selection Information	
Selection Method	SCAD
Solver	MILP
Choose Criterion	Validation ASE
Maximum Alpha	5.7
Minimum Alpha	2.7
Alpha Steps	4
Maximum Lambda	0.356758
Minimum Lambda	0.010306
Lambda Steps	10
Lambda Grid	LOGSPACE

Figure 9: SCAD Selection for Baseball Data

Model Selected by SCAD

Root MSE	0.56699
R-Square	0.63517
Adj R-Sq	0.62166
AIC	-11.13987
AICC	-10.29777
SBC	-136.44731
ASE (Train)	0.30780
ASE (Validate)	0.30043
ASE (Test)	0.42305

Parameter Estimates		
Parameter	DF	Estimate
Intercept	1	3.880058
Div_AW	1	-0.076455
nHits	1	0.009469
nBB	1	0.007454
YrMajor	1	0.105670
nError	1	-0.007611

Figure 10: SCAD Selection Summary

Selection Summary of SCAD

Selection Summary						
Step	Alpha	Lambda	Number of Effects	Validation ASE	Objective Value	Convergence Status
1	2.7	0.3567575275	4	0.4953	-668.3408999	Success
2	2.7	0.2406256213	4	0.3571	-313.8292103	Success
.	.	.	.	.	.	.
	.	.	.	.	.	.
	.	.	.	.	.	.
11	3.7	0.3567575275	4	0.4953	-847.7999658	Success
12	3.7	0.2406256213	5	0.3680	-394.969022	Success
13	3.7	0.1622970369	3	0.3286	-194.4330455	Success
14	3.7	0.1094660164	3	0.3417	-107.6217151	Success
15	3.7	0.0738325786	4	0.3382	-68.71425856	Success
16	3.7	0.0497985571	6	**0.3004***	-51.14739632	Success
17	3.7	0.0335881035	10	0.3356	-44.13832044	Success
18	3.7	0.0226544856	12	0.3290	-41.49368028	Success
19	3.7	0.015279985	16	0.3172	-40.58210989	Success
20	3.7	0.0103060359	17	0.3168	-40.3102798	Success
.	.	.	.	.	.	.
	.	.	.	.	.	.
	.	.	.	.	.	.
39	5.7	0.015279985	12	0.3283	-40.88894314	Success
40	5.7	0.0103060359	17	0.3167	-40.41193515	Success
* Optimal Value of Criterion						

You can also examine the stability of SCAD by randomly removing 5% of the data:

```
proc regselect data = mycas.baseball(where=(Drop='No'));
   partition roleVar=Role(train='TRAIN' validate='VAL' test='TEST');
   class Div;
   model logSalary = Div nAtBat nHits nHome nRuns nRBI nBB
                     yrMajor crAtBat crHits crHome crRuns crRbi
                     crBB nOuts nAssts nError;
   selection  method = scad(choose=validate);
run;
```

By comparing Figure 9 and Figure 11, you can see that there is only one new predictor (Div_AW) after you randomly remove 5% of the data. This indicates that SCAD selection is more stable than LASSO selection.

Figure 11: SCAD Selection of Baseball Data with 5% Decrease

Model Selected by SCAD

Root MSE	0.57547
R-Square	0.62913
Adj R-Sq	0.61161
AIC	-5.28005
AICC	-4.12805
SBC	-120.99517
ASE (Train)	0.31386
ASE (Validate)	0.29326
ASE (Test)	0.44765

Parameter Estimates

Parameter	DF	Estimate
Intercept	1	3.897459
Div_AW	1	-0.080476
Div_NE	1	0.006769
nHits	1	0.010391
nBB	1	0.004800
YrMajor	1	0.105692
nError	1	-0.006983

You can use the following statements to perform MCP selection, by using the MILP solver and NLP solver, respectively:

```
proc regselect data = mycas.baseball;
   partition roleVar=Role(train='TRAIN' validate='VAL' test='TEST');
   class Div;
   model logSalary = Div nAtBat nHits nHome nRuns nRBI nBB
                       yrMajor crAtBat crHits crHome crRuns crRbi
                       crBB nOuts nAssts nError;
   selection  method = mcp(choose=validate solver=MILP);
run;

proc regselect data = mycas.baseball;
   partition roleVar=Role(train='TRAIN' validate='VAL' test='TEST');
```

```
    class Div;
    model logSalary = Div nAtBat nHits nHome nRuns nRBI nBB
                       yrMajor crAtBat crHits crHome crRuns crRbi
                       crBB nOuts nAssts nError;
    selection  method = mcp(choose=validate solver=NLP);
run;
```

From Figure 12 and Figure 14, you can see that the grids of α and λ are exactly the same. The only difference is the solver specification. If you compare the Objective Value column in Figure 13 and Figure 15, you can see that for each fixed set of α and λ, the objective values that are obtained from the MILP solver are always less than the objective values from the NLP solver. The reason is that the MILP solver tries to find the global minimizer, whereas the NLP solver stops after finding a local minimizer. However, if you focus on the Validation ASE column, you can see that for each fixed set of α and λ, the validation ASE values are not overwhelmingly one-sided: sometimes the MILP solver is better, and sometimes the NLP solver is better. Keep in mind that the NLP solver has a much lower computational cost than the MILP solver.

Figure 12: MCP Selection Information with MILP Solver

Selection Information of MCP with MILP Solver

Selection Information	
Selection Method	MCP
Solver	MILP
Choose Criterion	Validation ASE
Maximum Alpha	4.7
Minimum Alpha	1.7
Alpha Steps	4
Maximum Lambda	0.356758
Minimum Lambda	0.010306
Lambda Steps	10
Lambda Grid	LOGSPACE

Figure 13: MCP Selection with MILP Solver Summary

Selection Summary of MCP with MILP Solver

Selection Summary						
Step	Alpha	Lambda	Number of Effects	Validation ASE	Objective Value	Convergence Status
1	1.7	0.3567575275	3	0.4003	-9.904680699	Success
2	1.7	0.2406256213	3	0.3417	-22.27385351	Success
3	1.7	0.1622970369	3	0.3417	-29.83887214	Success
4	1.7	0.1094660164	3	0.3417	-33.28036955	Success
5	1.7	0.0738325786	5	0.3261	-35.29243701	Success
6	1.7	0.0497985571	8	0.3372	-37.07470362	Success
7	1.7	0.0335881035	11	0.3310	-38.40728193	Success
8	1.7	0.0226544856	15	0.3170	-39.23592245	Success
9	1.7	0.015279985	15	0.3170	-39.70524681	Success
10	1.7	0.0103060359	17	0.3168	-39.93914886	Success
.	.	.	.	.	.	.
	.	.	.	.	.	.
	.	.	.	.	.	.
31	4.7	0.3567575275	3	0.4540	-5.416412394	Success
32	4.7	0.2406256213	3	0.3560	-13.44224942	Success
33	4.7	0.1622970369	3	0.3282	-21.48595792	Success
34	4.7	0.1094660164	3	0.3409	-28.21272999	Success
35	4.7	0.0738325786	4	0.3374	-32.60266116	Success
36	4.7	0.0497985571	6	**0.2957***	-34.87146611	Success
37	4.7	0.0335881035	10	0.3333	-36.68145074	Success
38	4.7	0.0226544856	12	0.3280	-38.12567578	Success
39	4.7	0.015279985	16	0.3168	-39.03960057	Success
40	4.7	0.0103060359	17	0.3167	-39.61196978	Success
* Optimal Value of Criterion						

Figure 14: MCP Selection Information with NLP Solver

Selection Information of MCP with NLP Solver

Selection Information	
Selection Method	MCP
Solver	NLP
Choose Criterion	Validation ASE
Maximum Alpha	4.7
Minimum Alpha	1.7
Alpha Steps	4
Maximum Lambda	0.356758
Minimum Lambda	0.010306
Lambda Steps	10
Lambda Grid	LOGSPACE

Figure 15: MCP Selection with NLP Solver Summary

Selection Summary of MCP with NLP Solver

Selection Summary						
Step	**Alpha**	**Lambda**	**Number of Effects**	**Validation ASE**	**Objective Value**	**Convergence Status**
1	1.7	0.3567575275	6	0.4003	-9.904687997	Success
2	1.7	0.2406256213	3	0.3609	-21.19893572	Success
3	1.7	0.1622970369	3	0.3243	-28.33055915	Success
4	1.7	0.1094660164	8	0.3264	-31.4814269	Success
5	1.7	0.0738325786	8	0.3408	-34.254436	Success
6	1.7	0.0497985571	11	**0.2984***	-35.8390344	Success
7	1.7	0.0335881035	13	0.3201	-38.00890719	Success
8	1.7	0.0226544856	18	0.3199	-39.06713142	Success
9	1.7	0.015279985	14	0.3293	-39.48039736	Success
10	1.7	0.0103060359	18	0.3125	-39.7931101	Success
.	.	.	.	.	.	.
	.	.	.	.	.	.
	.	.	.	.	.	.
31	4.7	0.3567575275	3	0.4540	-5.416459255	Success
32	4.7	0.2406256213	3	0.3560	-13.44224949	Success
33	4.7	0.1622970369	3	0.3457	-21.16069948	Success
34	4.7	0.1094660164	3	0.3409	-28.21273013	Success
35	4.7	0.0738325786	5	0.3700	-31.38563103	Success
36	4.7	0.0497985571	10	0.3071	-34.66186064	Success
37	4.7	0.0335881035	10	0.3037	-36.38336837	Success
38	4.7	0.0226544856	11	0.3038	-37.62750211	Success
39	4.7	0.015279985	15	0.3381	-38.92168992	Success
40	4.7	0.0103060359	17	0.3165	-39.58087726	Success
* Optimal Value of Criterion						

Conclusion

This paper summarizes the penalized variable selection methods—in particular, folded concave penalized (FCP) regression—and the SAS/STAT and SAS Viya procedures that use these methods. It provides several examples to demonstrate how you can use the REGSELECT procedure, available in SAS Viya, to perform variable selection by using the penalized regression methods. Although the results of the examples in the paper show that the SCAD method performs slightly better than the LASSO method, keep in mind that in practice, no single method consistently outperforms the rest. Furthermore, there are no universally best defaults for the tuning parameters in penalized regression methods. However, depending on your goal, an informed and judicious choice of these features can lead to models that have better predictive accuracy or models that are more interpretable. You should also experiment with different combinations of the options available in these procedures to learn more about their behavior.

References

Bickel, P. (2008). "Discussion of Sure Independence Screening for Ultrahigh Dimensional Feature Space by Fan and Lv." *Journal of the Royal Statistical Society, Series B* 70:883–884.

Cohen, R., and Rodriguez, R. N. (2013). "High-Performance Statistical Modeling." In *Proceedings of the SAS Global Forum 2013 Conference*. Cary, NC: SAS Institute Inc. http://support.sas.com/resources/papers/proceedings13/401-2013.pdf.

Collier Books (1987). *The 1987 Baseball Encyclopedia Update*. New York: Macmillan.

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.

Günes, F. (2015). "Penalized Regression Methods for Linear Models in SAS/STAT." In *Proceedings of the SAS Global Forum 2015 Conference*. Cary, NC: SAS Institute Inc. http://support.sas.com/rnd/app/stat/papers/2015/PenalizedRegression_LinearModels.pdf.

Liu, H., Yao, T., and Li, R. (2016). "Global Solutions to Folded Concave Penalized Nonconvex Learning." *Annals of Statistics* 44:629–659.

Marquardt, D. W., and Snee, R. D. (1975). "Ridge Regression in Practice." *American Statistician* 29:3–20.

Rodriguez, R. N. (2016). "Statistical Model Building for Large, Complex Data: Five New Directions in SAS/STAT Software." In *Proceedings of the SAS Global Forum 2016 Conference*. Cary, NC: SAS Institute Inc. http://support.sas.com/resources/papers/proceedings16/SAS4900-2016.pdf.

Rodriguez, R. N., and Cai, W. (2018). "Regression Model Building for Large, Complex Data with SAS Viya Procedures." In *Proceedings of the SAS Global Forum 2018 Conference*. Cary, NC: SAS Institute Inc. https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2018/2033-2018.pdf.

Tibshirani, R. (1996). "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society, Series B* 58:267–288.

Time Inc. (1987). “What They Make.” *Sports Illustrated* (April 20): 54–81.

Wang, Y. (2020). “A Survey of Methods in Variable Selection and Penalized Regression.” In *Proceedings of the SAS Global Forum 2020 Conference*. Cary, NC: SAS Institute Inc. https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4287-2020.pdf.

Zhang, C.-H. (2010). “Nearly Unbiased Variable Selection under Minimax Concave Penalty.” *Annals of Statistics* 38:894–942.

Zou, H. (2006). “The Adaptive Lasso and Its Oracle Properties.” *Journal of the American Statistical Association* 101:1418–1429.

Zou, H., and Hastie, T. (2005). “Regularization and Variable Selection via the Elastic Net.” *Journal of the Royal Statistical Society, Series B* 67:301–320.

Zou, H., and Zhang, H. H. (2009). “On the Adaptive Elastic-Net with a Diverging Number of Parameters.” *Annals of Statistics* 37:1733–1751.