Combining l^1 and Higher Order l^p Penalization in Regression Models

Austin David Brown

University of Minnesota

Abstract

We create the package Penalized Regression on Steroids (**pros**) to combine l^1 penalized regression with higher order l^p penalizations built upon the Elasticnet idea. The package is able to fit regression models with penalizations ranging from l^1 to l^{10}.

Keywords: Penalized Regression, C++, Python, R.

1. Introduction

In statistics and probability theory it is common to impose moment assumptions on a random variable $X : \Omega \to \mathbb{R}^n$ such as $E(|X|^k) < \infty$ for $k \in \mathbb{R}$. These constraints correspond to the L^p spaces which allow control over the width and the height of such random variables. We believe that analogous tools should be available for scientists tackling real problems. The problem may also be motivated geometrically. Consider for example an Elasticnet penalty $Q(x) = \frac{1}{2}|x| + \frac{1}{2}|y| + \frac{1}{2}|x|^2 + \frac{1}{2}|y|^2 \leq 1$ shown on the left and a new penalty $P(x) = \frac{1}{2}|x| + \frac{1}{2}|y| + \frac{1}{2}|x|^4 + \frac{1}{2}|y|^4 \leq 1$ shown on the right.
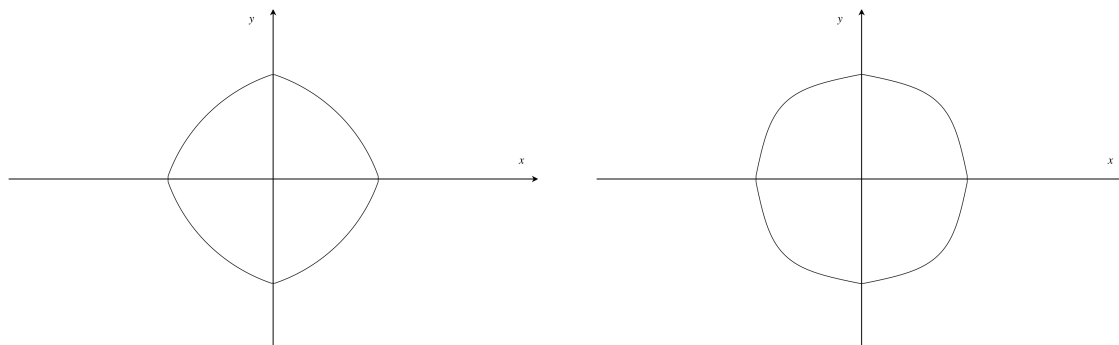
Figure 1: A particular ElasticNet penalty shown on the left and a new penalty shown on the right shown to "bow" out the penalization while retaining convexity.

A scientist may want the ability to control the height and width of the penalization while still retaining sparse solutions. In this project, we build the Penalized Regression on Steroids (**pros**) package available to the R programming language to expand upon the idea of ElasticNet by Zou and Hastie (2005), which we believe approximately encapsulates this idea.

2. Penalized Regression Implementation

In this section, we discuss how the package was implemented. In particular, the construction of the problem and the algorithms used. Let

$$L_\lambda(\beta) = \frac{1}{2} \|y - X\beta\|_2^2 + \lambda P(\beta)$$

be the objective to minimize with penalty

$$\lambda P(\beta) = \lambda \alpha_0 \|\beta\|_1 + \lambda \sum_{k=1}^{5} \alpha_k \|\beta\|_{2k}^{2k}$$

where $y \in \mathbb{R}^n$, $X \in M_{n \times p}(\mathbb{R})$, and $\beta \in \mathbb{R}^p$, $\lambda \in \mathbb{R}_+$ and α's are convex combinations. An important property of this penalty is convex and completely separable which allows for coordinate-wise optimization. In **glmnet** by Simon, Friedman, Hastie, and Tibshirani (2011), the authors use coordinate descent. Further, with the Elasticnet penalty by Zou and Hastie (2005), the coordinate wise minimization yields an analytic, closed form solution at each iteration avoiding any need for a line search or step size. The penalty that we propose lacks an analytic solution and thus a new algorithm is needed.

2.1. Optimization Algorithms and Implementation

The first algorithm utilized was the subgradient coordinate algorithm shown below.

Algorithm 1: Subgradient Coordinate Algorithm

Choose $\beta^0 \in \mathbb{R}^p$, tolerance $\delta > 0$, $R > 0$, and maximum iterations N.
Set $k \leftarrow 0$
Set the step size $h \leftarrow \frac{R}{\sqrt{1+N}}$
repeat
    Permute $I = \{1, \ldots, p\}$
    for $i \in I$ **do**
        $\beta_i^{k+1} \leftarrow \beta_i^k - hg^i$ where $g_i^k \in (\partial L_\lambda(\beta^k))_i$
        $k \leftarrow k + 1$
    end
until *Until the objective difference is less than δ*;

The drawbacks of this algorithm include lack of the descent property, no good stopping criterion, and many possible choices for the subgradient. The step size is optimal and chosen due to Nesterov (1998) with a worst case convergence rate of $O(\frac{1}{\sqrt{k}})$. Ultimately, no line search can be implemented and the step size must be tuned by the user, which may be difficult for many users. The stopping criterion is also expensive at $O(n^2)$ flops at each iteration.

Due to the separability of the penalization, a better algorithm is proximal gradient coordinate descent shown below.

Algorithm 2: Proximal Gradient Coordinate Descent

Choose $\beta^0 \in \mathbb{R}^p$ and tolerance $\delta > 0$;
Set $k \leftarrow 0$
repeat
    Randomly permute $I = \{1, \ldots, p\}$
    for $i \in I$ **do**
        Set the step size $h_i > 0$ or use line search.
        $\beta_i^{k+1} \leftarrow (\mathbf{prox}_{h_i L})_i (\beta_i^k - h^k \langle X_i, y - X\beta \rangle)$
        $k \leftarrow k + 1$
    end
until *Until the Moreau-Yoshida mapping $M_{h_k, f} < \delta$*;

The major benefit of this algorithm is that it is a descent method and thus line search may be implemented. Moreover, the stopping criterion is cheap to compute at $O(p)$ flops. The Moreau-Yoshida map recovers the worst case convergence of $O(\frac{1}{k})$ generally and coordinate descent algorithms are explored by Nesterov (2010).

2.2. Cross-Validation Algorithms and Implementation

In order to tune the Lagrangian dual variable λ, K-fold cross validation is implemented. To improve the speed of cross-validation, a warm start algorithm is used defined below.

Algorithm 3: Warm Start Cross-Validation

Choose a sequence of Langrangian dual variables $\lambda_1, \ldots, \lambda_N$, and initial value β^0.
Order $\lambda_{(1)}, \ldots, \lambda_{(N)}$ descending.
$\beta^{Warm} \leftarrow \beta^0$
for $k \in 1, \ldots, N$ **do**
    $\beta^k \leftarrow$ by Cross-Validation with $\lambda_{(k)}$ warm started with β^{Warm}.
    $\beta^{Warm} \leftarrow \beta^k$
end

A difficult problem in general is choosing the sequence of λ's. In **glmnet**, the default is to choose a sequence of length 100 starting from the first λ_{max} where β is completely zero to the lower value $.001\lambda_{max}$ on a log-scale. We believe, the lower λ value in the sequence is the real difficulty and this implementation does not address this. Due to time, we implement a similar default choice. We choose the lower $\lambda = cp$ where $c > 0$ is a tuning parameter and then create a sequence of length 100 from this lower value. The default choice comparisons are explored in a subsequent section.

2.3. Discussion

One source of serious confusion for the author that soaked up a lot of time was that **glmnet** uses a transformation for their λ values and these do not correspond with the typical Lasso model. Thus, direct comparisons are extremely confusing.

3. The Penalized Regression on Steroids Package

3.1. Package Architecture

The package **pros** is built using C++ and the Eigen library by Guennebaud, Jacob *et al.* (2010) is utilized for fast matrix computations. This is analogous to using Fortran with LA-PACK by Anderson, Bai, Dongarra, Greenbaum, McKenney, Du Croz, Hammarling, Demmel, Bischof, and Sorensen (1990). The **pros** library is not dependent on R. To interact with R, 2 interfacing layers are needed: an R to C interface and an R program defining user callable functions. The benefit is that many other popular programming languages can be interfaced easily. The following diagram illustrates this idea showing that the **pros** package is more general than just an R package.
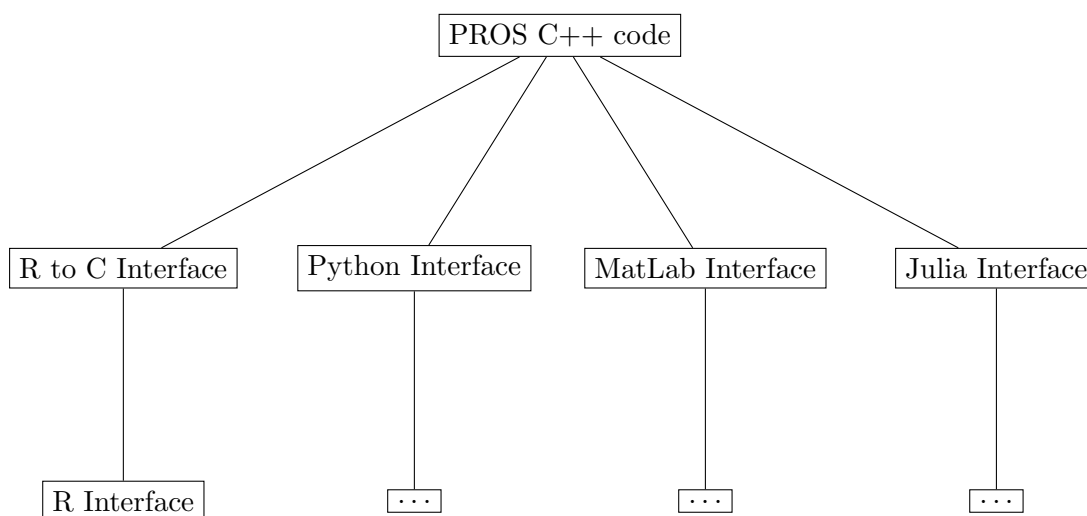

Figure 2: Illustrates interfacing the **pros** package with other languages.

The R interface in its simplest form is comprised of 2 functions. A single fitting function with prediction

```
> fit <- pros(X, y, alpha)
> predict(fit, X)
```

and a cross-validation function with prediction

```
> cv <- cv.pros(X, y, alpha)
> predict(cv, X)
```

We illustrate the usage of these functions in the following sections.

3.2. The Regression Fitting Function

The **pros** function is used to fit a single regression model with a specified penalization. The signature of this function at the time of this paper is as follows:

- `X` is an $n \times m$-dimensional matrix of the data.

- `y` is an n-dimensional vector of response values.

- `alpha` is a 6-dimensional vector of the convex combination corresponding to the penalization:

  - α_1 is the l^1 penalty.
  - α_2 is the l^2 penalty.
  - α_3 is the l^4 penalty.
  - α_4 is the l^6 penalty.
  - α_5 is the l^8 penalty.
  - α_6 is the l^{10} penalty.

- `lambdas` is a vector of dual penalization values to be evaluated.

- `step_size` is a tuning parameter defining the step size. Larger values are more aggressive and smaller values are less aggressive.

- `algorithm` is the optimization algorithm

  - `proximal_gradient_cd` uses proximal gradient coordinate descent.
  - `subgradient_cd` uses subgradient coordinate descent.

- `max_iter` is the maximum iterations the algorithm will run regardless of convergence.

- `tolerance` is the accuracy of the stopping criterion.

- `random seed` is the random seed used in the algorithms.

3.3. The Cross-Validation Function

The **cv.pros** function is used for cross-validation. The signature of this function at the time of this paper is as follows:

- `X` is an $n \times m$-dimensional matrix of the data.

- `y` is an n-dimensional vector of response values.

- `alpha` is a 6-dimensional vector of the convex combination corresponding to the penalization:

  - α_1 is the l^1 penalty.
  - α_2 is the l^2 penalty.
  - α_3 is the l^4 penalty.
  - α_4 is the l^6 penalty.
  - α_5 is the l^8 penalty.
  - α_6 is the l^{10} penalty.

- `lambda` is the Lagrangian dual penalization parameter.

- `step_size` is a tuning parameter defining the step size. Larger values are more aggressive and smaller values are less aggressive.

- `algorithm` is the optimization algorithm

  - `proximal_gradient_cd` uses proximal gradient coordinate descent.
  - `subgradient_cd` uses subgradient coordinate descent.

- `max_iter` is the maximum iterations the algorithm will run regardless of convergence.

- `tolerance` is the accuracy of the stopping criterion.

- `random seed` is the random seed used in the algorithms.

4. Numerical Examples

4.1. The Boston Housing Dataset

The Boston Housing dataset is popular from Harrison and Rubinfeld (1978). There are 13 predictors and the response is the median value of owner-occupied homes in $1000's. The analysis is setup in the following way:

- The data is randomly split into a training set with 404 observations and a test set with 102 observations.

- The predictor data is standardized.

- The **glmnet** library and the **pros** library were used for fitting.

- 10-fold cross-validation was used to tune the Lagrangian dual variable λ.

- The source code for this analysis is openly available at the **pros** repository available here Brown (2018).

We compare default settings between the libraries

Penalty	Tuning	Test Mean Squared Error
Lasso (glmnet)	$\alpha = 1$	TODO
ElasticNet (glmnet)	$\alpha = 1/2$	TODO
Lasso (pros)	$\alpha = 1$	
ElasticNet (pros)	$\alpha = 1/2$	
4th Moment (pros)	$\alpha = 1/2$	

Next, we compare tuned results using pros

Penalty	Tuning	Test Mean Squared Error
ElasticNet (pros)	$\alpha = .98$	TODO
4th Moment (pros)	$\alpha = 0.96$	TODO
10th Moment (pros)	$\alpha = 0.96$	TODo

4.2. The Prostate Dataset Analysis

This is a popular dataset from A. Stamey, N. Kabalin, E. Mcneal, M. Johnstone, Freiha, A. Redwine, and Yang (1989) and analyzed in the ElasticNet paper Zou and Hastie (2005). There are 8 predictors and the response is the log of the prostate specific antigen. The analysis is setup in the following way:

- The data is split into a training set with 67 observations and a test set with 30 observations.

- The predictor data is standardized.

- The **glmnet** library and the **pros** library were used for fitting.

- 10-fold cross-validation was used to choose the Lagrangian dual variable λ.

- The source code for this analysis is openly available at the **pros** repository available here Brown (2018).

We compare default settings between the libraries

Penalty	Tuning	Test Mean Squared Error
Lasso (glmnet)	$\alpha = 1$	0.4443432-0.4646501
ElasticNet (glmnet)	$\alpha = 1/2$	0.4539684-0.4554527
Lasso (pros)	$\alpha = 1$	
ElasticNet (pros)	$\alpha = 1/2$	
4th Moment (pros)	$\alpha = 1/2$	

Next, we compare tuned results using pros

Penalty	Tuning	Test Mean Squared Error
ElasticNet (pros)	$\alpha = .98$	0.4442702
4th Moment (pros)	$\alpha = 0.96$	0.4441759
10th Moment (pros)	$\alpha = 0.96$	0.4441322

References

A Stamey T, N Kabalin J, E Mcneal J, M Johnstone I, Freiha F, A Redwine E, Yang N (1989). "Prostate Specific Antigen in the Diagnosis and Treatment of Adenocarcinoma of the Prostate. II. Radical Prostatectomy Treated Patients." *The Journal of urology*, **141**, 1076–83. doi:10.1016/S0022-5347(17)41175-X.

Anderson E, Bai Z, Dongarra J, Greenbaum A, McKenney A, Du Croz J, Hammarling S, Demmel J, Bischof C, Sorensen D (1990). "LAPACK: A Portable Linear Algebra Library for High-performance Computers." In *Proceedings of the 1990 ACM/IEEE Conference on Supercomputing*, Supercomputing '90, pp. 2–11. IEEE Computer Society Press, Los Alamitos, CA, USA. ISBN 0-89791-412-0. URL http://dl.acm.org/citation.cfm?id=110382.110385.

Brown   A   (2018).   "Penalized   Regression   on   Steroids."
   http://github.com/austindavidbrown/pros.

Guennebaud G, Jacob B, *et al.* (2010). "Eigen v3." http://eigen.tuxfamily.org.

Harrison D, Rubinfeld D (1978). "Hedonic prices and the demand for clean air." *J. Environ. Economics and Management.*

Nesterov Y (1998). "Introductory Lectures On Convex Programming."

Nesterov Y (2010). "Efficiency of coordinate descent methods on huge-scale optimization problems."

Simon N, Friedman J, Hastie T, Tibshirani R (2011). "Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent." *Journal of Statistical Software*, **39**(5), 1–13. URL http://www.jstatsoft.org/v39/i05/.

Zou H, Hastie T (2005). "Regularization and variable selection via the Elastic Net." *Journal of the Royal Statistical Society, Series B*, **67**, 301–320.

Affiliation:

Austin David Brown
University of Minnesota
Department of Statistics (Graduate Student)
E-mail: Brow5079@umn.edu