Combining l^1 Penalization with Higher Moment Feasible Sets in Regression Models: a STAT8053 Project

Austin David Brown

November 30, 2018

TOOD library may crash by design
TODO talk about convergence guarantee

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. This can be interpreted as imposing "stability" on the random variable X. If statisticians so freely impose such constraints then we should build a tool to allow scientists and researchers to impose such constraints on their real problems. In this project, we build a package implements this idea. The goal is not to create the best predicting method, but instead build a tool that will allow researchers to explore stability and what a "stable" solution to their problem may look like.

I was genuinely curious as to what happens if the Elasticnet from Zou and Hastie [6] had higher moments. For simplicity, we consider exactly this with the penalty

$$P(\lambda) = \lambda\alpha_1 \|\beta\|_1 + \lambda \sum_{k=1}^{5} \alpha_k \|\beta\|_{2k}^{2k}$$

with $\sum_{k=1}^{10} \alpha_k = 1$ and 10 is chosen for no special reason. This has a geometric interpretation. Consider a particular Elasticnet penalty

$$1/2|x| + 1/2|y| + 1/2|x|^2 + 1/2|y|^2$$

shown on the left and a new penalization

$$1/2|x| + 1/2|y| + 1/2|x|^4 + 1/2|y|^4$$

shown on the right. It seems reasonable that a scientist or researcher may want to "bow" out the feasable set in a geometric sense.

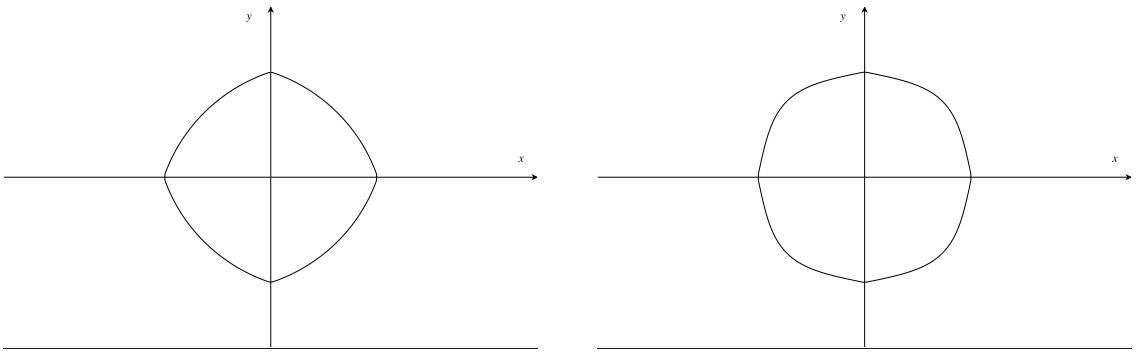

Figure 1: A particular ElasticNet penalty shown on the left and a new penalty shown on the right shown to "bow" out the feasible set.

Alternatively, the researcher wants to impose "stability" into their solutions by restricting the feasible set. This is analagous to moment conditions in some sense since the researcher wants to control the sparsity along with the height and width of their solution.

Another penalty I would like to explore in the future is combining the $\|\cdot\|_1$ and $\|\cdot\|_\infty$ as

$$P'(\lambda) = \lambda\alpha_1 \|\beta\|_1 + \lambda\alpha_2 \|\beta\|_\infty$$

This is be shown in the figure below.

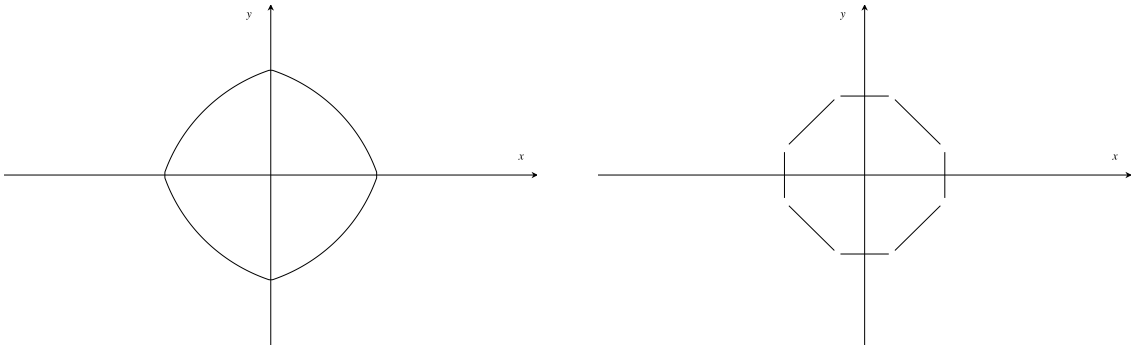

Figure 2: Elasticnet on the left and l1 and infinity norm on the right

I believe this can be solved with the ADMM algorithm, but I ran out of time.

2 Implementation

In this section, we discuss how the package was implemented. In the glmnet paper [5], Friedman, Hastie and Tibshirani use coordinate descent. With the elasticnet [6] penalty,

the coordinate wise minimization yields an analytic, closed form solution and line search or step size rules are not needed. With the penalty that we propose

$$P(\lambda) = \lambda\alpha_1 \|\beta\|_1 + \lambda \sum_{k=1}^{5} \alpha_k \|\beta\|_{2k}^{2k}$$

will not have an analytic solution and a new algorithm is needed. This penalization also not Lipschitz continuous and makes convergence analysis more difficult. An important point is that the l^1 penalty is not differentiable, but it is separable and so is this penalty and so coordinate-wise optimization will hold. Let

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

be the objective with $y \in \mathbb{R}^n$, $X \in M_{n\times p}(\mathbb{R})$ centered, and $\beta \in \mathbb{R}^p$. The first algorithm used for this problem was the subgradient coordinate method shown below.

Algorithm 1: Subgradient Coordinate Method

Choose $\beta^0 \in \mathbb{R}^p$ and tolerance $\delta > 0$;
Set $k \leftarrow 0$
repeat
  Set the step size $h^k \leftarrow \frac{R}{\sqrt{1+k}}$ for some $R > 0$ or use a constant step size.
  Permute $I = \{1, \ldots, p\}$
  for $i \in I$ **do**
    $\beta_i^{k+1} \leftarrow \beta_i^k - h^i g^i$ where $g^i \in (\partial L)_i$
  end
  $k \leftarrow k + 1$
until *Until the loss difference ΔL is less than δ*;

This algorithm is not a descent method, there is no good stopping criterion, and the non-differentiability is problematic to line search. The diminishing step size was chosen due to Nesterov's analysis in [7]. The stopping criterion is also expensive at $O(n^2)$ flops. In our implementation, we experiences solutions without sparsity, but instead with very small values, so a thresholding function is needed to produce sparsity.
A better algorithm we discovered later was 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
  Set the step size $h^k > 0$ with diminishing or line search.
  Randomly permute $I = \{1, \ldots, p\}$
  for $i \in I$ **do**
    $\beta_i^{k+1} \leftarrow (\mathbf{prox}_{h^k L})_i(\beta_i^k - h^k \langle X_i, y - X\beta \rangle)$
  end
  $k \leftarrow k + 1$
until *Until the Moreau-Yoshida mapping* $M_{h_k,f} < \delta$;

An important point is because the objective and penalization are separable, the proximal mapping is

$$\mathbf{prox}_L(x) = (\mathbf{prox}_L(x_i))_i) \,.$$

This is a descent method, line search can be done, and the stopping criterion is cheap to compute at $O(p^2)$. Further, the convergence theory is the same as coordinate descent due to the Moreau-Yoshida regularization. The random permutation is done so that convergence analysis can be rigorously established.
To improve the speed of cross-validation, warm starting (See [5]) was used shown 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

This can provide substantial speed improvements in practice.
For speed, $C{+}{+}$ was used along with the Eigen library [4] for matrix computations. This is analogous to using Fortran with LAPACK. To interact with R, 2 interfacing layers need to be created: an R to C interface and an R script that the user calls functions from. The benefit is that any other language can be interfaced easily. The following diagram illustrates the idea.

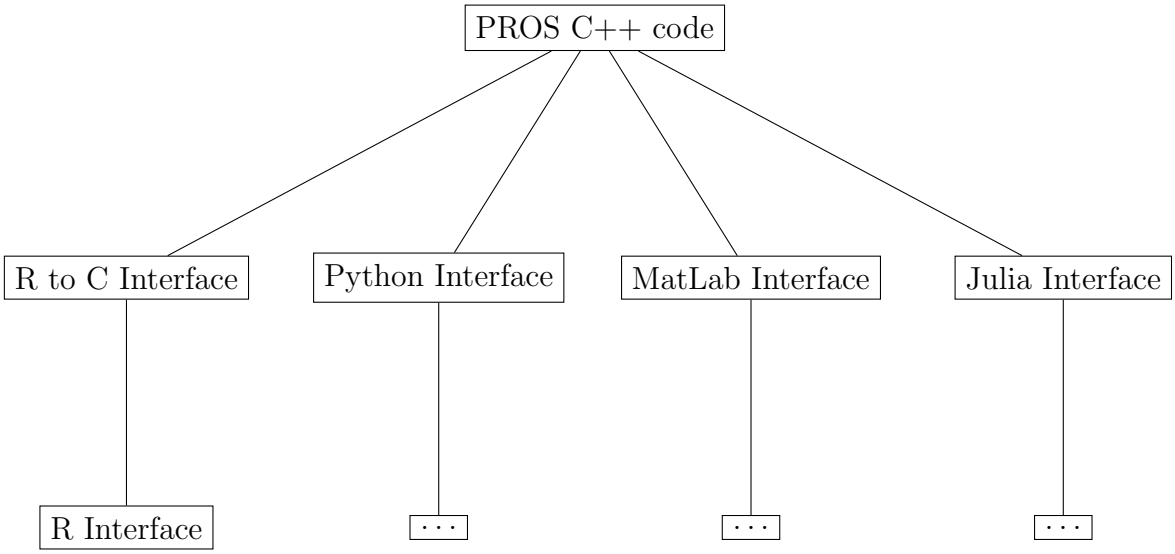

Figure 3: TODO

The R interface is purposefully simple and similar to popular packages such as **glmnet** [5]. There are single fitting and prediction functions

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

and warm-start cross-validation and prediction functions

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

The package has no dependencies and is available here for download here github.com/austindavidbrown/pros.

References

[1] PROS. github.com/austindavidbrown/pros

[2] Neal Parikh and Stephen Boyd. 2014. Proximal Algorithms. Found. Trends Optim. 1, 3 (January 2014), 127-239. DOI=10.1561/2400000003 http://dx.doi.org/10.1561/2400000003

[3] Stephen J. Wright. 2015. Coordinate descent algorithms. Math. Program. 151, 1 (June 2015), 3-34. DOI=10.1007/s10107-015-0892-3 http://dx.doi.org/10.1007/s10107-015-0892-3

[4] Guennebaud, Gaël (2013). Eigen: A C++ linear algebra library (PDF). Eurographics/CGLibs.

[5] Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/.

[6] 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.

[7] Yurii Nesterov. 2014. Introductory Lectures on Convex Optimization: A Basic Course (1 ed.). Springer Publishing Company, Incorporated.