Combining l^1 Penalization with Higher Moment Constraints in Regression Models

Austin David Brown

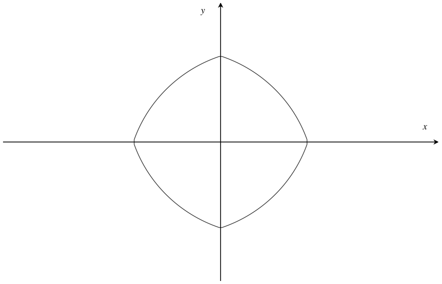
December 15, 2018

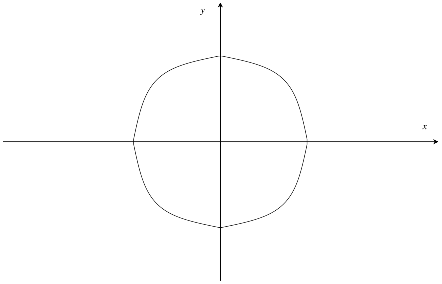
Motivation

- 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. Thus, these norms give us some ”statistics” about the random variable.
- If statisticians so freely impose such constraints then we should build a tool to allow scientists and researchers to impose similar constraints on their real problems.
- In this project, I built an R package to expand upon the idea of ElasticNet [8], which I think approximately encapsulates this idea.

Geometric Motivation

Consider for example an Elasticnet [8] 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. Extending this idea, 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.





It is possible that a scientist or researcher may want the option to "bow" out the feasible set even more or maybe Elasticnet [8] excludes a good solution.

The Setup

Define a new penalty to try to impose more "stability" for scientists and researchers (try to extend the Elasticnet [8] idea). Let

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

with penalty

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

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 or use separate tuning parameters instead.

An important property is that this is a convex and completely separable penalty. This allows us to use coordinate descent.

Algorithm Implementation 1

Algorithm 1: Subgradient Coordinate Method

```
Choose $\beta^0 \in \mathbb{R}^p$ and tolerance $\delta > 0$;
Set $k \leftarrow 0$
repeat
    Randomly choose $i_k \in \{1, \dots, p\}$
    $h_i^k \leftarrow \frac{R}{\sqrt{1+k}}$ for some $R > 0$ or use $\frac{R}{\sqrt{1+N}}$ where $N$ is the length of the
      algorithm.
    $\beta_{i_{k+1}} \leftarrow \beta_{i_k} - h^k g^{i_k}$ where $g^{i_k} \in (\partial L)_{i_k}$
    $k \leftarrow k + 1$
until Until the loss difference $\Delta L$ is less than $\delta$;
```

Drawbacks

- Not a descent method, so you cannot do line search.
- No good stopping criterion.
- There are many choices for the subgradient.
- Tends to produce really small values instead of truly sparse solutions.

A Better Algorithm

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 $h^k > 0$ constant, diminishing or by line search.
    Randomly choose $i_k \in \{1, \dots, p\}$
    $\beta_i^{k+1} \leftarrow (\mathbf{prox}_{h^k L})_i(\beta_i^k - h^k \langle X_i, y - X\beta \rangle)$
    $k \leftarrow k + 1$
until Until the Moreau-Yoshida mapping $M_{h_k, f} < \delta$;
```

Benefits

- A descent method, so line search can be implemented.
- Can be accelerated.
- Good stopping criterion at approximately $O(p)$ flops.
- Convergence theory is unchanged from differentiable functions.
- Coordinate descent theory for differentiable functions is still being actively developed.

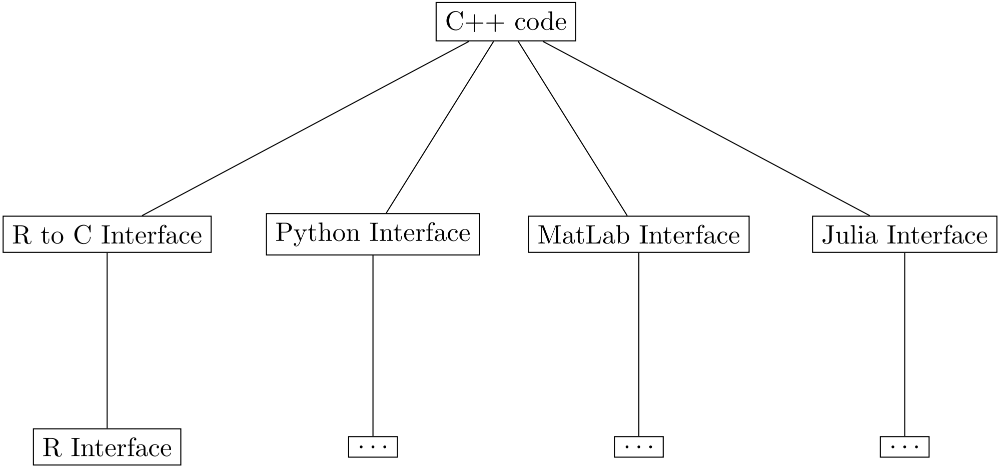
Cross-Validation Implementation

Algorithm 3: Warm Start Cross-Validation

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

Package Architecture

Written entirely in C++ for speed and uses Eigen [6] for linear algebra. It can be interfaced to many other popular languages.



Penalized Regression on Steroids

Installation:

```
> devtools::install_github("austindavidbrown/pros/R-package")
```

A single fit function with prediction

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

A cross-validation fit function with prediction

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

There is even a reference manual.

Yes, I know the name is bad.

The Boston Housing Dataset Analysis

The Boston Housing dataset is popular from Harrison et al. [1]. There are 13 predictors and the response is the median value of owner-occupied homes in \$1000s.

- The data is randomly split into a training set with 404 observations and a test set with 102 observations.
- The predictor data is standardized.
- Basic 10-fold cross-validation and no special tuning.
- The **glmnet** [7] library was used for the Lasso and the ElasticNet.
- **pros** [3] was used to fit the new penalty.
- The random seed was set to 8989.
- The code is available to you at the **pros** repository [3].

The Boston Housing Dataset Results

Penalty	**Tuning**	**Test MSE**
Lasso Penalty (glmnet)	$\alpha = (1, 0)$	28.52329
ElasticNet Penalty (glmnet)	$\alpha = (1/2, 1/2)$	27.46224
New Penalty (pros)	$\alpha = (1/2, 0, 1/2, 0, 0, 0)$	26.31973
New Penalty (pros)	$\alpha = (1/2, 0, 0, 0, 0, 1/2)$	26.3852
Tuned ElasticNet Penalty (glmnet)	$\alpha = (.25, .75)$	26.80933
New Penalty (pros)	$\alpha = (.25, 0, .75, 0, 0, 0)$	26.25065
Tuned New Penalty (pros)	$\alpha = (.1, 0, .9, 0, 0, 0)$	26.21775

The Boston Housing Dataset Notes

- The un-tuned 4th moment Penalty and un-tuned Elasticnet both removed age, but have different solutions otherwise.
- The 10th moment penalty produces no sparsity.

The Prostate Cancer Dataset Analysis

This is a popular dataset from Stamey et al. [2] and analyzed in the ElasticNet paper by Zou and Hastie [8]. There are 8 predictors and the response is the log of the prostate specific antigen.

- 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** [7] library was used to fit and tune the Lasso and a naive ElasticNet.
- **pros** [3] was used to fit the new penalty.
- 10-fold cross-validation and crude manual tuning were used.
- The random seed was set to 8989.
- The code is available to you at the **pros** repository [3].

The Prostate Cancer Dataset Results

Penalty	**Tuning**	**Test MSE**
Lasso Penalty (glmnet)	$\alpha = (1, 0)$	0.4443432-0.4646501
"Tuned" ElasticNet Penalty (glmnet)	$\alpha = (1/2, 1/2)$	0.4539684-0.4554527
ElasticNet (pros)	$\alpha = (1/2, 1/2, 0, 0, 0, 0)$	0.5255658
New Penalty (pros)	$\alpha = (1/2, 0, 1/2, 0, 0, 0)$	0.526071
New Penalty (pros)	$\alpha = (1/2, 1/4, 1/4, 0, 0, 0)$	0.5256688

The Prostate Cancer Dataset Results

- Seems Elasticnet [8] is better for this dataset.
- I am very confused as to why my results don't match and why glmnet gives varying results.
- I think it should preform similar. However, the solutions do have different sparsity, which is interesting.
- With manual tuning I do way better, which is strange. Maybe my CV is wrong? Maybe I broke something when I changed the random number generator.

Conclusion

- I think that there are many ”good” solutions to real problems and we can get ”statistics” on these solutions by imposing more norms on the feasible sets building upon the Elasticnet [8] idea.

- I would like to explore even more sparsity inducing penalizations such as

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

$$\lambda P''(\beta) = \lambda\alpha_1 \left\|\beta\right\|_1 + \lambda \sum_{k=2}^{10} \alpha_k \left\|\beta\right\|_k$$

Things to Address

- Currently, you have to tune a step size. I need to implement a fast line search. This is not too difficult actually just ran out of time.
- Add logistic regression
- Look into convergence analysis.
- C++ offers many random number generators. I use the 64-bit Mersenne Twister by Matsumoto and Nishimura (2000).
- minor bugs like overflow checking.
- I think my Cross-Validation implementation can be sped up.
- Optimize code to run faster maybe Fortran?

References

[1] Harrison, D. and Rubinfeld, D.L. (1978) Hedonic prices and the demand for clean air. J. Environ. Economics and Management 5, 81-102.

[2] Stamey, T.A., Kabalin, J.N., McNeal, J.E., Johnstone, I.M., Freiha, F., Redwine, E.A. and Yang, N. (1989) Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate: II. radical prostatectomy treated patients, Journal of Urology 141(5), 1076–1083.

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

[4] 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

[5] 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

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

[7] 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/.

[8] 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.

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