RlassoModels: Rigorous Lasso for High-Dimensional Estimation and Inference in Python

Matias Piqueras

Abstract. The **RlassoModels** package is a Python implementation of

1. Introduction

The ever-increasing availability to data and computing power is fundamentally reshaping empirical social scientific research. Instead of costly surveys, political scientists can now leverage easily accessible social media data to make point-estimates of ideology (Barberá, 2015/ed). Sociologists, are able to learn about the dynamics of large scale social networks and extract meaning from text corpuses that would take more than a lifetime to read (Sapiezynski et al., 2019; Kozlowski et al., 2019). In the field of economics, fundamental but difficult to empirically identify concepts, such as demand curves can be studied by using data of individual level transactions measured at an unprecedented scale (Cohen et al., 2016).

New data has also necessitated the use of new methods. The desirable asymptotic properties of traditional statistical methods such as ordinary least squares (OLS) implicitly assume and indeed require, that the amount of variables p is fixed as $n \to \infty$. This is most evidently seen in the expected prediction error given by $E\left[\frac{1}{n}\sum_{i=1}^{n}\left(x_i'\hat{\beta} - x_i'\beta\right)^2\right] = \frac{\sigma^2 p}{n}$. However, in many modern practical applications the data is high-dimensional, in the $p > n$ or even $p \gg n$ sense, whereby the fraction $\frac{p}{n}$ is nonnegliable and traditional methods badly behaved. To overcome this, researchers have turned their attention to machine learning models such as tree based learners and neural networks that are better attuned to this setting. In (**?**) the tree based lear

2. Related Software

lassopack implements Lasso with data-driven and theoretically motivated penalty in Stata (Ahrens et al., 2020). The extension **pdslasso** implements the post-double-selection and post-regularization methodology for inference in the presence of high-dimensional controls and/or instrumental variables (Ahrens et al., 2019). In R, the corresponding package is **HDM** by Belloni et al. (2014a). **RlassoModels** is a Python port of the aforementioned packages. To my knowledge, there exists no direct equivalent, however, **DoubleML** (Bach et al., 2021) and **EconML** (Keith Battocchi, 2019) both implement a generalization of Lasso based methods for inference to a broad class of learners, called double machine learning or debiased machine learnming and introduced in Chetverikov et al. (2016). The method also relies on orthogonalized moment conditions but in contrast to rigorous lasso, requires sample splitting to obtain unbiased estimates and consistent confidence bands.

More generally, **RlassoModels** aims to be a contribution to both the machine learning and causal inference Python community. Syntactially, it follows the **scikit-learn** estimator and provides two estimators (`Rlasso` and `RlassoLogit`) that are fully compatible with the API (Pedregosa et al., 2011). It should also be familiar to econometricians working in Python by relying on **statsmodels** to provide informative outputs when the package is used for causal inference on low-dimensional exogenous and/or endogenous variables (Seabold and Perktold, 2010).

The standard scientific stack **Numpy**, **Scipy** and **Pandas** are used throughout (Van Der Walt et al., 2011; Harris et al., 2020; Virtanen et al., 2020; McKinney et al., 2011). For optimization, two alternatives are offered. Either through the convex optimization library **cvxpy** (Diamond and Boyd, 2016) that compiles the problem into a cone program or through a custom C++ implementation of the coordinate descent algorithm, first introduced in Fu (1998). Both support the regular lasso and the square-root lasso loss functions with regressor specific penalty loadings. In the latter, interoperability and exposure to Python types in C++ is achieved through **Pybind11** and the library **Eigen** is used to perform most matrix operations (Jakob et al., 2017; Guennebaud et al., 2014).

3. Methods and Models

3.1. **The Sparse High-Dimensional Model Framework.** This section offers a brief introduction to the canonical sparse high-dimensional model and will provide the basis for following discussions of the rigorous lasso estimator and its associated methods for inference. For a more comprehensive treatment of the subject, see e.g. (Belloni and Chernozhukov, 2013) and (Belloni et al., 2014b).

Consider the regression function

$$y_i = f(z_i) + \epsilon_i, \quad \epsilon_i \sim N(0, \sigma^2), i = 1, \ldots, n \tag{1}$$

where z_i is the vector of regressors, y_i is the outcome and ϵ_i an i.i.d but possibly heteroscedastic and non-Gaussian error term. The functional form $f(\cdot)$ is unknown and possibly complex but can be approximated by the linear function $x_i'\beta_{f0} + r_{fi}$, where r_{fi} is the approximation error and $x_i := P(z_i)$ are technical transformations of the original regressor set and defined in p-dimensional space. p is allowed to be "large" relative to the number of observations, possibly $n \gg p$. Such situations naturally arise in many empirical applications. In country-level comparative studies the number of observations is small and fixed but the number of relevant controls for some social phenomena might be very large.

3.2. Penalized Regression.

4. Rigorous Lasso Estimation

4.1. Feasible Lasso.

4.2. Square-Root Lasso.

$$\widehat{\beta} = \arg\min \frac{1}{\sqrt{n}} \|y_i - x_i'\beta\|_2 + \frac{\lambda}{n} \sum_{j=1}^{p} \psi_j |\beta_j| \tag{2}$$

$$\widehat{\beta} = \arg\min \frac{1}{n} \|y_i - x_i'\beta\|_2^2 + \frac{\lambda}{n} \sum_{j=1}^{p} \psi_j |\beta_j| \tag{3}$$

5. Inference on Low-Dimensional Variables in the Presence of High-Dimensional Controls and Instruments

5.1. Rigorous Penalty Level.

6. API Design

```
# define model
rlasso_iv = RlassoIV(select_X=True, select_Z=True)
# fit to data
rlasso_iv.fit(X, y, D_exog=None, D_endog=d_endog, Z=Z)
# display results
rlasso_iv.summary()
```

7. Benchmarks

Runtime comparisons to **lassopack** and **HDM** were performed on an Apple MacBook Air with a M1 8-core CPU. A correlated design matrix was generated similar to Belloni et al. (2011) and Ahrens et al. (2020) using the Toeplitz correlation matrix $x_i \sim N(0, \Sigma)$ where $\Sigma_{jk} = 0.9^{|j-k|}$. In total 20×20 matrices were generated for 20 equidistant number of observations $n \in \{100, \ldots, 10,000\}$ and covariates $p \in \{5, \ldots, 800\}$ respectively. The true support, i.e. number of active coefficients, is set to be a fraction of p such that $\beta_j = \mathbb{1}\{j \leq s\}$, for $j = 1, \ldots, p$ and where $s = \frac{p}{5}$. All simulations Calls to Stata

8. Monte-Carlo Examples

This section aims to highlight some of the finite sample properties of the rigorous lasso estimator that make it suitable for inferential research. The results are not new and the Monte-Carlo examples have already been displayed of several papers. As such, the section is intended as a proof of concept of the implementation

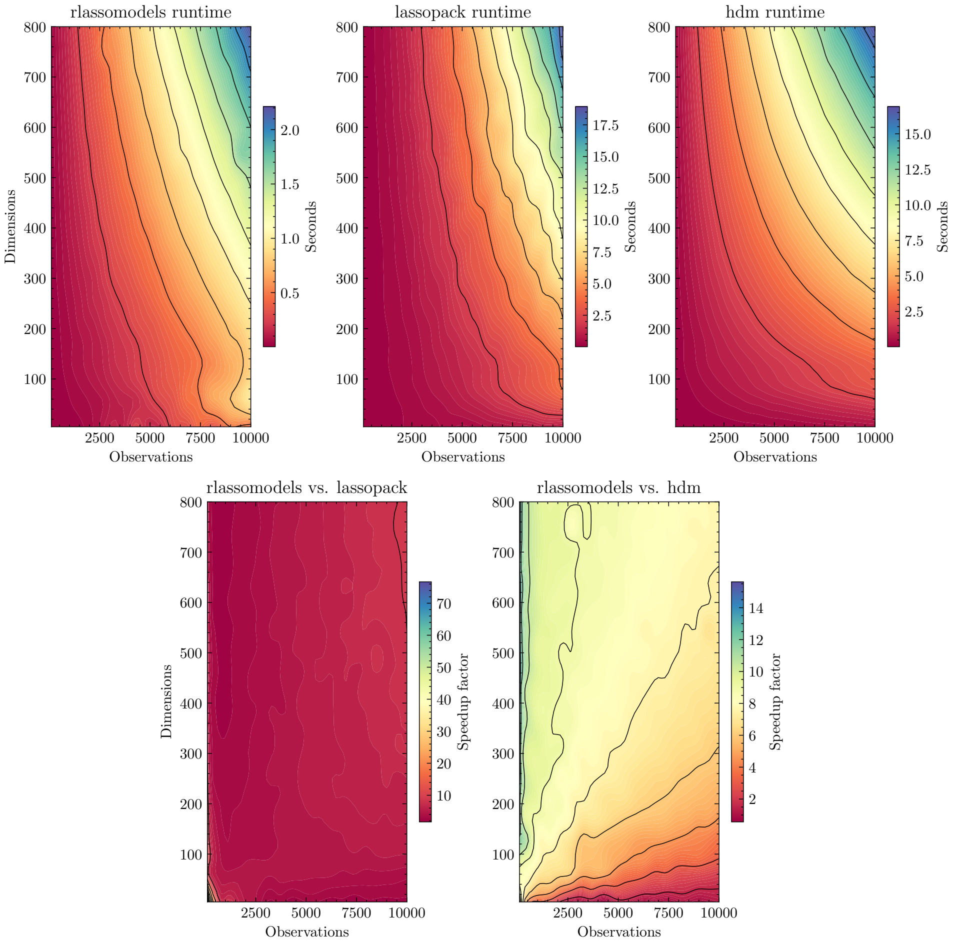

FIGURE 1. The figure shows finite sample performance of rigorous lasso and other common alternatives in terms of relative empirical risk and F1 score.

8.1. **Rigorous Lasso and post-lasso.** As a first step we analyze the performance of the rigorous choice of penalty level and penalty loadings relative to other common alternatives. Specifically, we consider the cross-validation method (CV), the Akaike information criterion (AIC) and the Bayesian information criterion (BIC). Data is generated the same way as inj Section 7, using the Toeplitz matrix, but with fixed $p = \dim(x_i) = 200$, $n = 100$ and the true support given by $\beta_j = \mathbb{1}\{j \leq 5\}$. The DGP linking x_i to y_i is linear with a stochastic noise term drawn from $\epsilon_i \sim N(0,1)$, for which we vary σ between 0.1 and 3.5. For each value of σ, 100 simulations are repeated and three performance metrics computed. Following Belloni et al. (2011), we compute the relative empirical risk of each estimator by

$$\hat{\delta} = \frac{\frac{1}{R}\sum_{r=1}^{R}\|\tilde{\beta}_r - \beta_0\|_2}{\frac{1}{R}\sum_{r=1}^{R}\|\beta_r^* - \beta_0\|_2} \tag{4}$$

where β^* is the oracle estimator, defined as running ordinary least squares on the regressors set with the true support and $\tilde{\beta}$ is the estimator under consideration. Secondly, the root-mean-squared error is computed. Lastly, the F1 score is computed to assess how well the estimator performs model selection by considering both the number of false positives and false negatives. The simulation results are summarized in Figure 2 and more details are given in Table **??**. From the first plot we can see that normal rigorous lasso is outperformed by CV, AIC and BIC. However, when doing post-lasso OLS, it achieves near oracle performance for $\sigma < 2$. This result is understandable from looking at the second plot which shows that rigorous lasso performs close to perfect model selection for the same values of σ. In contrast, CV and AIC perform poorly in terms of F1 as they are overfitting to the data and selecting too many coefficients.

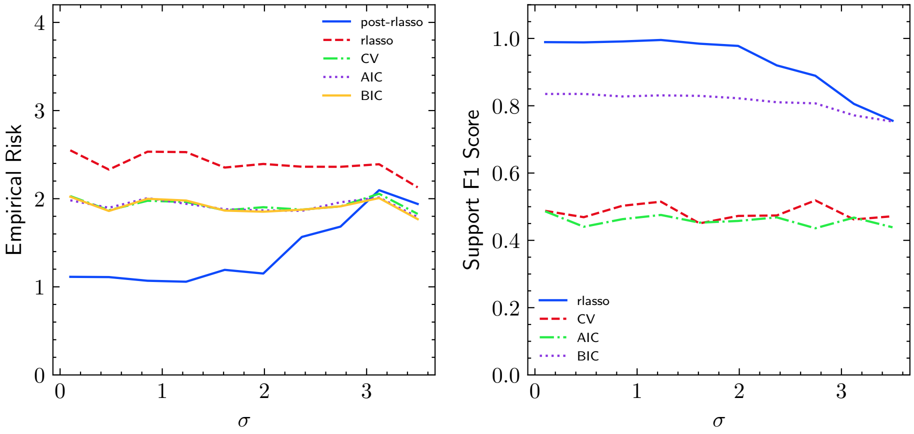

Figure 2. The figure shows finite sample performance of rigorous lasso and other common alternatives in terms of relative empirical risk and F1 score.

8.2. **Inferential properties.** We now move away from the prediction performance of rigorous lasso and focus on the performance of the post-double-selection and post-partialing out estimators when the goal is inference.

9. Empirical Examples

Appendix A. Optimization Details

Appendix B. Penalty Loadings

Appendix C. Further Monte-Carlo Experiments

References

Ahrens, Achim, Christian B. Hansen, and Mark Schaffer (2019): "PDSLASSO: Stata Module for Post-Selection and Post-Regularization OLS or IV Estimation and Inference," .

Ahrens, Achim, Christian B. Hansen, and Mark E. Schaffer (2020): "Lassopack: Model Selection and Prediction with Regularized Regression in Stata," *The Stata Journal: Promoting communications on statistics and Stata*, 20, 176–235.

Bach, Philipp, Victor Chernozhukov, Malte S. Kurz, and Martin Spindler (2021): "DoubleML – An Object-Oriented Implementation of Double Machine Learning in R," *arXiv:2103.09603 [cs, econ, stat]*.

Barberá, Pablo (2015/ed): "Birds of the Same Feather Tweet Together: Bayesian Ideal Point Estimation Using Twitter Data," *Political Analysis*, 23, 76–91.

Belloni, Alexandre and Victor Chernozhukov (2013): "Least Squares after Model Selection in High-Dimensional Sparse Models," *Bernoulli*, 19.

Belloni, Alexandre, Victor Chernozhukov, and Christian Hansen (2014a): "High-Dimensional Methods and Inference on Structural and Treatment Effects," *Journal of Economic Perspectives*, 28, 29–50.

Belloni, A., V. Chernozhukov, and C. Hansen (2014b): "Inference on Treatment Effects after Selection among High-Dimensional Controls," *The Review of Economic Studies*, 81, 608–650.

Belloni, A., V. Chernozhukov, and L. Wang (2011): "Square-Root Lasso: Pivotal Recovery of Sparse Signals via Conic Programming," *Biometrika*, 98, 791–806.

Chetverikov, Denis, Mert Demirer, Esther Duflo, Christian Hansen, Whitney K. Newey, and Victor Chernozhukov (2016): "Double Machine Learning for Treatment and Causal Parameters," Tech. rep., The IFS.

Cohen, Peter, Robert Hahn, Jonathan Hall, Steven Levitt, and Robert Metcalfe (2016): "Using Big Data to Estimate Consumer Surplus: The Case of Uber," Working Paper 22627, National Bureau of Economic Research.

Diamond, Steven and Stephen Boyd (2016): "CVXPY: A Python-embedded Modeling Language for Convex Optimization," *The Journal of Machine Learning Research*, 17, 2909–2913.

Fu, Wenjiang J (1998): "Penalized Regressions: The Bridge versus the Lasso," *Journal of computational and graphical statistics*, 7, 397–416.

Guennebaud, Gael, Benoit Jacob, et al. (2014): "Eigen: A C++ Linear Algebra Library," *URL http://eigen. tuxfamily. org, Accessed*, 22.

Harris, Charles R, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. (2020): "Array Programming with NumPy," *Nature*, 585, 357–362.

Jakob, Wenzel, Jason Rhinelander, and Dean Moldovan (2017): "Pybind11–Seamless Operability between C++ 11 and Python," *URL: https://github. com/pybind/pybind11.*

Keith Battocchi, Eleanor Dillon, Greg Lewis Paul Oka Miruna Oprescu Vasilis Syrgkanis Maggie Hei (2019): "EconML: A Python Package for ML-Based Heterogeneous Treatment Effects Estimation," .

Kozlowski, Austin C., Matt Taddy, and James A. Evans (2019): "The Geometry of Culture: Analyzing the Meanings of Class through Word Embeddings," *American Sociological Review*, 84, 905–949.

McKinney, Wes et al. (2011): "Pandas: A Foundational Python Library for Data Analysis and Statistics," *Python for high performance and scientific computing*, 14, 1–9.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. (2011): "Scikit-Learn: Machine Learning in Python," *the Journal of machine Learning research*, 12, 2825–2830.

Sapiezynski, Piotr, Arkadiusz Stopczynski, David Dreyer Lassen, and Sune Lehmann (2019): "Interaction Data from the Copenhagen Networks Study," *Scientific Data*, 6, 315.

Seabold, Skipper and Josef Perktold (2010): "Statsmodels: Econometric and Statistical Modeling with Python," in *Python in Science Conference*, Austin, Texas, 92–96.

Van Der Walt, Stefan, S Chris Colbert, and Gael Varoquaux (2011): "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in science & engineering*, 13, 22–30.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt

(2020): “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, 17, 261–272.