Journal of Machine Learning Research 24 (2023) 1-8 Submitted 2/22; Revised 4/23; Published 6/23

L0Learn: A Scalable Package for Sparse Learning using L0 Regularization

Hussein Hazimeh HAZIMEH@GOOGLE.COM
Google Research

Rahul Mazumder RAHULMAZ@MIT.EDU
Massachusetts Institute of Technology

Tim Nonet TIM.NONET@GMAIL.COM
Massachusetts Institute of Technology

Editor: Alexandre Gramfort

Abstract

We present `L0Learn`: an open-source package for sparse linear regression and classification using ℓ_0 regularization. `L0Learn` implements scalable, approximate algorithms, based on coordinate descent and local combinatorial optimization. The package is built using C++ and has user-friendly R and Python interfaces. `L0Learn` can address problems with millions of features, achieving competitive run times and statistical performance with state-of-the-art sparse learning packages. `L0Learn` is available on both CRAN and GitHub.[1]

Keywords: sparsity, sparse regression, sparse classification, ℓ_0 regularization, coordinate descent, combinatorial optimization

1. Introduction

High-dimensional data is common in many important applications of machine learning, such as genomics and healthcare (Bycroft et al., 2018). For statistical and interpretability reasons, it is desirable to learn linear models with sparse coefficients (Hastie et al., 2015). Sparsity can be directly obtained using ℓ_0 regularization, which controls the number of nonzero coefficients in the model. To illustrate this, let us consider the standard linear regression problem with a data matrix $X \in \mathbb{R}^{n\times p}$, regression coefficients $\beta \in \mathbb{R}^p$, and a response vector $y \in \mathbb{R}^n$. Using $\|\beta\|_0$ to denote the number of nonzero entries in β, the ℓ_0-regularized least-squares problem is given by:

$$\min_{\beta} \quad \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_0, \tag{1}$$

where $\lambda \geq 0$ is a regularization parameter. Problem (1) is a fundamental problem in statistics and machine learning known as *best subset selection*, since it searches for a subset of the features that leads to the best fit (in terms of squared error) (Hocking and Leslie, 1967). Statistical optimality properties of estimator (1) have been discussed in Greenshtein (2006); Raskutti et al. (2011); Zhang et al. (2014). Problem (1) is NP-hard and poses computational challenges. Various approaches have been proposed to approximate solutions to (1).

[1] Links: `https://cran.r-project.org/package=L0Learn` and `https://github.com/hazimehh/L0Learn`

©2023 Hussein Hazimeh, Rahul Mazumder, and Tim Nonet.

License: CC-BY 4.0, see `https://creativecommons.org/licenses/by/4.0/`. Attribution requirements are provided at `http://jmlr.org/papers/v24/22-0189.html`.

These include continuous proxies to the ℓ_0 norm such as the ℓ_1 norm (Tibshirani, 1996) and nonconvex penalties such as SCAD and MCP (Fan and Li, 2001; Zhang, 2010). For these continuous penalties, specialized software packages have been developed, e.g., `glmnet` (Friedman et al., 2010), `ncvreg` (Breheny and Huang, 2011), `sparsenet` (Mazumder et al., 2011), and `picasso` (Ge et al., 2019). Approaches for cardinality-constrained problems include greedy heuristics (e.g., stepwise methods), iterative hard thresholding (IHT) (Blumensath and Davies, 2008), BeSS (Wen et al., 2020), `abess` (Zhu et al., 2020), among others.

Recently, there have been significant advances in computing globally optimal solutions to ℓ_0-regularized problems, by using mixed integer programming (Wolsey and Nemhauser, 1999)—see for example Bertsimas et al. (2016); Bertsimas and Van Parys (2020); Hazimeh et al. (2022), and references therein. For example, ℓ_0-regularized regression problems can be solved to optimality in minutes to hours for $p \sim 10^7$ when highly sparse solutions are desired (Hazimeh et al., 2022). In several applications however, high-quality approximate solutions for ℓ_0-regularized problems may be more practical than obtaining global optimality certificates. Our recent works (Hazimeh and Mazumder, 2020; Dedieu et al., 2021) propose fast, approximate algorithms for computing high-quality solutions to ℓ_0-regularized problems, with running times comparable to that of fast ℓ_1 solvers. Our algorithms are based on a combination of coordinate descent (CD) and local combinatorial optimization, for which we establish convergence guarantees. Local search often results in improved solution quality over using CD alone. Indeed, the experiments in Hazimeh and Mazumder (2020); Dedieu et al. (2021) indicate that our algorithms can outperform state-of-the-art methods based on ℓ_1, MCP, SCAD, IHT, and others in terms of different statistical metrics (prediction, estimation, variable selection) and across a wide range of statistical settings.

This paper introduces `L0Learn`: a package for ℓ_0-regularized linear regression and classification. In addition to the ℓ_0 penalty, `L0Learn` supports continuous penalties such as the ℓ_1 or squared ℓ_2 norm. `L0Learn` implements highly optimized CD and local combinatorial optimization algorithms (Hazimeh and Mazumder, 2020; Dedieu et al., 2021) in C++, along with user-friendly R and Python interfaces to support fitting and visualizing models.

2. Package Overview

Problem formulation. In `L0Learn`, we consider a supervised learning setting with samples $\{(x_i, y_i)\}_{i=1}^n$, where $x_i \in \mathbb{R}^p$ is the i-th feature vector and $y_i \in \mathbb{R}$ is the corresponding response. Let $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be an associated loss function (such as squared error or logistic loss). `L0Learn` approximately minimizes the empirical risk, penalized with an $\ell_0\ell_1$ or $\ell_0\ell_2$ regularization. Specifically, for a fixed $q \in \{1, 2\}$, we compute approximate solutions to

$$\min_{\beta_0,\beta} \quad \sum_{i=1}^{n} L(y_i, \beta_0 + x_i^T\beta) + \lambda\|\beta\|_0 + \gamma\|\beta\|_q^q, \tag{2}$$

where λ and γ are non-negative regularization parameters. While the ℓ_0 penalty performs variable selection, the ℓ_q regularization induces shrinkage to help mitigate overfitting, especially in low-signal settings (Mazumder et al., 2023; Hazimeh and Mazumder, 2020). `L0Learn` supports regression using squared-error loss, classification using logistic and squared-hinge losses; and also accommodates box constraints on the β_is.

```
# Assume the data matrix (x) and response (y) have been loaded
fit <- L0Learn.fit(x, y, penalty="L0") # Fit an L0 regularized regression model
plot(fit) # Plot the regularization path
cv_fit <- L0Learn.cvfit(x, y, penalty="L0", nFolds=5) # 5-fold cross validation
plot(cv_fit) # Plot the cross-validation error
```

Figure 1: Simple examples of using `L0Learn` in R.

Overview of the algorithms. `L0Learn` uses a combination of (i) cyclic CD and (ii) local combinatorial optimization. The choice of CD is inspired by its strong performance in sparse learning with continuous penalties (Friedman et al., 2010; Mazumder et al., 2011). Standard convergence results for cyclic CD, e.g., Tseng (2001), do not apply for the discontinuous objective in (2). Hence, in Hazimeh and Mazumder (2020), we show that a variant of cyclic CD converges to a local minimizer of Problem (2). Given a CD solution (β_0, β), the local combinatorial optimization algorithm searches a local neighborhood for solutions with a better objective.[2] After finding a new improved solution using local search, we run CD with the new solution in an attempt to further improve the current solution. We keep iterating between local search and CD, until convergence—see Hazimeh and Mazumder (2020) for details.

Efficient computation of the regularization path. `L0Learn` solves (2) over a grid of regularization parameters. We use computational schemes such as warm starts, active sets, greedy cyclic order for CD, and correlation screening (Hazimeh and Mazumder, 2020), to speed up the algorithm. Due to the nature of the ℓ_0 penalty, different values of λ in (2) can lead to the same solution. Thus, to avoid duplicate solutions and unnecessary computations, we develop a new method that computes a data-dependent grid of λs, which is guaranteed to avoid duplicate solutions—see Hazimeh and Mazumder (2020) for details.

Implementation and development. All the algorithms in the package are implemented in C++, along with high-level R and Python interfaces. For linear algebra operations, we rely on the `Armadillo` library (Sanderson and Curtin, 2016), which is accelerated by Basic Linear Algebra Subprograms (BLAS) (Lawson et al., 1979). The R interface is integrated with C++ using `RcppArmadillo` (Eddelbuettel and Sanderson, 2014). The Python interface is integrated with C++ using `carma` (Urlus, 2023) and `pybind11` (Jakob et al., 2017). `L0Learn` supports both dense and sparse data matrices. Sparse matrices generally speed up computation and reduce memory requirements. In all functions that require the data matrix as an input argument, we use function templates that accept a generic matrix type. Thus, exactly the same code is used for both dense and sparse matrices, but each matrix type uses specialized linear algebra implementations.

For development, we use continuous integration based on Travis CI to build and test the package. Our unit and integration tests are primarily implemented in R based on `testthat` (Wickham, 2011) and have a coverage of 97% (as measured by `covr`). `L0Learn` achieved the highest rating (A) for code quality by the code analysis tool Codacy (`www.codacy.com`). We have additional CI checks that use Github Actions for our Python bindings to ensure the Python bindings interact with the C++ library identically to the R bindings.

[2] We implement the case where the neighborhood consists of all solutions obtained by removing one variable from the support of β and adding another previously zero variable to the support.

3. Usage and Documentation

`L0Learn` can be installed in R by executing `install.packages("L0Learn")` and can be installed in Python by executing `pip install l0learn`. It is supported on Linux, macOS, and Windows. In Figure 1, we provide simple examples of how to use `L0Learn` in R. More elaborate examples and a full API documentation can be found in `L0Learn`'s Vignette and Reference Manual, which are available on `L0Learn`'s main pages[1]. Similar examples and API documentation are available in Python as well.

4. Experiments

Our main goal in these experiments is to compare the running time of `L0Learn` with similar toolkits, designed for sparse learning problems. Specifically, we compare with `glmnet`, `ncvreg`, `picasso`, and `abess`. For space constraints, we focus on linear regression, and refer the reader to Dedieu et al. (2021) for sparse classification experiments. Our experiments also shed some light on the statistical performance of the different approaches: for in-depth studies of statistical properties, see Hazimeh and Mazumder (2020); Hastie et al. (2020); Mazumder et al. (2023). We note that there are modern approaches for solving ℓ_0-regularized problems to global optimality e.g., Bertsimas and Van Parys (2020); Hazimeh et al. (2022, 2023). However, due to their focus on optimality certification, they are usually (much) slower than the competing toolkits we present here (Hazimeh et al., 2022).

Setup. Following Hazimeh and Mazumder (2020), we consider synthetic data as per a linear regression model under the fixed design setting (exponential correlation model with $\rho = 0.3$). We take $n = 1000$, $k = 50$, signal-to-noise ratio (SNR)[3] to be 5 and vary $p \in \{10^3, 10^4, 10^5\}$. In `L0Learn`, we used the default CD algorithm with the $\ell_0\ell_2$ penalty. In `picasso`, we used ℓ_1 regularization and changed the convergence threshold (prec) to 10^{-10} so that its solutions roughly match those of `glmnet`. In `ncvreg`, we used the (default) MCP penalty. All competing methods are tuned to minimize MSE on a validation set with the same size as the training set. In `L0Learn`, `ncvreg`, and `abess`, we tune over a two-dimensional grid consisting of 100 λ values (chosen automatically by the toolkits) and 100 γ values (in the range $[10^{-2}, 10^2]$ for `L0Learn`, $[1.5, 10^3]$ for `ncvreg`, $[1, 10^3]$ for `abess`[4]). Experiments were performed on a Linux c5n.2xlarge EC2 instance running R 4.0.2.

Metrics. We report the running time in seconds for computing a path of 100 solutions. (L0Learn, `abess`, and `ncvreg` require additional time for tuning their second parameter γ). In addition, given an estimator $\hat{\beta}$, we compute the following statistical metrics: (i) prediction error (PE) given by $\|X\hat{\beta} - X\beta^*\|_2/\|X\beta^*\|_2$, (ii) the number of false positives (FP), i.e., the nonzero variables in $\hat{\beta}$ that are not in β^*, and (iii) the support size (SS).

Results. In Table 1, we report the metrics averaged over 10 repetitions. The results indicate that `L0Learn` achieves best-in-class run times and statistical performance. While we only show results on synthetic data sets due to space constraints, `L0Learn` is useful and can lead to good improvements on real data sets, as demonstrated in many studies,

[3] Similar to Bertsimas et al. (2016), SNR := $\text{Var}(X\beta^*)/\sigma^2$.

[4] We found out that `abess` requires a different range for γ (the parameter of the squared ℓ_2 reguralizer) to perform well. This is expected since `abess` considers a different, cardinality-constrained formulation.

	$p = 10^3$				$p = 10^4$				$p = 10^5$			
	Time	PE$\times 10^2$	FP	SS	Time	PE$\times 10^2$	FP	SS	Time	PE$\times 10^2$	FP	SS
L0Learn	**0.09 (0.01)**	**9.4 (1.2)**	**0 (0)**	**50 (0)**	**0.49 (0.0)**	**9.4 (0.8)**	**0 (0)**	**50 (0)**	**4.4 (0.4)**	**9.5 (1.1)**	**0 (0)**	**50 (0)**
glmnet	0.55 (0.02)	19.8 (1.0)	154 (22)	204 (22)	0.94 (0.0)	26.4 (0.9)	300 (22)	350 (22)	8.0 (0.1)	32.3 (1.5)	485 (38)	535 (38)
picasso	1.46 (.15)	19.8 (1.0)	157 (24)	207 (24)	2.92 (0.0)	26.4 (0.9)	303 (22)	353 (22)	15.5 (0.2)	32.3 (1.5)	483 (35)	533 (35)
abess	0.43 (0.04)	11.0 (1.8)	3.9 (3)	53.9 (3)	7.21 (0.1)	16.2 (7.8)	2 (0)	52 (0)	-	-	-	-
ncvreg	1.35 (0.22)	**9.4 (1.2)**	2 (5)	52 (5)	3.74 (0.4)	**9.4 (0.7)**	4 (7)	54 (7)	19.7 (0.6)	9.6 (1.0)	1 (1)	51 (1)

Table 1: The mean and standard error of the running time (s), prediction error (PE), number of false positives (FP), and support size (SS). A dash indicates failure due to memory issues.

including Hazimeh and Mazumder (2020); OShea et al. (2021); Cao et al. (2021); Li et al. (2023).

5. Conclusion

We introduced `L0Learn`: a scalable package for ℓ_0-regularized regression and classification. The package is implemented in C++ along with R and Python interfaces. It offers two approximate algorithms: a fast coordinate descent-based method and a local combinatorial search algorithm that helps in improving solution quality. Our experiments indicate that `L0Learn` is highly scalable and can outperform popular sparse learning toolkits in important high-dimensional settings. The package has been successfully used in a variety of applications in healthcare and genetics; and has also proven to be effective in warm starting exact ℓ_0 regularization solvers (Hazimeh et al., 2022). As a future direction, it would be interesting to study the performance of CD and local combinatorial search in (ℓ_0-based) pruning of neural networks (Benbaki et al., 2023).

Acknowledgements

We acknowledge research support from the Office of Naval Research and National Science Foundation.

References

Riade Benbaki, Wenyu Chen, Xiang Meng, Hussein Hazimeh, Natalia Ponomareva, Zhe Zhao, and Rahul Mazumder. Fast as chita: Neural network pruning with combinatorial optimization. *arXiv preprint arXiv:2302.14623*, 2023.

Dimitris Bertsimas and Bart Van Parys. Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *The Annals of Statistics*, 48(1):300–323, 2020.

Dimitris Bertsimas, Angela King, and Rahul Mazumder. Best subset selection via a modern optimization lens. *Annals of Statistics*, 44(2):813–852, 2016.

Thomas Blumensath and Mike Davies. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14(5-6):629–654, 2008.

Patrick Breheny and Jian Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The annals of applied statistics*, 5(1):232, 2011.

Clare Bycroft, Colin Freeman, Desislava Petkova, Gavin Band, Lloyd T Elliott, Kevin Sharp, Allan Motyer, Damjan Vukcevic, Olivier Delaneau, and Jared O'Connell. The uk biobank resource with deep phenotyping and genomic data. *Nature*, 562(7726):203–209, 2018.

Chen Cao, Jingni He, Lauren Mak, Deshan Perera, Devin Kwok, Jia Wang, Minghao Li, Tobias Mourier, Stefan Gavriliuc, Matthew Greenberg, et al. Reconstruction of microbial haplotypes by integration of statistical and physical linkage in scaffolding. *Molecular biology and evolution*, 38(6):2660–2672, 2021.

Antoine Dedieu, Hussein Hazimeh, and Rahul Mazumder. Learning sparse classifiers: Continuous and mixed integer optimization perspectives. *Journal of Machine Learning Research*, 22(135):1–47, 2021.

Dirk Eddelbuettel and Conrad Sanderson. Rcpparmadillo: Accelerating r with high-performance c++ linear algebra. *Computational Statistics & Data Analysis*, 71:1054–1063, 2014.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL `http://www.jstatsoft.org/v33/i01/`.

Jason Ge, Xingguo Li, Haoming Jiang, Han Liu, Tong Zhang, Mengdi Wang, and Tuo Zhao. Picasso: A sparse learning library for high dimensional data analysis in r and python. *J. Mach. Learn. Res.*, 20(44):1–5, 2019.

Eitan Greenshtein. Best subset selection, persistence in high-dimensional statistical learning and optimization under l1 constraint. *The Annals of Statistics*, 34(5):2367–2386, 2006.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

Trevor Hastie, Robert Tibshirani, and Ryan Tibshirani. Best Subset, Forward Stepwise or Lasso? analysis and recommendations based on extensive comparisons. *Statistical Science*, 35(4):579–592, 2020.

Hussein Hazimeh and Rahul Mazumder. Fast best subset selection: Coordinate descent and local combinatorial optimization algorithms. *Operations Research*, 68(5):1517–1537, 2020. doi: 10.1287/opre.2019.1919. URL `https://doi.org/10.1287/opre.2019.1919`.

Hussein Hazimeh, Rahul Mazumder, and Ali Saab. Sparse regression at scale: Branch-and-bound rooted in first-order optimization. *Mathematical Programming*, 196(1-2):347–388, 2022.

Hussein Hazimeh, Rahul Mazumder, and Peter Radchenko. Grouped variable selection with discrete optimization: Computational and statistical perspectives. *The Annals of Statistics*, 51(1):1–32, 2023.

Ronald R Hocking and RN Leslie. Selection of the best subset in regression analysis. *Technometrics*, 9(4):531–540, 1967.

Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11 – seamless operability between c++11 and python, 2017. https://github.com/pybind/pybind11.

Chuck L Lawson, Richard J. Hanson, David R Kincaid, and Fred T. Krogh. Basic linear algebra subprograms for fortran usage. *ACM Transactions on Mathematical Software (TOMS)*, 5(3):308–323, 1979.

Xiong Li, Xu Meng, Haowen Chen, Xiangzheng Fu, Peng Wang, Xia Chen, Changlong Gu, and Juan Zhou. Integration of single sample and population analysis for understanding immune evasion mechanisms of lung cancer. *npj Systems Biology and Applications*, 9(1): 4, 2023.

Rahul Mazumder, Jerome H. Friedman, and Trevor Hastie. Sparsenet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106(495): 1125–1138, 2011. doi: 10.1198/jasa.2011.tm09738. URL `https://doi.org/10.1198/jasa.2011.tm09738`. PMID: 25580042.

Rahul Mazumder, Peter Radchenko, and Antoine Dedieu. Subset selection with shrinkage: Sparse linear modeling when the snr is low. *Operations Research*, 71(1):129–147, 2023.

Robert J OShea, Sophia Tsoka, Gary JR Cook, and Vicky Goh. Sparse regression in cancer genomics: Comparing variable selection and predictions in real world data. *Cancer informatics*, 20:11769351211056298, 2021.

Garvesh Raskutti, Martin Wainwright, and Bin Yu. Minimax rates of estimation for high-dimensional linear regression over l_q-balls. *IEEE transactions on information theory*, 57 (10):6976–6994, 2011.

Conrad Sanderson and Ryan Curtin. Armadillo: a template-based c++ library for linear algebra. *Journal of Open Source Software*, 1(2):26, 2016.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494, 2001. ISSN 1573-2878. doi: 10.1023/A:1017501703105. URL `http://dx.doi.org/10.1023/A:1017501703105`.

Ralph Urlus. CARMA: bidirectional conversions between Numpy and Armadillo, 4 2023. URL `https://github.com/RUrlus/carma`.

Canhong Wen, Aijun Zhang, Shijie Quan, and Xueqin Wang. Bess: an r package for best subset selection in linear, logistic and cox proportional hazards models. *Journal of Statistical Software*, 94:1–24, 2020.

Hadley Wickham. testthat: Get started with testing. *The R Journal*, 3(1):5–10, 2011.

Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*, volume 55. John Wiley & Sons, 1999.

Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, February 2010.

Yuchen Zhang, Martin J Wainwright, and Michael I Jordan. Lower bounds on the performance of polynomial-time algorithms for sparse linear regression. In *Conference on Learning Theory*, pages 921–948. PMLR, 2014.

Junxian Zhu, Canhong Wen, Jin Zhu, Heping Zhang, and Xueqin Wang. A polynomial algorithm for best-subset selection problem. *Proceedings of the National Academy of Sciences*, 117(52):33117–33123, 2020.