abess: A Fast Best-Subset Selection Library in Python and R

Jin Zhu1
Xueqin Wang2
Liyuan Hu1
Junhao Huang1
Kangkang Jiang1
Yanhang Zhang3
Shiyun Lin4
Junxian Zhu5

1Department of Statistical Science, Sun Yat-Sen University, Guangzhou, GD, China
2Department of Statistics and Finance/International Institute of Finance, School of Management, University of Science and Technology of China, Hefei, Anhui, China
3School of Statistics, Renmin University of China, Beijing, China
4Center for Statistical Science, Peking University, Beijing, China
5Saw Swee Hock School of Public Health, National University of Singapore, Singapore

Editor: Alexandre Gramfort

Abstract

We introduce a new library named abess that implements a unified framework of best-subset selection for solving diverse machine learning problems, e.g., linear regression, classification, and principal component analysis. Particularly, abess certifiably gets the optimal solution within polynomial time with high probability under the linear model. Our efficient implementation allows abess to attain the solution of best-subset selection problems as fast as or even 20x faster than existing competing variable (model) selection toolboxes. Furthermore, it supports common variants like best subset of groups selection and $\ell_2$ regularized best-subset selection. The core of the library is programmed in C++. For ease of use, a Python library is designed for convenient integration with scikit-learn, and it can be installed from the Python Package Index (PyPI). In addition, a user-friendly R library is available at the Comprehensive R Archive Network (CRAN). The source code is available at: https://github.com/abess-team/abess.

Keywords: best-subset selection, high-dimensional data, splicing technique

1. Introduction

Best-subset selection (BSS) is imperative in machine learning and statistics. It aims to find a minimally adequate subset of variables to accurately fit the data, naturally reflecting Occam’s razor principle of simplicity. Nowadays, the BSS also has far-reaching applications in every facet of research like medicine and biology because of the surge of large-scale data sets across a variety of fields. As a benchmark optimization problem in machine learning and statistics, the BSS is also well-known as an NP-hard problem (Natarajan, 1995). However, recent progress shows that the BSS can be efficiently solved (Huang et al., 2018; Zhu et al., 2020; Gómez and Prokopyev, 2021). Especially, the ABESS algorithm using a splicing
Table 1: The supported best-subset selection solvers. PCA: principal component analysis.

<table>
<thead>
<tr>
<th>Learning</th>
<th>Target</th>
<th>Solver (Reference)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised</td>
<td>$y \in \mathbb{R}$</td>
<td>LinearRegression (Zhu et al., 2020)</td>
</tr>
<tr>
<td></td>
<td>$y \in {0, 1}$</td>
<td>LogisticRegression (Hosmer et al., 1989)</td>
</tr>
<tr>
<td></td>
<td>$y \in {0, 1, 2, \ldots}$</td>
<td>PoissonRegression (Vincent and Claire 2010)</td>
</tr>
<tr>
<td></td>
<td>$y \in (0, \infty)$</td>
<td>GammaRegression (Vincent and Claire 2010)</td>
</tr>
<tr>
<td></td>
<td>$y \in {\text{type1, type2, \ldots}}$</td>
<td>MultinomialRegression (Krishnapuram et al., 2005)</td>
</tr>
<tr>
<td></td>
<td>$y \in {\text{level1, level2, \ldots}}$</td>
<td>OrdinalRegression (Wurm et al., 2021)</td>
</tr>
<tr>
<td></td>
<td>$y \in \mathbb{R} \times {0, 1}$</td>
<td>CoxPHSurvivalAnalysis (Pölsterl, 2020)</td>
</tr>
<tr>
<td></td>
<td>$y \in \mathbb{R}^d$</td>
<td>MultiTaskRegression (Zhang and Yang, 2017)</td>
</tr>
</tbody>
</table>

Unsupervised

| Dimension reduction | SparserPCA (d'Aspremont et al., 2008) |
| Matrix decomposition | RobustPCA (Cai et al., 2019) |

2. Architecture

Figure 1 shows the architecture of abess, and each building block will be described as follows. The Data class accepts the (sparse) tabular data from Python and R interfaces, and returns an object containing the predictors that are (optionally) screened (Fan and Lv, 2008) or normalized. The Algorithm class implements the generic splicing technique for the BSS with the additional support for group-structure predictors (Zhang et al., 2021), $\ell_2$-regularization for parameters (Bertsimas and Parys, 2020), and nuisance selection (Sun and Zhang, 2021). The concrete algorithms are programmed as subclasses of Algorithm by rewriting the virtual function interfaces of class Algorithm. Seven implemented BSS tasks are presented in Figure 1. Beyond that, the modularized design facilitates users to extend the library to various machine learning tasks by writing a subclass of Algorithm. The Metric class assesses the estimation returned by the Algorithm class by the cross-validation or information criteria like the Akaike information criterion and the high dimensional Bayesian information criterion (Akaike, 1998; Wang et al., 2013). Python and R interfaces collect and process the results of the Algorithm and Metric classes. The abess Python library is compatible with scikit-learn (Pedregosa et al., 2011). For each solver (e.g., LinearRegression) in abess, Python users can not only use a familiar scikit-learn API to

1. https://pypi.org/project/abess
2. https://cran.r-project.org/web/packages/abess
4. https://github.com/abess-team/abess/actions
Figure 1: \textit{abess} software architecture.

```r
library(abess)
dat <- generate.data(n = 300, p = 1000, beta = c(3, -2, 0, 0, 2, rep(0, 995)))
best_est <- extract(abess(dat$x, dat$y, family = "gaussian"))
cat("Selected subset:", best_est$support.vars,
"and coefficient estimation:", round(best_est$support.beta, digits = 2))
## Selected subset: x1 x2 x5 and coefficient estimation: 2.96 -2.05 1.9
```

Figure 2: Using the \textit{abess} R library on a synthetic data set to illustrate its optimality. The data set comes from a linear model with the true sparse coefficients given by \texttt{beta}.

3. Usage Examples

Figure 2 shows that the \textit{abess} R library exactly selects the effective variables and accurately estimates the coefficients. Figure 3 illustrates the integration of the \textit{abess} Python interface with \textit{scikit-learn}'s modules to build a non-linear model for diagnosing malignant tumors. The output of the code reports the information of the polynomial features for the selected model among candidates, and its corresponding area under the curve (AUC), which is 0.966, indicating the selected model would have an admirable contribution in practice.

4. Performance

We compare \textit{abess} with popular variable selection libraries in Python and R through regression, classification, and PCA. The libraries include: \textit{scikit-learn} (a benchmark Python library for machine learning), \textit{celer} (a fast Python solver for \(\ell_1\)-regularization optimization, Massias et al., 2018, 2020), and (an elastic-net R solver for sparse PCA, Zou et al., 2006). All computations are conducted on a Ubuntu platform with Intel(R) Core(TM) i9-9940X.
from abess.linear import LogisticRegression
from sklearn.datasets import load_breast_cancer
from sklearn.pipeline import Pipeline
from sklearn.metrics import make_scorer, roc_auc_score
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import GridSearchCV

# combine feature transform and model:
pipe = Pipeline([('poly', PolynomialFeatures(include_bias=False)),
                 ('logreg', LogisticRegression())])

param_grid = {  
    'poly_interaction_only': [True, False],  
    'poly_degree': [1, 2, 3]
}

# Use cross validation to tune parameters:
scorer = make_scorer(roc_auc_score, greater_is_better=True)
grid_search = GridSearchCV(pipe, param_grid, scoring=scorer, cv=5)

# load and fitting example data set:
X, y = load_breast_cancer(return_X_y=True)
grid_search.fit(X, y)

# print the best tuning parameter and associated AUC score:
print([grid_search.best_params_, grid_search.best_score_])

# >>> [{'poly_degree': 2, 'poly_interaction_only': True}, 0.9663829492654472]

Figure 3: Example of using the abess Python library with scikit-learn.

<table>
<thead>
<tr>
<th>Library</th>
<th>Version</th>
<th>Superconductivity (3895 x 85400)</th>
<th>Cancer (118 x 22215)</th>
<th>Musk (7074 x 166)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MSE</td>
<td>NNZ</td>
<td>Runtime</td>
</tr>
<tr>
<td>scikit-learn ($\ell_1$)</td>
<td>1.0.0</td>
<td>33.56</td>
<td>1126.70</td>
<td>1043.96</td>
</tr>
<tr>
<td>celer</td>
<td>0.6.1</td>
<td>88.58</td>
<td>30.00</td>
<td>173.25</td>
</tr>
<tr>
<td>scikit-learn ($\ell_0$)</td>
<td>1.0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>abess</td>
<td>0.4.5</td>
<td>41.72</td>
<td>81.50</td>
<td>110.41</td>
</tr>
</tbody>
</table>

Table 2: Average performance on the superconductivity data set (for regression), the cancer and the musk data sets (for classification) (Chin et al., 2006; Dua and Graff, 2017; Hamidieh, 2018) based on 20 randomly drawn test sets. NNZ: the number of non-zero elements. Runtime is measured in seconds. scikit-learn ($\ell_1$): LassoCV (for regression) and LogisticRegressionCV (for classification). celer: LassoCV (for regression) and LogisticRegression (for classification). scikit-learn ($\ell_0$): Orthogonal-MatchingPursuit (for regression). X: not available. $\dagger$: memory overflow.

CPU @ 3.30GHz and 48 RAM. Python version is 3.9.1 and R version is 3.6.3. Table 2 displays the regression and classification analysis results, suggesting abess derives parsimonious models that achieve competitive performance in few minutes. Particularly, for the cancer data set, it is more than 20x faster than scikit-learn ($\ell_1$). The results of the sparse PCA (SPCA) are demonstrated in Table 3. Compared with elasticnet, abess consumes less than a tenth of its runtime but explains more variance under the same sparsity level.

5. Conclusion

abess is a fast and comprehensive library for solving various BSS problems with statistical guarantees. It offers user-friendly interfaces for both Python and R users, and seamlessly integrates with existing ecosystems. Therefore, the abess library is a potentially indispensable toolbox for machine learning and related applications. Future versions of abess intend to support other important machine learning tasks, and adapt to advanced machine learning pipelines in Python and R (Lang et al., 2019; Feurer et al., 2021; Binder et al., 2021).

<table>
<thead>
<tr>
<th>Library</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elassticnet</td>
<td>abess</td>
<td>abess</td>
<td>abess</td>
</tr>
<tr>
<td>Explained variance</td>
<td>1.37</td>
<td>2.28</td>
<td>1.61</td>
</tr>
<tr>
<td>Runtime (seconds)</td>
<td>15.87</td>
<td>1.06</td>
<td>19.77</td>
</tr>
</tbody>
</table>

Table 3: Performance of the SPCA when 5, 10, 20 elements in the loading vector of the first principal component are non-zero. The data set has 217 observations, each of which has 1,413 genetic factors (Christensen et al., 2009). elasticnet: version 1.3.0.
Acknowledgments

We would like to thank three reviewers for their constructive suggestions and valuable comments, which have substantially improved this article and the abess library. Wang’s research is partially supported by NSFC (72171216, 71921001, 71991474), The Key Research and Development Program of Guangdong, China (2019B020228001), and Science and Technology Program of Guangzhou, China (202002030129). Zhu’s research is partially supported by the Outstanding Graduate Student Innovation and Development Program of Sun Yat-Sen University (19lgys64). Zhang’s research is supported by the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (22XNH161). We are grateful to UCI Machine Learning Repository for sharing the superconductivity and musk data sets.

References


Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher,


