An Introduction to abess

Jin Zhu

Contents

Briey introduction ................................................................. 1
Quick example ................................................................. 1
    Fixed support size best subset selection ............................... 1
    Adaptive best subset selection ........................................... 2

Briey introduction

The R package abess implement a polynomial algorithm in the paper for best-subset selection problem:

\[
\min_{\beta \in \mathbb{R}^p} \frac{1}{2n} \|y - X\beta\|_2^2, \text{ subject to } \|\beta\|_0 \leq s,
\]

where \(\| \cdot \|_2\) is the \(\ell_2\) norm, \(\|\beta\|_0 = \sum_{i=1}^{p} I(\beta_i \neq 0)\) is the \(\ell_0\) norm of \(\beta\), and the sparsity level \(s\) is usually an unknown non-negative integer. Next, we present an example to show how to use the abess package to solve a simple problem.

Quick example

Fixed support size best subset selection

We generate a design matrix \(X\) containing 300 observation and each observation has 1000 predictors. The response variable \(y\) is linearly related to the first, second, and fifth predictors in \(X\):

\[
y = 3X_1 + 1.5X_2 + 2X_5 + \epsilon,
\]

where \(\epsilon\) is a standard normal random variable.

```r
library(abess)

synthetic_data <-
    generate.data(n = 300, p = 1000,
                      beta = c(3, 1.5, 0, 0, 2, rep(0, 995)))

dat <- cbind.data.frame("y" = synthetic_data["y"],
                      synthetic_data["x"])

dim(synthetic_data["x"])

## [1] 300 1000

head(synthetic_data["y"])

## [,1]
## [1,] -4.063922
## [2,] 3.855246
## [3,] -3.041391
## [4,] -1.081257
## [5,] 4.986772
## [6,] 4.470901
```

```
Then, we use the main function `abess` in the package to fit this dataset. By setting the arguments `support.size = s`, `abess` function conducts Algorithm 1 in the paper for best-subset selection with a sparsity level $s$. In our example, we set the options: `support.size = 3`, and we run Algorithm 1 with the following command:

```r
abess_fit <- abess(y ~ ., data = dat, support.size = 3)
```

The output of `abess` comprises the selected best model:

```r
str(abess_fit[["best.model"]])
```

```
## List of 6
## $ beta : Named num [1:1000] 2.96 1.45 0 0 1.91 ...
## ..- attr(*, "names")= chr [1:1000] "x1" "x2" "x3" "x4" ...
## $ coef0 : num -0.018
## $ support.index: int [1:3] 1 2 5
## $ support.size : int 3
## $ dev : num 1.31
## $ tune.value : num 117
```

The best model's support set is identical to the ground truth, and the coefficient estimation is the same as the oracle estimator given by `lm` function:

```r
lm(y ~ ., data = dat[, c(1, c(1, 2, 5) + 1)])
```

```
## Coefficients:
## (Intercept) x1 x2 x5
## -0.01802 2.96418 1.45091 1.90592
```

Adaptive best subset selection

Imagining we are unknown about the true sparsity level in real world data, and thus, we need to determine the most proper one. The Algorithm 3 in the paper is designed for this scenario. `abess` is capable of performing this algorithm:

```r
abess_fit <- abess(y ~ ., data = dat)
```

The output of `abess` also comprises the selected best model:

```r
str(abess_fit[["best.model"]])
```

```
## List of 6
## $ beta : Named num [1:1000] 2.96 1.45 0 0 1.91 ...
## ..- attr(*, "names")= chr [1:1000] "x1" "x2" "x3" "x4" ...
## $ coef0 : num -0.018
## $ support.index: int [1:3] 1 2 5
## $ support.size : int 3
## $ dev : num 1.31
## $ tune.value : num 117
```

The output model accurately detect the true model size, which implies the Algorithm 3 efficiently find both the optimal sparsity level and true effective predictors.