An Introduction to `subDebiased`

Waverly Wei

Jan.2021

Contents

Introduction

subDebiased is a package that implements two bootstrap-assisted estimators: bootstrap-assisted desparsified Lasso and R-split. The two methods remove the subgroup selection bias and regularization bias indeced by high-dimensional covariates.

Installation

```
devtools::install_github("WaverlyWei/subDebiased")
```

Quick start

First we load the `subDebiased` package:

```
library(subDebiased)
```

We generate high-dimensional data with 2 subgroups of interest. We predefine a set of tuning parameters, denoted as r.

```
  library(MASS)
  library(glmnet)

  p <- 200 # number of confounders

  n <- 100 # sample size

  ngroups <- 2 # number of subgroups/treatments;

  s0 <- 4

  m <- ngroups

  Sigma <- matrix(0,p,p)
```

```r
  for (i in 1:n){
      for(j in 1:p){
        Sigma[i,j] <- 0.5^(abs(i-j))
      }
    }

    # generate X
    X <- mvrnorm( n = n, mu = rep(0,p), Sigma = Sigma )

    Z <- matrix(0,n,m)

    for(i in 1:n){
      for(j in 1:m){
        Z[i,j] <- rbinom(1,1,exp(X[i,2*j-1] + X[i,2*j])/(1+exp(X[i,2*j-1] + X[i,2*j])))
      }
    }


    # noise: heter/homo
    noise.y <- 1

    betas <- 1

    #index of the subgroups
    w.index <- seq(1, m, 1)

    x <- cbind(Z,X)


    ## Model: Y = Z * beta + X * gamma + noise

    # Generate coefficients
    beta <- c(rep(0,m-1),betas)

    gamma <- c(rep(1, s0), rep(0, p-s0))

    beta0 <- c(beta, gamma)

    # Generate noise
    noise <- mvrnorm( n = 1, mu = rep(0,n), Sigma = diag(n) * noise.y )

    # Generate response Y
    Y <- 0.5 + x %*% beta0 + noise

    ## parameters in the function
    r <- 1/(3*1:10)
```

Bootstrap-calibrated Desparsified Lasso

Bootstrap iterations are recommended to be B = 200. Here we use B = 5 for demonstration purpose.

```r
desparse_res <- BSDesparseLasso(y = Y,
                                x = x,
```

```r
                        r = r,
                        G = w.index,
                        B = 5)
```

Result summary

The tuning parameter is selected through `cvDesparse`.

```r
desparse_res$LowerBound
```

```
##       95%
## 0.5097611
```

```r
desparse_res$UpperBound
```

```
##      95%
## 2.061519
```

`betaMax` is the bias-reduced maximum beta estimate.

```r
desparse_res$betaMax
```

```
## [1] 0.8820112
```

`betaEst` contains the beta estimate for each subgroup.

```r
desparse_res$betaEst
```

```
## [1] 0.3232306 1.2856400
```

`op` is the cross-validated optimal tuning.

```r
desparse_res$op
```

```
## [1] 0.3333333
```

Bootstrap-calibrated R-Split

Bootstrap iterations are recommended to be B = 200, BB = 1000. Here we use B = 5 and BB = 10 as demo. The tuning parameter is selected through `cvSplit`.

```r
rsplit_res <- BSSplitLasso(y = Y,
                           x = x,
                           r = r,
                           G = w.index,
                           B = 5, BB = 10)
```

Result summary

```r
rsplit_res$LowerBound
```

```
##       95%
## 0.7358887
```

```r
rsplit_res$UpperBound
```

```
##      95%
## 1.346438
```

`betaMax` is the bias-reduced maximum beta estimate.

```r
rsplit_res$betaMax
```

```
## [1] 0.9857366
```

`betaEst` contains the beta estimate for each subgroup.

```r
rsplit_res$betaEst
```

```
## [1] -0.2233648  1.0411635
```

`modelSize` contains the R-split model size for each bootstrap iteration.

```r
summary(rsplit_res$modelSize)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    5.00    5.25    6.00    5.70    6.00    6.00
```

`op` is the cross-validated optimal tuning paramter

```r
rsplit_res$op
```

```
## [1] 0.03333333
```