A demonstration of the SIS package

Yang Feng, Arce Domingo

2023-06-29

We provide a detailed demo of the usage for the `SIS` package. This package implements the sure independence screening algorithm.

Sure Independence Screening

Installation

`SIS` can be installed from Github.

```
library(devtools)
devtools::install_github("statcodes/SIS", force=TRUE)
```

Then we can load the package, as well as other relevant packages:

```
library(SIS)
library(pROC)
#> Type 'citation("pROC")' for a citation.
#>
#> Attaching package: 'pROC'
#> The following objects are masked from 'package:stats':
#>
#>     cov, smooth, var
```

Quickstart

We will show in this section how to use the SIS package.

First, we generate a linear model with the first five predictors as signals.

```
set.seed(0)
n = 400
p = 50
rho = 0.5
corrmat = diag(rep(1 - rho, p)) + matrix(rho, p, p)
corrmat[, 4] = sqrt(rho)
corrmat[4, ] = sqrt(rho)
corrmat[4, 4] = 1
corrmat[, 5] = 0
corrmat[5, ] = 0
corrmat[5, 5] = 1
cholmat = chol(corrmat)
x = matrix(rnorm(n * p, mean = 0, sd = 1), n, p)
x = x %*% cholmat

# gaussian response
set.seed(1)
b = c(4, 4, 4, -6 * sqrt(2), 4/3)
y = x[, 1:5] %*% b + rnorm(n)
```

SIS screening without iteration

Next, we apply the SIS screening without iteration.

```
# SIS without regularization
model10 = SIS(x, y, family = "gaussian", iter = FALSE)

# Getting the final selected variables after regularization step
model10$ix
#> [1] 3 2 1 5 4

# Getting the ranked list of variables from the screening step
model10$sis.ix0
#>  [1]  3  2  1  5 29  9 31 47 24 30 46  7 11 25 43 35 42 18 48  6 21 32 10 50 22
#> [26]  4  8 44 34 36 26 41 33 17 19 16 13 14 12 23 45 49 39 20 15 27 38 28 40 37

# The top 10 ranked variables from the screening step
model10$ix0[1:10]
#>  [1]  3  2  1  5 29  9 31 47 24 30
```

Iterative SIS

Now, we apply the SIS screening with iteration and combined with SCAD penalty.

```
# SIS with regularization
model11 = SIS(x, y, family = "gaussian", penalty = "SCAD", iter = TRUE)
```

```
#> Iter 1 , screening:  3 2 1 5 29 9 31 47 24 30 46 7 11 25 43 35 42 18 48 6 21 32 10 50 22 4 8 44 34
#> Iter 1 , selection:  3 2 1 5 4
#> Iter 1 , conditional-screening:  24 13 45 42 16 37 27 25 39 47 7 26 23 6 35 21 44 36 31 30 11 40 19
#> Iter 2 , screening:  3 2 1 5 4 24 13 45 42 16 37 27 25 39 47 7 26 23 6 35 21 44 36 31 30 11 40 19 48
#> Iter 2 , selection:  3 2 1 5 4
#> Model already selected

# Getting the final selected variables
model10$ix
#> [1] 3 2 1 5 4

# The top 10 ranked variables for the final screening step
model11$ix0[1:10]
#>  [1]  3  2  1  5  4 24 13 45 42 16

# The top 10 ranked variables for each screening step
lapply(model11$ix_list, f <- function(x) {
    x[1:10]
})
#> [[1]]
#>  [1]  3  2  1  5 29  9 31 47 24 30
#>
#> [[2]]
#>  [1]  3  2  1  5  4 24 13 45 42 16
```

Screening with binary response

```
set.seed(2)
feta <- x[, 1:5] %*% b
fprob <- exp(feta)/(1 + exp(feta))
y <- rbinom(n, 1, fprob)
model21 <- SIS(x, y, family = "binomial", tune = "bic")
#> Iter 1 , screening:  1 3 2 5 29 30 31 25 17 13 14 32 9 18 27 26 4 48 43 44 47 7 23 46 21 12 6 20 42
#> Iter 1 , selection:  1 3 2 5 4
#> Iter 1 , conditional-screening:  11 7 17 27 46 41 26 49 47 9 24 14 40 10 36 42 34 8 28 39 21 20 18
#> Iter 2 , screening:  1 3 2 5 4 11 7 17 27 46 41 26 49 47 9 24 14 40 10 36 42 34 8 28 39 21 20 18 31
#> Iter 2 , selection:  1 3 2 5 4
#> Model already selected

# Getting the final selected variables
model21$ix
#> [1] 1 3 2 5 4

# The top 10 ranked variables for the final screening step
model11$ix0[1:10]
#>  [1]  3  2  1  5  4 24 13 45 42 16

# The top 10 ranked variables for each screening step
lapply(model11$ix_list, f <- function(x) {
    x[1:10]
})
```

```
#> [[1]]
#>  [1]  3  2  1  5 29  9 31 47 24 30
#> 
#> [[2]]
#>  [1]  3  2  1  5  4 24 13 45 42 16
```

Screening with Cox model for survival data

```
set.seed(4)
b <- c(4, 4, 4, -6 * sqrt(2), 4/3)
myrates <- exp(x[, 1:5] %*% b)
Sur <- rexp(n, myrates)
CT <- rexp(n, 0.1)
Z <- pmin(Sur, CT)
ind <- as.numeric(Sur <= CT)
y <- survival::Surv(Z, ind)
model41 <- SIS(x, y, family = "cox", penalty = "lasso", tune = "bic", varISIS = "aggr",
    seed = 41)
#> Iter 1 , screening:  1 2 3 4 5 6 9 18 21 24 29 30 31 34 35 37 40 41 42 44 47 48
#> Iter 1 , selection:  1 2 3 4 5 6 9 18 21 24 29 30 31 35 37 40 41 42 44 47 48
#> Iter 1 , conditional-screening:  7 8 10 11 12 13 14 15 16 17 19 20 22 23 25 26 27 28 32 33 34 36 38
#> Iter 2 , screening:  1 2 3 4 5 6 9 18 21 24 29 30 31 35 37 40 41 42 44 47 48 7 8 10 11 12 13 14 15
#> Iter 2 , selection:  1 2 3 4 5 6 9 18 29 31 40 42 47 12 14 22 23 27 28 33 34 39 43 49 50
#> Iter 2 , conditional-screening:  7 8 10 11 13 15 16 17 19 20 21 24 25 26 30 32 35 36 37 38 41 44 45
#> Iter 3 , screening:  1 2 3 4 5 6 9 18 29 31 40 42 47 12 14 22 23 27 28 33 34 39 43 49 50 7 8 10 11
#> Iter 3 , selection:  1 2 3 4 5 6 9 18 29 31 40 42 47 12 14 22 23 27 28 33 34 39 43 49 50
#> Model already selected
model42 <- SIS(x, y, family = "cox", penalty = "lasso", tune = "bic", varISIS = "cons",
    seed = 41)
#> Iter 1 , screening:  1 2 3 4 5 6 7 9 11 15 16 17 19 20 21 22 23 24 29 30 31 32 34 35 37 39 41 44 45
#> Iter 1 , selection:  1 2 3 4 5 6 9 22 23 24 29 31 34 37 39 46 47 48 50
#> Iter 1 , conditional-screening:  7 8 10 11 12 13 14 15 16 17 18 19 20 21 25 26 27 28 30 32 33 35 36
#> Iter 2 , screening:  1 2 3 4 5 6 9 22 23 24 29 31 34 37 39 46 47 48 50 7 8 10 11 12 13 14 15 16 17
#> Iter 2 , selection:  1 2 3 4 5 6 9 22 23 29 31 34 39 47 50 12 14 18 27 28 33 40 42 43 49
#> Iter 2 , conditional-screening:  7 8 10 11 13 15 16 17 19 20 21 24 25 26 30 32 35 36 37 38 41 44 45
#> Iter 3 , screening:  1 2 3 4 5 6 9 22 23 29 31 34 39 47 50 12 14 18 27 28 33 40 42 43 49 7 8 10 11
#> Iter 3 , selection:  1 2 3 4 5 6 9 22 23 29 31 34 39 47 50 12 14 18 27 28 33 40 42 43 49
#> Model already selected
model41$ix
#>  [1]  1  2  3  4  5  6  9 18 29 31 40 42 47 12 14 22 23 27 28 33 34 39 43 49 50
model42$ix
#>  [1]  1  2  3  4  5  6  9 22 23 29 31 34 39 47 50 12 14 18 27 28 33 40 42 43 49
```

Screening with multi-categorical response

```
y <- as.factor(iris$Species)
noise <- matrix(rnorm(nrow(iris) * 200), nrow(iris), 200)
x <- cbind(as.matrix(iris[, -5]), noise)
```

```
model21 <- SIS(x, y, family = "multinom", penalty = "lasso")
#> Iter 1 , screening:  4 3 1 2
#> Iter 1 , selection:  4 3 1 2
#> Iter 1 , conditional-screening:  112 66 106
#> Iter 2 , screening:  4 3 1 2 112 66 106
#> Iter 2 , selection:  4 3 1 2 112 66 106
#> Maximum number of variables selected

# Getting the final selected variables
model21$ix
#> [1]   4   3   1   2 112  66 106

# The top 10 ranked variables for the final screening step
model21$ix0[1:10]
#>  [1]   4   3   1   2 112  66 106  NA  NA  NA

# The top 10 ranked variables for each screening step
lapply(model21$ix_list, f <- function(x) {
    x[1:10]
})
#> [[1]]
#>  [1]   4   3   1   2  10  72  86 199  14  52
#>
#> [[2]]
#>  [1]   4   3   1   2 112  66 106  46  49 100
```

Real data example (leukemia): Iterative Sure Independence Screening paired with elastic-net

```
# Loading data: Gene expression data from 7129 genes and 38 patients with acute leukemias (27 in class

load('leukemia.train.RData')
load('leukemia.test.RData')

# Getting the predictors and response variables
x_train <- as.matrix(leukemia.train[,1:7129])
y_train <- leukemia.train[,7130]
x_test <- as.matrix(leukemia.test[,1:7129])
y_test <- leukemia.test[,7130]


# Calling SIS and calculating the time taken for the algorithm to run
start.time <- Sys.time()
sis <- SIS(x_train, y_train, family = "binomial", penalty='enet',
           tune='cv', nfolds = 10, iter = TRUE, iter.max = 10,
           seed = 123, nsis=dim(x_train)[1]/2, standardize = TRUE,
           boot_ci=TRUE)
#> Iter 1 , screening:  3320 4847 2020 1745 5039 1834 461 4196 3847 2288 1249 6201
#> Iter 1 , selection:  3320 4847 2020 1745 5039 1834 461 4196 3847 2288 1249 6201
#> Iter 1 , conditional-screening:  5954 1779 3328 2674 5788 4444 4025
#> Iter 2 , screening:  3320 4847 2020 1745 5039 1834 461 4196 3847 2288 1249 6201 5954 1779 3328 2674
#> Iter 2 , selection:  3320 4847 2020 1745 5039 1834 461 4196 3847 2288 1249 6201 5954 1779 3328 2674
#> Maximum number of variables selected
#> [1] 10
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
#> Time difference of 6.906336 secs
```

```
# Getting the AUC in the test set
pred <- predict(sis, x_test, type = "class")
auc(pred[, 1], y_test)
#> Setting levels: control = 0, case = 1
#> Setting direction: controls < cases
#> Area under the curve: 0.9348

# Getting the confidence intervals of the selected variables,
# calculated using bootstrap
sis$cis[2:12, c(1, 5, 6, 7)]
#>      var       Est CI_low_perc CI_up_perc
#> 2  V1249 3.7944196   1.5332677  5.7711607
#> 3  V1745 2.8085418   1.7013228  6.8591982
#> 4  V1779 5.4985536   1.9600411  7.8946735
#> 5  V1834 3.1819491   2.0632329  8.4217949
#> 6  V2020 5.2269255   2.7869270  8.5800960
#> 7  V2288 1.6004097   1.1659975  4.5920686
#> 8  V2674 0.7062523   0.4555075  0.9738599
#> 9  V3320 4.4277800   2.5764195 11.1584262
#> 10 V3328 0.8707467   0.5699275  1.1660537
#> 11 V3847 4.9828435   2.4017975  9.2489171
#> 12 V4025 1.0050921   0.4986696  1.9027701
```

Real data example (prostate cancer): Iterative Sure Independence Screening paired with elastic-net

```
# Loading data: Gene expression data from 12600 genes and 52 patients with prostate tumors and 50 norma
load('prostate.train.RData')
load('prostate.test.RData')

# Getting the predictors and response variables
x_train <- as.matrix(prostate.train[,1:12600])
y_train <- prostate.train[,12601]
x_test <- as.matrix(prostate.test[,1:12600])
y_test <- prostate.test[,12601]

# Calling SIS and calculating the time taken for the algorithm to run
start.time <- Sys.time()
sis <- SIS(x_train, y_train, family = "binomial", penalty='enet',
           nfolds = 10, iter = TRUE, iter.max = 10,tune='cv',
           seed = 123, nsis=dim(x_train)[1]/2, standardize = TRUE,
           boot_ci=TRUE)
#> Iter 1 , screening:  6185 8965 4365 10138 6866 9172 8123 7067 12148 9050 9850 10494 8850 10956 288 1
#> Iter 1 , selection:  6185 8965 4365 10138 6866 9172 8123 7067 12148 9050 9850 10494 8850 10956 288 1
#> Iter 1 , conditional-screening:  2515 1253 11091 512 1126 10213 8205 7217 3459 10615 2245 10137 7293
#> Iter 2 , screening:  6185 8965 4365 10138 6866 9172 8123 7067 12148 9050 9850 10494 8850 10956 288 1
#> Iter 2 , selection:  6185 8965 4365 10138 6866 9172 8123 7067 12148 9050 9850 10494 8850 10956 288 1
#> Iter 2 , conditional-screening:  10949 9319
#> Iter 3 , screening:  6185 8965 4365 10138 6866 9172 8123 7067 12148 9050 9850 10494 8850 10956 288 1
#> Iter 3 , selection:  6185 8965 4365 10138 6866 9172 8123 7067 12148 9050 9850 10494 8850 10956 288 1
#> Maximum number of variables selected
#> [1] 10
end.time <- Sys.time()
time.taken <- end.time - start.time
time.taken
#> Time difference of 1.838951 mins
```

```
# Getting the AUC in the test set
pred <- predict(sis, x_test, type = "class")
auc(pred[, 1], y_test)
#> Setting levels: control = 0, case = 1
#> Setting direction: controls < cases
#> Area under the curve: 0.95

# Getting the confidence intervals of the selected variables,
# calculated using bootstrap
sis$cis[2:12, c(1, 5, 6, 7)]
#>       var       Est CI_low_perc CI_up_perc
#> 2  V10137 0.8958555   0.7100986  1.0000000
#> 3  V10138 1.2478882   1.0000000  1.7949098
#> 4  V10213 0.7086252   0.4955523  0.9270748
#> 5  V10494 1.3509784   1.1300804  2.0666220
#> 6  V10537 0.5985402   0.4590505  0.8176723
#> 7  V10553 1.0929825   1.0000000  1.5264328
#> 8  V10615 1.1520628   1.0000000  1.3313993
#> 9  V10949 1.9577615   1.1109656  2.8109651
#> 10 V10956 1.6791987   1.2712965  2.2355798
#> 11 V11091 0.6663747   0.5081175  0.8840754
#> 12  V1126 1.3880061   1.0723830  1.9140236
```