A demonstration of the AVGAS package

Lillian Li

A short demo of the usage for the AVGAS package. This package implements the Genetic Algorithm for high-dimensional linear regression models with two-way interaction effects under Strong, Weak, or No heredity condition.

Introduction + The Model Setup + Genetic Algorithm + The fitness function: ABC criterion Installation * Examples on Simulated Data

Introduciton

The Model Setup

Suppose the dataset consists of $(\mathbf{X}, \mathbf{Y})$, where $\mathbf{Y} = (y_1, \ldots, y_n)^T$ is the $n \times 1$ response vector, and $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_p)$ is the $n \times p$ design matrix with n observations and p covariates. In high-dimensional linear regression with interactions, suppose the data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$are *i.i.d.* from the model

$$Y = \beta_1 X_1 + \cdots + \beta_p X_p + \beta_{1,2} X_1 X_2 + \cdots + \beta_{p-1,p} X_{p-1} X_p + \epsilon, \tag{1}$$

where $\epsilon \sim N(0, \sigma^2)$ is the noise, β_i is the coefficient of the main effect X_i, and $\beta_{i,j}$ is the coefficient of the interaction effect $X_i X_j$ (product of X_i and X_j). We restate strong, weak, and no hierarchical structures. **Strong heredity** assumes that if an interaction effect has non-zero effect, then both of its corresponding main effects should have non-zero effect. That is, if $\beta_{i,j} \neq 0$, then $\beta_i \neq 0$ and $\beta_j \neq 0$ for $1 \leq i \neq j \leq p$. **Weak heredity** requires that if $\beta_{i,j} \neq 0$, then $\beta_i \neq 0$ or $\beta_j \neq 0$ for $1 \leq i \neq j \leq p$. **No heredity** assumes no relationship between $I(\beta_{i,j} \neq 0)$, $I(\beta_i \neq 0)$, and $I(\beta_j \neq 0)$.

Genetic Algorithm

Genetic Algorithm (GA), a stochastic search algorithm inspired by biological evolution and natural selection. A genetic algorithm in variable selection is a mathematical optimization technique that identifies the optimal or a near-optimal subset of solutions for a given problem. This process first creates a population/generation of potential models and then evaluates their performances based on a predefined criterion usually called the fitness or objective function. Then an evolution process (such as crossover and mutation) is conducted in each generation to generate new models. Better candidate models (with larger fitness values) will be selected during the evolution process through a series of generations. The best model of the last generation will be selected as the final model. Genetic algorithms can be particularly useful in variable selection when it is impossible to list all the candidate models.

The fitness or objective function: ABC criterion The criterion ABC is defined as follows:

$$ABC(I) = \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i^I \right)^2 + 2 r_I \sigma^2 + \lambda \sigma^2 C_I, \tag{2}$$

where C_I is the descriptive complexity term of model I, $\hat{\mathbf{Y}}^I$ is the projection of the response vector $\mathbf{Y}$ onto the column space of the design matrix of the model I with rank r_I. The terms $\lambda \geq 5.1/\log 2$ is a positive constant, and $\{\sigma$ is the standard deviation of the noise term which in practice, is unknown. In such a case, we estimate σ by fitting a lasso regression on the full model.

For details, please refer to the paper by *Ye C, Yang Y (2019). "High-dimensional adaptive minimax sparse estimation with interactions." IEEE Transactions on Information Theory, 65(9), 5367–5379. doi:10.1109/TIT.2019.2913417*

Installation

AVGAS is now available on CRAN and can be easily instally by one-line code.

```
install.packages("AVGAS")
```

Then load the package:

```
library(glmtrans)
```

Examples on Simulated Data

We generate data from the model setup under $(n = 400, p = 100, \sigma = 0.1)$. The covariates $X_1, \ldots, X_{400}$ are generated i.i.d. from a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \Sigma)$, where the (i, j)-th entry of the covariance matrix is $\Sigma_{jk} = 0.5^{|j-k|}$ for $1 \leq j, k \leq 100$. Denote A as the index sets of the main effects, and B as the index set of interaction effects. Concatenate the coefficients as β. Consider the following data-generating models:

A={1,2,...,10}, and B= {(1,2),(1,3),(2,3),(2,5),(3,4),(6,8),(6,10),(7,8),(7,9),(9,10)} with true regression coefficients $\beta = (3, 3, 3, 3, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1)^T$.

```
set.seed(0)
p <- 100
interaction.ind <- t(combn(100,2))
sigmajk <- matrix((0.5)^(abs(outer(1:100, 1:100, "-"))),
                  nrow = 100, ncol = 100)
X <- mvtnorm::rmvnorm(400, rep(0,nrow(sigmajk)), sigmajk)
epl <- rnorm(400,0,0.1)
beta.est <- matrix(0, nrow = 20, ncol=1)
beta.est[c(1:20),] <- c(rep(3,5), rep(2,10), rep(1,5))
y <- Extract(X, c(1:10,101,102,200,202,298,587,589,680,681,865),
             interaction.ind)%*%beta.est+epl
y <- as.numeric(y)
AVGAS(X, y, nmain.p=100, r1=10, r2=10, heredity = "Strong",
      interaction.ind = interaction.ind, q=40, take = 1)
```

```
## $final_model
## $final_model[[1]]
##  [1] "X.1"     "X.2"     "X.3"     "X.4"     "X.5"     "X.6"     "X.7"
##  [8] "X.8"     "X.9"     "X.10"    "X.1X.2"  "X.1X.3"  "X.2X.3"  "X.2X.5"
## [15] "X.3X.4"  "X.6X.8"  "X.6X.10" "X.7X.8"  "X.7X.9"  "X.9X.10"
##
##
## $cleaned_candidate_model
##
##     "X.1"     "X.2"     "X.3"     "X.4"     "X.5"     "X.6"     "X.7"     "X.8"
```

```
##
##     "X.9"    "X.10"  "X.1X.2"  "X.1X.3"  "X.2X.3"  "X.2X.5"  "X.3X.4"  "X.6X.8"
##                                           ABCscore
## "X.6X.10"  "X.7X.8"  "X.7X.9" "X.9X.10" "37.7047"
```

Plotting the interaction effects

```
bbb <- initial( X, y, heredity = "Strong", nmain.p=100, interaction.ind = interaction.ind, r1 = 10, r2=
interpool <- bbb$InterRank
ccc <- as.data.frame(interpool)
inter <- ccc[,1]
scores <- ccc[,2]
gp <- ggplot2::ggplot(ccc,ggplot2::aes(x = stats::reorder(as.character(inter),
                                          + as.numeric(scores)),
                                       y = as.numeric(scores))) +
     geom_point() +
  ggplot2::theme(axis.text.x = ggplot2::element_text(angle = 90)) +
  ggplot2::labs(y = "ABC Scores", x = "Interaction Index")
gp
```