Package 'swa'

October 18, 2017

Title Subsampling Winner Algorithm for Classification

Version 0.8.1

Description This algorithm conducts variable selection in the classification setting. It repeatedly subsamples variables and runs linear discriminant analysis (LDA) on the subsampled variables. Variables are scored based on the AUC and the t-statistics. Variables then enter a competition and the semi-finalist variables will be evaluated in a final round of LDA classification. The algorithm then outputs a list of variable selected. Qiao, Sun and Fan (2017) <http://people.math.binghamton.edu/qiao/swa.html>.

Depends R (>= 3.3.1), ROCR, reshape, ggplot2

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests testthat

NeedsCompilation no

Author Xingye Qiao [aut, cre],
Jiayang Sun [aut],
Yiying Fan [aut]

Maintainer Xingye Qiao <qiao@math.binghamton.edu>

Repository CRAN

Date/Publication 2017-10-18 17:10:37 UTC

R topics documented:

swa *Subsampling Winner Algorithm for Variable Selection in the Classification Setting*

Description

This function conducts variable selection in the classification setting. The algorithm repeatedly subsamples variables and runs linear discriminant analysis (LDA) via linear regression based on the subsampled variables. Variables are scored based on the AUC and the t-statistics in linear regression. Variables then enter a competition and the semi-finalists will be used in a final round of LDA (or linear regression.) The semi-finalists and the final linear model are returned.

Usage

```
swa(X, y, S = c(3, 5, 7, 10, 15, 20, 30, 40), q = 20, m = 500,
  MPplot = FALSE, screening = TRUE)
```

Arguments

X	The p by n design matrix where p is the dimension and n is the sample size.
y	The binary class labels (0 or 1.)
S	Size(s) of subsamplings. This could be a set of candidate sizes, or a single size. In the former case, all sizes will be tried. A vector of default values is c(3,5,7,10,15,20,30,40).
q	Number of semi-finalists. The default is 20.
m	Number of subsamplings. The default is 500. One may increase the value of m to see if the outcomes have been stabilized.
MPplot	Boolean input showing whether the multi-panel diagnostic plot is drawn. The default is FALSE.
screening	Boolean input showing whether the variables are pre-screened based on marginal variance. The default is TRUE.

Value

A list with the indices of the semi-finalists, summary of the final regression model, and summary of the stepwise selection of the final model.

Examples

```
set.seed(1)
x <- matrix(rnorm(20*100), ncol=20)
b = c(0.5,1,1.5,2,3);
y <- x[,1:5]%*%b + rnorm(100)
y <- as.numeric(y<0)
X <- t(x)
swa(X,y,c(3,5,10,15,20),5,500,MPplot = TRUE) ;
```

```
## The MP plots show that the upper arm sets, i.e., the points above the elbow point of each
## panel plot, started to be stabilized at s = 10. Thus, we fix s = 10 below.
sum_swa <- swa(X,y,10,5,500)
sum_swa

## The final model perfectly recovers the five true variables. The stepwise selection removes
## X1, which has the weakest signal (hence is not expected to be recovered.) We next check the
## results with a larger m = 1000.
sum_swa <- swa(X,y,10,5,1000)
sum_swa
# It now captured X2,X3,X4,X5, and also a noise variable X6 that is not as significant as the
## former 4 covariates X2-X4. We next look at the estimates of all 20 coefficients including
## those not selected.
coef_swa = numeric(20)
column_index = as.numeric(substring(rownames(sum_swa$step_sum$coef[-1,]),2,5))
coef_swa[column_index] <- sum_swa$step_sum$coef[-1,1]

#### Compare the coefficients from the SWA on all covariates with the coefficients from the
## oracle LDA. We first calculate the coefficients from the oracle LDA.
n = 100
n0 = sum(y==0)
n1 = sum(y==1)
yy=y
yy[y==0] <- -n/n0 ;
yy[y==1] <- n/n1 ;
X = X[1:5,]
X = X - apply(X,1,median);
X = X/apply(X,1,mad) ;
designX = t(X) ;
colnames(designX) <- paste('X',1:5,sep='') ;
dataXY = data.frame(cbind(yy,designX)) ; dim(dataXY) ;
LDA <- lm(yy ~ ., data = dataXY)
sum_LDA <- summary(LDA)
coef_LDA = numeric(20)
coef_LDA[as.numeric(substring(rownames(sum_LDA$coef[-1,]),2,5))] <- sum_LDA$coef[-1,1]
####  Compare the coefficients using correlation.
cor(coef_swa,coef_LDA); rbind(coef_swa, coef_LDA);
####  There is an excellent correlation between the coefficients recovered from the SWA on
## the full data and the coefficients from the benchmark, ie., the oracle LDA.

####  Next we try on a set of smaller coefficients that should be harder to recover from
## the noisy data
x <- matrix(rnorm(20*100), ncol=20)
b = c(0.3,0.6,0.9,1.2,1.5);
y <- x[,1:5]%*%b + rnorm(100)
y <- as.numeric(y<0)
X <- t(x)
swa(X,y,10,5,500) ;
####  SWA does a perfect job in recovering the true variables.
```

Index