Bagged **L**ogistic Regr**ess**ion (BLESS) Tutorial

Kyle Gardiner, Xuekui Zhang, Li Xing

2024-01-31

Introduction

Bagged **L**ogistic Regr**ess**ion (BLESS) in an ensemble algorithm that is inspired by the bagging procedures used by other machine learning algorithms like Random Forest. The algorithm works by bootstrapping data to fit logistic regression as its base learner over multiple iterations. From these iterations, votes are aggregated to provide a list of biomarkers (i.e SNPs) ranked with respect to their association with the outcome of interest.

Example Data

Example data has been provided (`data.rda`) and can be found in the GitHub repository **here**. The dimensions of this data are 197 x 1011. The data frame contains 197 observations with 1000 features/predictors/biomarkers (`X.1`-`X.1000`), 10 covariates (`Cov.1`-`Cov.10`), and the binary outcome variable (`Y`)

```
load("data.rda")
BLESS_data[1:5, 1:5]
```

```
##   X.1 X.2 X.3 X.4 X.5
## 1   2   2   1   2   1
## 2   1   1   2   2   1
## 3   1   1   1   1   2
## 4   2   2   0   1   1
## 5   2   2   2   1   1
```

```
BLESS_data[1:5, c(1001:1004, 1011)]
```

```
##   Cov.1 Cov.2      Cov.3      Cov.4 Y
## 1 78.62     0 -0.0262782 -0.0160650 0
## 2 71.51     0 -0.0249967 -0.0195297 0
## 3 77.01     1 -0.0247467 -0.0130260 1
## 4 69.00     0 -0.0260312 -0.0121795 0
## 5 67.66     1 -0.0264302 -0.0201746 0
```

Model Fitting

Data Setup

First, we need to setup an empty dataframe to store the identified biomarkers from each iteration of the logistic regression process

```
predictor.df<-data.frame(Predictor=character(0))
```

BLESS Algorithm

The data mentioned above is used as an example to showcase the procedure of applying BLESS. For BLESS, we create a loop that will subsample the data (both observations and predictors), build a logistic regression model using the selected data, identify which predictors are significant, and store the significant predictors in the empty dataframe previously made.

```
for(ii in 1:5000){ # 5000 is the number of iterations performed.
                   # This number can be changed depending on the data being analyzed

  # setting seed each iteration for reproducibility
  set.seed(ii)

  # Subsampling Observations and Predictors
  ## Randomly subsampling 90% of observations
  obs.id<-sample(nrow(BLESS_data),round(nrow(BLESS_data)*0.9))

  ## Randomly subsampling 30 predictors
  ### 30 predictors is roughly square root the numbers of predictors
  sample.SNPs<-sample(ncol(BLESS_data[,-1001]),30)

  ## Combining selected observations and predictors with the outcome Y
  sub.data<-BLESS_data[obs.id,c(sample.SNPs,1011)]


  # Fitting a Logistic Model with Each Subset Data
  myfit<-glm(Y~., data=sub.data, family="binomial")


  # Determining Which Predictors Have p-values < 0.05
  selected.predictor.id<-which(coef(summary(myfit))[-1,4]<0.05)

  # Extracting the Names of the Selected Predictors
  selected.predictor<-row.names(coef(summary(myfit))[-1,])[selected.predictor.id]

  # Creating a Temporary Dataframe with the Selected Predictors and
  # Binding Them by Row to get the Selected Predictors Over 5000 Iterations
  temp.predictor.df<-data.frame(Predictor=selected.predictor)
  predictor.df<-rbind(predictor.df,temp.predictor.df)
}
```

Now we will sort the frequency of selected predictors and convert them into a ranked list.

```
# Sorting the Frequency of Selected Predictors in Descending Order
char.counts<-sort(table(predictor.df),decreasing=TRUE)

# Converting the `char.counts` into a dataframe
predictor.freq_5000<-as.data.frame(char.counts)
```

Finally, we can display the top 10 ranked predictors over all the iterations.

```
head(predictor.freq_5000)
```

```
##   Predictor Freq
## 1     X.404  135
## 2     X.366  127
## 3      X.26  126
## 4       X.4  126
## 5     X.324  125
## 6      X.25  122
```