A study of bootstrapping on developing classification model

Xinlei Chen, Guojing Wu and Yujing Yao

April 19, 2019

Abstract

This report discusses a study of bootstrap to develop classification model based on the proteins expression levels. Our goal is to build a predictive model based on logistic regression to facilicate down syndrome diagnosis, and we compared methods including and Pathwise Coordinate Descent with regularized logistic regression and smoothed bootstrap estimation. Our result shows that .....

Background

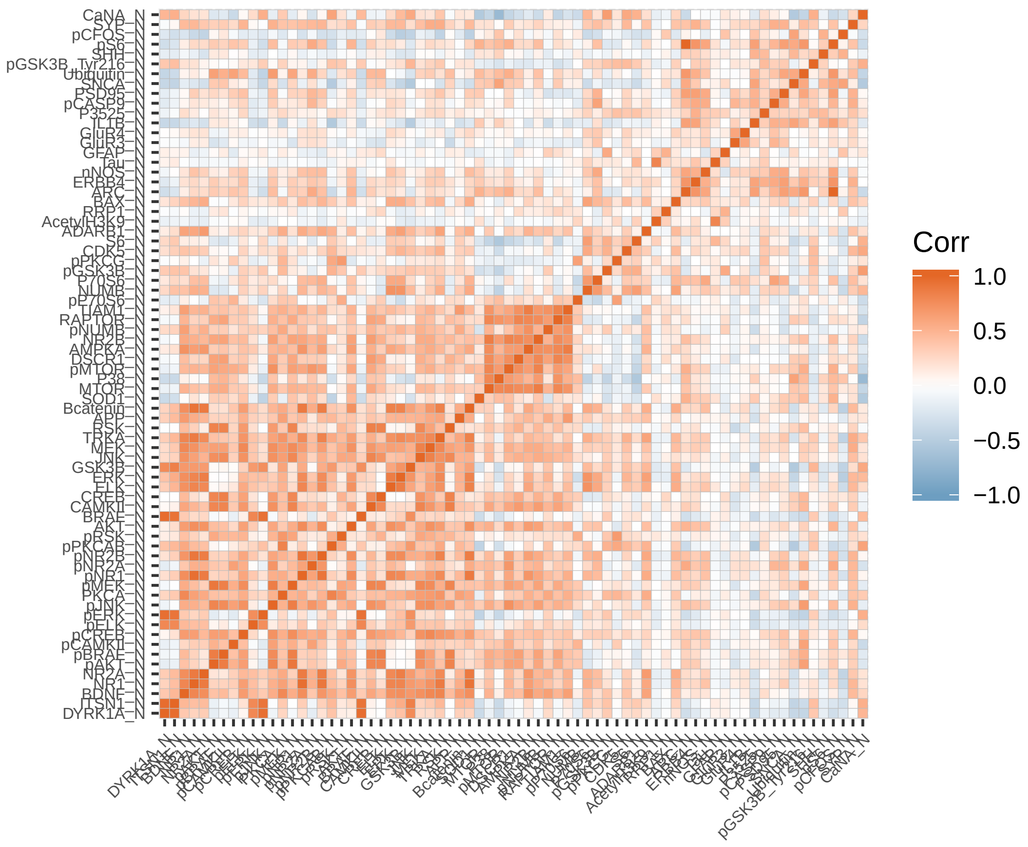
The data *Down.csv* consists of the expression levels of 77 proteins/protein modifications that produced detectable signals in the nuclear fraction of cortex. It has 1080 rows and 79 columns. The first column **MouseID** identifies individual mice; The column **2-78** are values of expression levels of 77 proteins. Column 79 indicates whether the mouse is a control or has Down syndrome. The goal is to develop classification model based on the proteins expression levels.

We found that data is missing for some of the covariates, so we deleted those high missing rate(>20%). For those covariates with missing rate < 20%, we assumed them to be missing completely at random(MCAR), so we would analyze the complete cases in the following.

!!!need a missing plot here

!!! explain why we need to apply regulized method: singular

Figure 1: multicollinearity plot of the dataset



Method

Logistic Regression

Let y be the vector of n response random variable, X denote the $n \times p$ design matrix (X_i denote the ith row) and β denote the $p \times 1$ coefficient. Let $\eta = E(y) = X\beta$ and given the link function as $g(\eta) = \log \frac{\eta}{1-\eta}$, we have the logistic regression model written as:

$$\log(\frac{\eta}{1-\eta}) = X\beta$$

The likelihood of this logistic regression is:

$$L(\beta; X, y) = \prod_{i=1}^{n} \{(\frac{\exp(X_i\beta)}{1+\exp(X_i\beta)})^{y_i}(\frac{1}{1+\exp(X_i\beta)})^{1-y_i}\}$$

Maximizing the likelihood is equivalent to maximizing the log likelihood:

$$l(\beta) = \sum_{i=1}^{n} \{y_i(X_i\beta) - \log(1+\exp(X_i\beta))\}$$

Pathwise Coordinate Descent with regularized logistic regression

Regularization is the common variable selection approaches for high-dimensional covariates. The best known Regularization is called LASSO, which is to add $L1$-penalty to the objective function. In the context of logistic regression, we are aiming to maximize the penalized log likelihood:

$$\max_{\beta \in \mathbb{R}^{p+1}} \frac{1}{n} \sum_{i=1}^{n} \{y_i(X_i\beta) - \log(1 + \exp(X_i\beta))\} - \lambda \sum_{j=0}^{p} |\beta_j|$$

for some $\lambda \geq 0$. Here the $x_{i,j}$ are standardized so that $\sum_i x_{i,j}/n = 0$ and $\sum_i x_{i,j}^2/n = 0$.

The Newton algorithm for maximizing the log likelihood amounts to interatively reweighted least squares. Hence if the current estimate of the parameter is $\tilde{\beta}$, we can form a quardratic approximation to the negative log likelihood by taylor expansion around the current estimate, which is:

$$f(\beta) = -\frac{1}{2n} \sum_{i=1}^{n} w_i(z_i - \sum_{j=0}^{p} x_{i,j}\beta_j)^2 + C(\tilde{\beta})$$

where

$$z_i = \tilde{\beta}_0 + x_i^T \tilde{\beta} + \frac{y_i - \tilde{p}(x_i)}{\tilde{p}(x_i)(1 - \tilde{p}(x_i))}, \text{working response}$$
$$w_i = \tilde{p}(x_i)(1 - \tilde{p}(x_i)), \text{working weights}$$

and $\tilde{p}(x_i)$ is evaluated at the current parameters, the last term is constant. The Newton update is obtained by minimizing the $f(\beta)$

The coordinate descent algorithm to solve the penalized weighted least squares problem

$$\min_{\beta \in \mathbb{R}^{p+1}} \{-f(\beta) + \lambda P(\beta)\}$$

The above amounts to a sequence of nested loops:

- outer loop: start with λ that all the coefficients are forced to be zerp, then decrement λ;
- middle loop: update the quardratic $f(\beta)$ using the current estimates of parameters;
- inner loop: run the coordinate descent algorithm on the penalized weighted least square problem.

In our problem, care is taken to avoid coefficients diverging in order to achieve fitted probabilities of 0 or 1 which is the warning message by the R package. When a probability with $\epsilon = 1e - 5$ of 1, we set it to 1, and set the weights to ϵ. 0 is treated similarly.

Smoothing bootstrap and inference

Classical statistical theory ignores model selection in assessing estimation accuracy. Here we consider bootstrap methods for computing standard errors and confidence intervals that take model selection into account. The methodology involves bagging, also known as bootstrap smoothing, to tame the erratic discontinuities of selection-based estimators.

Point estimation:

- First we need to prepare a couple of candidate models
- for each bootstrap in bootstrap with B times, select the best model and get estimates for the coefficient denoted as $t(y^*)$
- smooth $\hat{\mu} = t(y)$ by averaging over the bootstrap replications, defining

$$\tilde{\mu} = s(y) = \frac{1}{B} \sum_{i=1}^{B} t(y^*)$$

Bootstrap smoothing (Efron and Tibshirani 1996), a form of model averaging, is better known as "bagging" in the prediction literature.

smoothed interval In addition to the percentile confidence interval, the nonparametric delta-method estimate of standard deviation for $s(y)$ in the nonideal case

$$\tilde{sd}_B = [\sum_{i=1}^{n} \hat{cov}_j^2]^{1/2}$$

where

$$hatcov_j = \sum_{i=1}^{B} (Y_{ij}^* - Y_{.j}^*)(t_i^* - t_.^*)/B$$

with $Y_{.j}^* = \sum_{i=1}^{B} Y_{ij}^*/B$ and $t_.^* = \sum_{i=1}^{B} t_i^*/B = s(y)$.

Results

Estimation path and cross validation for Lasso

Fig. 2 shows us that as the λ increases, all the variable estimates of parameters shrink accordingly since we penalize all the parameters, though in some. When $\lambda = 0$, the result is the same as least square method and when λ is too large, all the estimates of parameters shrink to 0.

Model selection for smoothed bootstrap

We are still dealing with regularized lasso with this dataset, we wanted to apply smoothed bootstrap in order to get better estimates and inference result. Our potential model includes reguliized lasso with lasso penalty, ridge penalty, SCAD and MCP, ... about penalty, ridge penalty, SCAD and MCP, and we wanted to select the best one through bootstrap.

Cross validation for model prediction comparison

We used 5-fold cross-validation to compare two different models, one is lasso, the other is selected from the bootstrap.

```
       [,1]   [,2]
[1,] 0.0277 0.0459
```

Inference for smoothed bootstrap

Conclusions

References

1 Efron, Bradley. "Estimation and accuracy after model selection." Journal of the American Statistical Association 109.507 (2014): 991-1007.

Appendix A

```r
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE, comment = "")
library(tidyverse)
library(ggpubr) # common legend
library(kableExtra) # table
library(glmnet) # lasso
library(parallel)
library(doParallel)
library(foreach)
library(iterators)
library(ggcorrplot) # for correlation heatmap
library(ncvreg)# for SCAD and MCP
nCores <- 4
registerDoParallel(nCores)
set.seed(99999)
options(knitr.table.format = "latex")
oridata <- read.csv("Down.csv")
mydata <- oridata%>%
  dplyr::select(.,-c(BAD_N,BCL2_N,H3AcK18_N,EGR1_N,H3MeK4_N))%>%
  filter(complete.cases(.))
n <- dim(mydata)[1]
ncol <- dim(mydata)[2]
list <- c(2:(ncol - 1))
namesX <- names(mydata)[-c(1,ncol)]
# standardize
dataX <- do.call(cbind, lapply(list, function(x) (mydata[,x] - mean(mydata[,x]))/sd(mydata[,x])))
colnames(dataX) <- namesX
# design matrix
X <- data.frame(dataX) %>%
  mutate(., intercept = 1)
# response
y <- as.vector(ifelse(mydata[,ncol] == "Down syndrome", 1, 0))
dat <- list(y = y, X = as.matrix(X))
# check correlation
cor(dataX) %>%
  ggcorrplot(.,ggtheme = ggplot2::theme_gray,
             colors = c("#6D9EC1", "white", "#E46726"),
             tl.cex = 6)
### regulized logistic regression: middle loop and inner loop
sfun <- function(beta,lambda) sign(beta) * max(abs(beta)-lambda, 0)
reglogitlasso <- function(lambda, dat, start, tol=1e-10, maxiter = 200,...){
  p <- dim(dat$X)[2]
  n <- length(dat$y)
  betavec <- start
  i <- 0
  loglik <- 0
  prevloglik <- Inf
  res <- c(0, loglik, betavec)
  while (i < maxiter && abs(loglik - prevloglik) > tol && loglik < Inf) {
    i <- i + 1
    prevloglik <- loglik
    for (j in 1:p) {
      u <- dat$X %*% betavec
      expu <- exp(u)
      prob <- expu/(expu+1)
      w <- prob*(1-prob)
      w <- ifelse(abs(w-0) < 1e-5, 1e-5, w)
      z <- u + (dat$y-prob)/w
      znoj <- dat$X[,-j] %*% betavec[-j]
      betavec[j] <- sfun(mean(w*(dat$X[,j])*(z - znoj)), lambda)/(mean(w*dat$X[,j]*dat$X[,j]))
    }
    loglik <- sum(w*(z-dat$X %*% betavec)^2)/(2*n) + lambda * sum(abs(betavec))
    res <- rbind(res, c(i, loglik, betavec))}
  return(res)
}
intial = rep(0, dim(dat$X)[2])
# corres <- reglogitlasso(lambda = exp(-10), dat, start = intial,tol=1e-5)
# start from -1 to -10

## pathwise update: outer loop
path <- function(data,grid){
  start <- rep(0, dim(data$X)[2])
  betas <- NULL
  for (x in 1:length(grid)){
    cor.result <- reglogitlasso(lambda = grid[x],dat = data,start= start)
    lasbeta <- cor.result[nrow(cor.result),3:dim(cor.result)[2]]
    start <- lasbeta
    betas <- rbind(betas,c(lasbeta))
  }
  return(data.frame(cbind(grid,betas)))
}
# path.out <- path(dat,grid=exp(seq(-1,-10, length=100)))
# colnames(path.out) <- c("grid",colnames(X))
# path.plot <- path.out %>%
#   gather(key = par, value = estimate, c(2:dim(path.out)[2])) %>%
#   ggplot(aes(x = log(grid), y = estimate, group = par, col = par)) +
#   geom_line()+
#   ggtitle("Figure 1: A path of solutions with a sequence of descending lambda's") +
#   xlab("log(Lambda)") +
#   ylab("Estimate") +
#   theme(legend.position = "right",
#         legend.text = element_text(size = 6))
# path.plot
##### 5-fold cross-validation to choose beta lambda
cvresult <- function(dat,grid,K){
  n <- dim(dat$X)[1]
  folds <- sample(1:K, n, replace=TRUE)
  start <- rep(0, dim(dat$X)[2])
  cv.error <- vector()
  cv.se <- vector()
  for (x in 1:length(grid)) {
  cv.errors <- vector()
  for (i in 1:K){
    cor.result <- reglogitlasso(lambda = grid[x],dat = list(X=dat$X[folds!=i,],y=dat$y[folds!=i]),
                                start = start)
    lasbeta <- cor.result[nrow(cor.result),3:dim(cor.result)[2]]
    u <- as.matrix(dat$X[folds == i,]) %*% lasbeta
    expu <- exp(u)
    prob <- expu / (1 + expu)
    y <- as.vector(dat$y[folds==i])
    cv.errors[i] = mean((y-prob)^2)
  }
  start <- lasbeta
  cv.error[x] <- mean(cv.errors)
  cv.se[x] <- sqrt(var(cv.errors)/K)
  }
  return(cbind(grid,cv.error,cv.se))
}
# result <- cvresult(dat,grid=exp(seq(-1,-10, length=100)),K=5)
# best.lambda <- result[which.min(result[,2]),1]
# result <- data.frame(result)
# cv.plot <-
#     ggplot(result, aes(x=log(result$grid), y=result$cv.error)) +
#     geom_errorbar(aes(ymin=result$cv.error-result$cv.se, ymax=result$cv.error+result$cv.se),
#                   colour=1) +
#     geom_line() +
#     geom_point(size=0.8,colour = 4) +
#     ggtitle("Figure 2: Lasso regression by 5 fold cross validation")+
#     xlab("log(Lambda)") + ylab("MSE") +
#     geom_vline(xintercept = log(best.lambda),col=3,lty=3) +
#     annotate("text", log(best.lambda), 0.1, label = paste("best log(lambda) = ", round(log(best.lambd
# cv.plot

#### final estimation for beta
# finlasso <- as.matrix(path(dat,grid = exp(seq(-1,log(best.lambda), length=100))))
# colnames(finlasso) <- c("grid",colnames(X))
# lasso.beta <- finlasso[nrow(finlasso),2:dim(finlasso)[2]]
############ get the estimate and se by smmothing bootstrap
n <- length(y)
B <- 5000
mse <- function(model){
  temp <- predict(model,dataX)
  ynew <- exp(temp)/(1+exp(temp))
  mse <- sum((y-ynew)^2)/n
  return(mse)
}
taskFun <- function(){
  bootid <- sample(c(1:n),replace = T)
  model <- list()
  ynum <- vector()
  boot.x <- dataX[bootid,]
  boot.y <- y[bootid]
  for (j in 1:n) ynum[j] <- sum(j==bootid)
  las.lambda <- cv.glmnet(x=boot.x, y=boot.y,alpha=1, family="binomial")$lambda.min
  rid.lambda <- cv.glmnet(x=boot.x, y=boot.y, alpha=0,family="binomial")$lambda.min
  scad.lambda <- cv.ncvreg(X=boot.x, y=boot.y,penalty="SCAD",family="binomial")$lambda.min
  mcp.lambda <- cv.ncvreg(X=boot.x, y=boot.y,penalty="MCP", family="binomial")$lambda.min
  model[[1]] <- glmnet(x=boot.x, y=boot.y,alpha=1, family="binomial",lambda = las.lambda)
  model[[2]] <- glmnet(x=boot.x, y=boot.y,alpha=0, family="binomial",lambda = rid.lambda)
  model[[3]] <- ncvreg(X=boot.x, y=boot.y,penalty="SCAD", family="binomial",lambda = scad.lambda)
  model[[4]]<- ncvreg(X=boot.x, y=boot.y,penalty="MCP", family="binomial",lambda = mcp.lambda)
  min <- which.min(unlist(lapply(1:4, function(x)mse(model[[x]]))))[1]
  model.chosen <- min
  coef.chosen <- as.vector(coef(model[[min]]))
  subject.1 <- predict(model[[min]],dataX,type = "response")[1]
  return(cbind(model.chosen,subject.1,t(coef.chosen),t(ynum)))
}
# out <- foreach(i = 1:B, .combine = rbind) %dopar% taskFun()
# colnames(out) <- c("model","subject.1","intercpt",colnames(dataX),1:n)
# model <- data.frame(out)%>%
#   mutate(.,reg.model = ifelse(model ==1,"Lasso",
#                                        ifelse(model==2, "Ridge",
#                                               ifelse(model==3,"SCAD","MCP"))))%>%
#   dplyr::select(.,-c(model))%>%
#   group_by(.,reg.model)%>%
#   summarise(.,percent = n()/B)
# output-model
# kable(model, "latex", caption = "Bootstrap for model selection based on mse criteria", booktabs = T)
#   kable_styling(latex_options = c("hold_position", "scale_down")) %>%
#   add_footnote(c("B=10000"),
#                notation = "alphabet")
################ cross validation for comparison of prediction error
cvcomp <- function(dat,K){
  n <- dim(dat$X)[1]
  folds <- sample(1:K, n, replace=TRUE)
  start <- rep(0, dim(dat$X)[2])
  cv.error.1 <- vector()
  cv.error.2 <- vector()
  for (i in 1:K){
    cv.x <- dat$X[folds!=i,]
    cv.y <- dat$y[folds!=i]
    min.lam <- cv.glmnet(x=dat$X[folds!=i,],y=dat$y[folds!=i],alpha=1, family="binomial")$lambda.min
    las <- glmnet(x=dat$X[folds!=i,],y=dat$y[folds!=i], alpha=1,family="binomial",lambda = min.lam)
    # change according to the result of model selection
    min.new <- cv.glmnet(x=dat$X[folds!=i,],y=dat$y[folds!=i],alpha=0, family="binomial")$lambda.min
    new <- glmnet(x=dat$X[folds!=i,],y=dat$y[folds!=i], alpha=0,family="binomial",lambda = min.new)
    y.las <- predict(las,dat$X[folds == i,],type = "response")>0.5
    y.new <- predict(new,dat$X[folds == i,],type = "response")>0.5
    # misclassification rate.
    cv.error.1[i] = sum(y.las!=dat$y[folds == i])/length(y[folds == i])
    cv.error.2[i] = sum(y.new!=dat$y[folds == i])/length(y[folds == i])
  }
  return(cbind(mean(cv.error.1),mean(cv.error.2)))
}
cvresult <- round(cvcomp(dat,K=5),4)
cvresult
# plot for subject one
# subject.1 <- unlist(lapply(out[,2],mean))
# ggplot() + aes(subject.1)+ geom_histogram(colour="black", fill="white")

## inference
# ycount <- out[,c((3+1+dim(dataX)[2]):(3+n+dim(dataX)[2]))]
# meany <- apply(ycount, 2, mean)
# par <- dim(dataX)[2]+1
# coef <- round(apply(out[,c(3:(3+dim(dataX)[2]))],2,mean),4)
# sd <- vector()
# for (k in 1:par) {
#   covj <- t(ycount-(as.vector(meany)%*%t(rep(1,B))))%*%(out[,c(2+k)]-rep(coef[k],B))
#   sd[k] <- round(sqrt((t(covj)%*%covj)/B),4)
# }
# coeff <- cbind(coef,sd,pnorm(coef/sd))
# colnames(coeff) <- c("coefficient","standard.err", "p-val")
# rownames(coeff) <- c("intercept", colnames(dataX))
#
# kable(coeff, "latex", caption = "Smoothed bootstrap estimation for best model", booktabs = T) %>%
#   kable_styling(latex_options = c("hold_position", "scale_down")) %>%
#   add_footnote(c("B=5000"),
#                notation = "alphabet")
```