Welcome to part III of debunking the optimism corrected bootstrap in high dimensions (quite high number of features) in the Christmas holidays. I suggest avoiding this method until at some point it has been reassessed thoroughly to find how bad this situation is with different numbers of dimensions. Yes, I know for some statisticians this is your favorite method and they tell people how their method is lacking in statistical power, but clearly this has much worse issues, at least on some data. People are currently using this method in genomic medicine, where we have high numbers of features low samples. Just re-run the code yourself and make up your own mind if in doubt. I have now 3 implementations (excluding Caret) confirming the bias. One written by me, two independent statisticians. Let’s run some more experiments.

This time I have used a different persons implementation using the ‘glm’ function, i.e. logistic regression to show the same misleading trend occurs, i.e. positive results on purely random data. The second implementation is from another unnamed statistician, there was a bug in their code so I had to correct it, there may be further errors in it so feel free to check.

If you disagree with these findings, please show your own implementation of the method in R (not using Caret) following the same experiment of increasing number of features that are purely noise very high with binary labels. Also, check the last two posts and re-run the code your self, before making your mind up and providing counter arguments. I am practically certain there is a serious problem with this method. Yet again, with situations like this, don’t rely on Mr X’s publication from 10 years ago, use null simulated data sets with different dimensions to investigate the behavior of the method in question your self.

There are no real predictors in this data, all of them are random data from a normal distribution. The bias is less bad using glm than with glmnet. This method is being used (with high numbers of features) as we speak by Mr Fudge and his friends so they can get into better journals. The system is corrupt.

You should be able to copy and paste this code directly into R to repeat the results. I can’t vouch for this code because I did not write it, but both implementations show the same result.

Implementation 1. A glm experiment.

### example of logistic regression optimism corrected bootstrapping

auc.adjust <- function(data, fit, B){

fit.model <- fit

data$pred.prob <- fitted(fit.model)

auc.app <- roc(data[,1], data$pred.prob, data=data)$auc # require 'pROC'

auc.boot <- vector (mode = "numeric", length = B)

auc.orig <- vector (mode = "numeric", length = B)

o <- vector (mode = "numeric", length = B)

for(i in 1:B){

boot.sample <- sample.rows(data, nrow(data), replace=TRUE) # require 'kimisc'

fit.boot <- glm(formula(fit.model), data = boot.sample, family = "binomial")

boot.sample$pred.prob <- fitted(fit.boot)

auc.boot[i] <- roc(boot.sample[,1], boot.sample$pred.prob, data=boot.sample)$auc

data$pred.prob.back <- predict.glm(fit.boot, newdata=data, type="response")

auc.orig[i] <- roc(data[,1], data$pred.prob.back, data=data)$auc

o[i] <- auc.boot[i] - auc.orig[i]

}

auc.adj <- auc.app - (sum(o)/B)

boxplot(auc.boot, auc.orig, names=c("auc.boot", "auc.orig"))

title(main=paste("Optimism-adjusted AUC", "\nn of bootstrap resamples:", B), sub=paste("auc.app (blue line)=", round(auc.app, digits=4),"\nadj.auc (red line)=", round(auc.adj, digits=4)), cex.sub=0.8)

abline(h=auc.app, col="blue", lty=2)

abline(h=auc.adj, col="red", lty=3)

}

## generate random data

xx.glmnet <- matrix(nrow=53,ncol=1)

xx.glmnet <- data.frame(xx.glmnet)

xx.glmnet[,1] <- c(rep(0,25),rep(1,28))

test <- matrix(rnorm(53\*1000, mean = 0, sd = 1),

nrow = 53, ncol = 1000, byrow = TRUE)

xx.glmnet <- cbind(xx.glmnet,test)

colnames(xx.glmnet) <- c('outcome',paste('Y',seq(1,1000),sep=''))

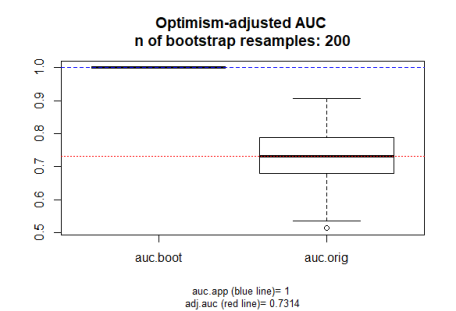
## 1. make overfitted model

model <- glm(outcome ~., data = xx.glmnet, family = "binomial")

## 2. estimate optimism and correct

auc.adjust(xx.glmnet, model, B=200)

Let's have a look at the results, which agree nicely with our previous findings using a from scratch implementation of the method. So the red line is supposedly our corrected AUC, but the AUC should be 0.5 when running on random data.



Implementation 2: Another glmnet experiment

### example of optimism corrected bootstrapping implementation

library(boot)

library(pROC)

library(glmnet)

#glmnet -all predictors penalised

compare\_opt\_glmnet<- function(orig\_data\_glmnet, i){

# orig\_data\_glmnet = the whole data

train\_data\_glmnet<- orig\_data\_glmnet[i, ] # get bootstrap

model\_glmnet<- model\_process\_glmnet(train\_data\_glmnet)

# return a summary optimism results

AUC\_results\_glmnet <- c(

# boot against original

AUC\_train1=roc(orig\_data\_glmnet$outcome,as.vector(predict(model\_glmnet,type="response", newx = model.matrix (outcome~.,orig\_data\_glmnet )[,-1], s = 'lambda.min')))$auc,

# boot against boot

AUC\_train2=roc(train\_data\_glmnet$outcome,as.vector(predict(model\_glmnet,type="response", newx = model.matrix (outcome~.,train\_data\_glmnet )[,-1], s = 'lambda.min')))$auc,

AUC\_diff=roc(train\_data\_glmnet$outcome,as.vector(predict(model\_glmnet,type="response", newx = model.matrix (outcome~.,train\_data\_glmnet )[,-1], s = 'lambda.min')))$auc-

roc(orig\_data\_glmnet$outcome,as.vector(predict(model\_glmnet,type="response", newx = model.matrix (outcome~.,orig\_data\_glmnet )[,-1], s = 'lambda.min')))$auc

)

return(AUC\_results\_glmnet)

}

model\_process\_glmnet<- function(the\_data\_glmnet){

model\_final\_glmnet <- cv.glmnet (model.matrix (outcome~.,the\_data\_glmnet )[,-1],the\_data\_glmnet$outcome,alpha=1,family = "binomial")

return(model\_final\_glmnet)

}

# generate random data

test <- matrix(rnorm(53\*1000, mean = 0, sd = 1),

nrow = 53, ncol = 1000, byrow = TRUE)

test <- data.frame(test)

labs <- factor(c(0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,1,0,1,1,0,1,0,1,0,0,0,0,0,

1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0))

labs <- data.frame(as.matrix(labs))

colnames(labs)[1] <- 'outcome'

xx.glmnet <- cbind(labs,test)

## 1. make overfitted model

tempd <- xx.glmnet[,2:ncol(xx.glmnet)]

labels <- xx.glmnet[,1]

overfittedmodel <- cv.glmnet(as.matrix(tempd),y=labels,alpha=1,family="binomial")

lasso.prob <- predict(overfittedmodel,type="response",newx=as.matrix(tempd),s='lambda.min')

pred.cligen <- prediction(lasso.prob, labels)

auc <- roc(labels,as.vector(lasso.prob))

overfitted\_auc <- as.numeric(auc$auc)

## 2. try to correct for optimism

Repeats = 100

res\_opt\_glmnet <- boot(xx.glmnet, statistic = compare\_opt\_glmnet, R = Repeats)

optimism\_glmnet <- mean(res\_opt\_glmnet$t[ , 3])

enhanced\_glmnet <- overfitted\_auc - optimism\_glmnet

Here are the results of the above code on random data only (pure madness).

> enhanced\_glmnet

[1] 0.8916488

If in doubt compare your results with LOOCV and k fold cross validation, this is how I discovered this method is dodgy by seeing MASSIVE differences on null data-sets where no real predictive ability should be found. These two methods are more widely used and reliable.

Sorry @ Frank Harrell, I don’t have time atm to do the extra bits you suggest.  
Game on @ Davis Vaughan. Do your own implementation with the same data as I have used with glmnet and glm.

The steps of this method, briefly, are as follows:

1. Fit a model M to entire data S and estimate predictive ability C.
2. Iterate from b=1…B:
   1. Take a resample from the original data, S\*
   2. Fit the bootstrap model M\* to S\* and get predictive ability, C\_boot
   3. Use the bootstrap model M\* to get predictive ability on S, C\_orig
3. Optimism O is calculated as mean(C\_boot – C\_orig)
4. Calculate optimism corrected performance as C-O.