STAT 201B Final Project: The .632 Bootstrap Estimator

Ryan Anderson
raanderson@g.ucla.edu

March 18, 2023

Abstract

The .632 bootstrap is a nonparametrically improved estimator for prediction error. It is a shrinkage estimator formed by a convex combination of two more basic estimators for prediction error: the apparent error estimator and the bootstrap-smoothed leave-one-out cross-validation error. We provide theoretical considerations as to the utility of this estimator and provide simulated results as a demonstration of its efficacy.

1 Introduction

The .632 bootstrap is a nonparametrically improved estimator for prediction error. It is a shrinkage estimator formed by a convex combination of two more basic estimators for prediction error: the apparent error estimator and the bootstrap-smoothed leave-one-out cross-validation error.

The need for such estimators arises naturally in classification and regression problems. A typical setup is as follows: given (X, y) covariates and outcomes data, we form a prediction rule $r(x_i)$. With a loss function $L(y_i, r(x_i))$, we consider the apparent error, $\bar{err} = \frac{1}{N} \sum L(y_i, r(x_i))$. This could also be simply described as the training error, where the training error is taken to be our entire dataset.

Now let F_i be the empirical distribution of the X_i with row i deleted, and consider bootstrap estimates x^* taken from this distribution. Then the leave one out bootstrap estimator is given by $\hat{Err}^1 = \frac{1}{N} \sum E_{F_i} L(y_i, r(x^*))$.

With these two in mind, the .632 estimator can be stated simply as $\hat{Err}^{.632} = 0.368 * \bar{err} + 0.632 * \hat{Err}^1$.

This basic reason for the improvement gained here is twofold: the first is that the bootstrap-smoothed leave-one-out cross-validation estimator can smooth the discontinuities of simple cross-validation, and thus lead to lower variance than the cross-validation estimator. The second, however, is that this variance-reduction smoothing leads to an upward bias in the estimator.

More details, including theoretical justifications for the sources of bias, are provided in the notes below.

2 Literature Review

Fleshing out the theory of estimating prediction error was a major research project of scholars like Efron, Tibshirani, Gong, Breiman, Mallows, and Wahba beginning in the mid-1970s. The essential problem of using the training error as an estimate for the prediction error was obvious early on – because the same data is being used to train and test the prediction rule, the apparent error incurs the covariance penalty, and thus underestimates the prediction error: $E[Err_i] = E[err_i] + 2 * cov(\hat{\mu}_i, y_i)$.

Mallows (1973) introduced the C_p estimate which recognized and corrected for this – later improvements would include Stein's unbiased risk estimator and formulas involving Akaike's information criterion, albeit parametrically.

By contrast, Efron (1983) compares different nonparametric improvements upon the apparent error. This would be followed up by Efron and Tibshirani (1997) where a fuller theory of comparison among estimators is developed. Somewhat humorously, Efron notes in the 1983 paper that the although the .632 estimator performs best among the estimators surveyed, the theoretical justification for it is weak, "leaving open the possibility that the estimator's success...was a fluke". The theoretical justification provided in that paper, despite the author's low opinion, has some attractiveness, and so will be resummarized here.

3 Theoretical Notes

Parametrically, we saw above that under the assumed distribution of the outcome variable $y \sim F(\mu, \sigma I)$, with $\hat{\mu} = (X^T X)^{-1} X^T Y$, the apparent error understates the prediction error by $2 * cov(\hat{\mu}_i, y_i)$. A non-parametric explanation can instead be given by thinking about the space of covariate vectors as a sort of metric space.

Compare the training dataset X (in the case of the apparent error taken to be the full X with no cross-validation) to a new data point x_0. We assume that prediction error increases as the distance between x_0 and the closest point in X to x_0 grows. Therefore, for any data point not contained inside the training set, the apparent error will underestimate the true error.

Nonparametric improvements need to "spread out" the training data over the covariate space in order to get rid of the downward bias. Leave-one-out cross-validation does this correctly, but at a cost of high variance. Adding a bootstrap, however, smooths out the

variance while retaining the benefit of small bias. This bootstrapped-smoothed version of the CV estimator turns out to in fact have an upward bias. The source of this upward bias lies in the fact that at each bootstrapped CV step we are using only a proportion of the sample.

In order for the apparent error calculation to be unbiased, we need for the distance between the test data point x_0 and the training dataset X to be zero - the probability of that is $P(\delta(x_0, X) = 0)$. Given a empirical distribution produced by the bootstrap, we can calculate this probability as the chance of not being included in any of the bootstrap samples: $P(d(x_0^*, X^*) = 0) = 1 - (1 - \frac{1}{n})^n \approx 1 - \frac{1}{e} = 0.632$ for n large enough.

If we assume that the true prediction error Err is the same under no cross validation (err) and under bootstrap-smoothed cross-validation (Err^1), and furthermore that the true error is a linear function of distance from a test point to the training dataset, $Err = \nu + \beta\Delta$, then we end up with a relation that $Err = 0.318 * err + 0.632 * Err^1$, which we estimate as $\hat{Err}^{0.632} = 0.318 * \bar{err} + 0.632 * \hat{Err}^1$.

4 Results

We use for our simulated example the *spam* data from Efron and Hastie (n.d.), which consists of 56 covariates on 4,600 emails and an outcome variable labeled SPAM, which labels an email as either spam or genuine. We use regression trees via the R package *rpart* to classify emails as spam or not.

We compare the performance of two estimators of prediction error for the classification problem: one is our bootstrapped-smooth estimator and the other is the naive LOOCV estimate, which performs leave-one-out cross-validation to predict the label of the out of sample. Efron (1983) also compared both and found the 632 estimator to always outperform.

Our loss function is 0-1 loss. We perform 100, 200, 300, 400, 500, and 1000 bootstrap iterations for the 632 estimator. Our data is subsetted on each run to take 20 rows at random with replacement from the *spam* data – this is similar to the scale of the data used by Efron (1983). It is notable that the 632 estimator is so computationally complex that retaining the full data set was infeasible on a modern Macbook.

The results are displayed in the figure and table below – the 632 estimator outperforms the naive LOOCV estimator in terms of MSE at all iterations, and, as expected, improves as the bootstrap depth increases.

References

Efron, B. (1983, June). Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation. *Journal of the American Statistical Association*, *78*(382), 316–331. Retrieved 2023-03-18, from http://www.tandfonline.com/doi/abs/10.1080/01621459.1983.10477973 doi: 10.1080/01621459.1983.10477973

Efron, B., & Gong, G. (1983, February). A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation. *The American Statistician*, *37*(1), 36. Retrieved 2023-03-18, from https://www.jstor.org/stable/2685844?origin=crossref doi: 10.2307/2685844

Efron, B., & Hastie, T. (n.d.). *Computer Age Statistical Inference*.

Efron, B., & Tibshirani, R. (1997, June). Improvements on Cross-Validation: The 632+ Bootstrap Method. *Journal of the American Statistical Association*, *92*(438), 548–560. Retrieved 2023-03-15, from https://doi.org/10.1080/01621459.1997.10474007 (Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/01621459.1997.10474007) doi: 10.1080/01621459.1997.10474007

Mallows, C. L. (1973, November). Some Comments on C P. *Technometrics*, *15*(4), 661. Retrieved 2023-03-18, from https://www.jstor.org/stable/1267380?origin=crossref doi: 10.2307/1267380

Table 1: Results from simulation of spam data

	Bootstrap Iterations	632 Ests	CV Ests
1	100.00	0.46	0.55
2	200.00	0.24	0.50
3	300.00	0.16	0.35
4	400.00	0.26	0.55
5	500.00	0.27	0.45
6	1000.00	0.22	0.50

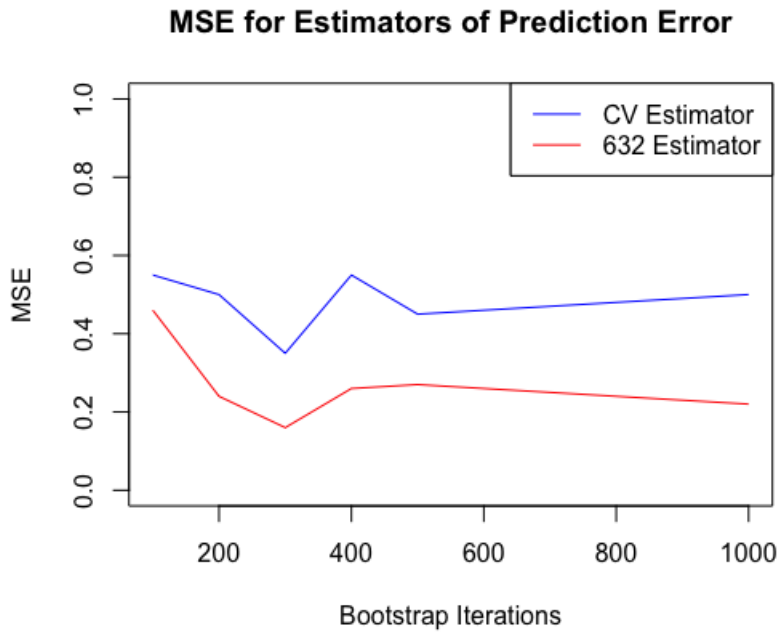

Figure 1: Results from comparison simulation of spam data

```
    library(MASS)
library(lattice)
library(caret)
library(boot)
library(rpart)
library(xtable)

spam_data <- read.csv("/Users/ryananderson/Documents/STAT 201B HW/Final Project
length(spam_data[1,])
B <- 1000
trunc_spam_data <- spam_data[sample(1:length(spam_data$spam),25,replace=FALSE),
```

```
covars_spam_data <- trunc_spam_data %>% subset(select=-spam)
a <- 1:length(trunc_spam_data$spam)
boot_errors <- rep(0,length(trunc_spam_data$spam))
in_boot_count <- rep(0,length(trunc_spam_data$spam))
for(i in 1:length(trunc_spam_data$spam)){
  for(j in 1:B){
    # Pull bootstrap indices and not in bootstrap indices
    boot_indices <- sample(a,replace=TRUE)
    if(!i %in% boot_indices){ next }
    # Form new train data matrix from bootstrap indices
    new_mat <- trunc_spam_data[boot_indices,]
    # Generate glm
    boot_glm <- rpart(spam ~ ., data=new_mat)
    # Find ith error in this boot run and add to total error for this index
    boot_err <- ifelse(trunc_spam_data$spam[i] == predict(boot_glm,covars_spam_
    boot_errors[i] <- boot_errors[i] + boot_err
    in_boot_count[i] <- in_boot_count[i] + 1
  }
}
loobs_err <- sum(boot_errors)/sum(in_boot_count)

apparent_model <- rpart(spam ~ ., data=trunc_spam_data)
apparent_error <- predict(apparent_model,covars_spam_data)
apparent_error_walled <- ifelse(apparent_error <= 0.5,0,1)
apparent_error_total <- (1 / length(apparent_error)) * (sum(ifelse(trunc_spam_c

est_632_err <- 0.632 * loobs_err + 0.328 * apparent_error_total

LOO_errors <- c()
for(i in 1:length(trunc_spam_data$spam)){
  LOO_covars <- trunc_spam_data[-i,]
  LOO_y <- trunc_spam_data$spam[i]
  LOO_model <- rpart(spam~.,data=LOO_covars)
  LOO_predict <- predict(LOO_model,trunc_spam_data[i,])
```

```
  LOO_predict_barrier <- ifelse(LOO_predict < 0.5,0,1)
  LOO_errors[i] <- ifelse(LOO_y == LOO_predict_barrier,0,1)
}
cv_est_err <- (1 / length(trunc_spam_data$spam)) * sum(LOO_errors) + apparent_e
```