Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

NON-PARAMETRIC UNCERTAINTY BIAS AND VARIANCE ESTIMATION VIA NESTED BOOTSTRAPPING AND INFLUENCE FUNCTIONS

Kimia Vahdat
Sara Shashaani
Edward P. Fitts Department of Industrial and Systems Engineering
North Carolina State University
915 Partners Way,
Raleigh, NC 27607, USA

ABSTRACT

In using limited datasets, modeling the uncertainty via non-parametric methods arguably provides more robust estimators of the unknown value of interest. We propose a novel nested bootstrap method that accounts for the uncertainty from various sources (input data, model, and estimation) more robustly. The nested bootstrap is particularly apt to the more nuanced conditional settings in constructing prediction rules but is easily generalizable. We utilize influence functions to estimate the bias due to input uncertainty and devise a procedure to correct the estimators' bias in a simulation optimization routine. Implementations in the context of feature selection via simulation optimization on two simulated datasets prove a significant improvement in robustness and accuracy.

1 INTRODUCTION

In a data-driven simulation, there are three primary sources of uncertainty: stochastic error (intrinsic error), input model risk (extrinsic error), and learning model discrepancy error (Song and Nelson 2019). The finiteness of simulation runs and data points cause the first two error types, respectively. Intrinsic error is well studied in the literature; for an extensive survey on quantification of extrinsic errors, see (Lam 2016). The third error type refers to the logic model's misrepresentations of the underlying system, which is often neglected. While simulation runs can be increased in principle to reduce the intrinsic error, reducing the extrinsic error by collecting more data points may not be possible in many practical problems. Nevertheless, quantifying the effect and contribution of each error type in the performance estimation is worthwhile. When estimating an estimator's confidence intervals (CI), especially in optimization, not considering all sources of the errors causes under-coverage of the CI and misleads the search. In particular, the bias induced by these errors can shift the estimators' location substantially. This paper aims to introduce an estimator that considers all sources of uncertainty and attempts debiasing estimates of the desired objective.

Despite a rise in the literature of input model risk quantification over the recent years (Barton et al. 2018; Corlu and Biller 2015), fewer studies concentrate on non-parametric input distributions (Lam and Zhou 2017; Barton et al. 2018; Vahdat and Shashaani 2020) and detecting input model induced bias (Morgan et al. 2019). In this paper, we characterize the bias due to the extrinsic error in non-parametric settings that do not limit the analysis to any family of distributions. With a novel formulation, we specifically emphasize the extrinsic error bias in data-driven prediction rules through the lens of simulation optimization (SO). Evaluating prediction rules is particularly challenging as it requires careful sampling schemes to reduce the optimism bias, which occurs when the same data used for building a model is also used for evaluating it. A frequent assumption is that estimators in a simulation problem are unbiased to use the law of large numbers. However, with low availability of data, estimators can have a high bias. Our two main contributions in this paper seek to improve prediction rules' accuracy and robustness to make them more reliable. First, we review the existing bootstrapping and cross-validation sampling schemes, and propose a nested bootstrapping method that results in a prediction error with low variance and small bias. Second, employing von-Mises expansion (Fernholz 2012) for the true and estimated distributions, we introduce an approach to debias the error estimates.

1.1 Notation

In this paper, small letters and capital letters denote deterministic and stochastic values, respectively. Bold letters represent vectors and script letters are used for a collection of vectors. We represent a given data point by z that consists of d covariates in a $\mathcal{X}^d$ space and a (real) response variable, hence $z \in \mathcal{X}^d \times \mathbb{R}$. Our decision variable is $\boldsymbol{x} \in \mathcal{S}^d$ in the space of interest, $\mathcal{S}$, and we define $q(z_0, \boldsymbol{x}, \boldsymbol{z})$ as the ℓ_2 norm of the loss that is the difference between the true response of point z_0 and its predicted response using a model trained by a set of data $\boldsymbol{z}$ and using parameters $\boldsymbol{x}$ (in this paper we use linear regression and the parameters are the features included in the training, i.e., $\mathcal{S} = \{0,1\}$). We let $\mathcal{M}^* := \left(\boldsymbol{M}^1, \boldsymbol{M}^2, \cdots, \boldsymbol{M}^{b'}\right)^\intercal$ be a collection of random vectors, each $\boldsymbol{M}^b \in \mathbb{Z}_+^n$ recording the number of draws of point i for a bootstrap following the Mult$(n, \boldsymbol{p}^a)$ (multinomial) distribution, with $\boldsymbol{p}^a = (\frac{1}{n}, \cdots, \frac{1}{n}) \in \mathbb{R}^n$. We denote the cdf of $\boldsymbol{p}^a$ with F^a, and the correct and unknown cdf of the dataset on hand with F^c. Lastly, we denote $\mathcal{M}^{**} := \left([\boldsymbol{M}^{1,1}, \boldsymbol{M}^{1,2}, \cdots, \boldsymbol{M}^{1,r'}], \cdots, [\boldsymbol{M}^{b',1}, \boldsymbol{M}^{b',2}, \cdots, \boldsymbol{M}^{b',r'}]\right)^\intercal$ as a collection of random vectors, with each $\boldsymbol{M}^{b,r} \in \mathbb{Z}_+^n$ recording the number of draws of point i for bootstrap r that follows the Mult$(n, \boldsymbol{P}^b)$ distribution, with $\boldsymbol{P}^b = (\frac{M_1^b}{n}, \cdots, \frac{M_n^b}{n}) \in \mathbb{R}^n$, and F^b its cdf.

1.2 Problem Statement: Robustly Calibrating Prediction Rules

Let us consider the loss function of a predictive rule with a fixed learning structure (e.g., linear regression) for a given set of hyper-parameters $\boldsymbol{x}$. Following the notation introduced, the true objective function is

$$\min_{\boldsymbol{x}} \theta(\boldsymbol{x}|F^c) = \mathbb{E}_{\boldsymbol{Z}}\left[\mathbb{E}_{Z_0}\left[q(Z_0, \boldsymbol{x}, \boldsymbol{Z}) | \boldsymbol{Z} \sim F^c\right]\right], \tag{1}$$

where the inner expectation is with respect to the unseen data with both train and test sets following the true distribution F^c. Note, the outer expectation is over training *sets* and the inner expectation is over test *points*. This is one complication of such problems. Additionally, without access to F^c, the desired objective function (1) is adjusted by employing the empirical distribution of the data,

$$\min_{\boldsymbol{x}} \theta(\boldsymbol{x}|F) = \mathbb{E}_{\boldsymbol{Z}}[\mathbb{E}_{Z_0}\left[q(Z_0, \boldsymbol{x}, \boldsymbol{Z}) | \boldsymbol{Z} \sim F^{(0)}\right]]. \tag{2}$$

In the above $\boldsymbol{Z} \sim F^{(0)}$ implies that each point in the training set $\boldsymbol{Z}$ is drawn from the empirical cdf that excludes the point Z_0 to avoid overfitting; see expression (6) in (Efron and Tibshirani 1997). This is the second complication of calibrating prediction rules. Typically the empirical cdf of the whole data, i.e., $F = F^a$ and $F^{(0)} = F^{a,(0)}$ is used to compute (2). We extend this formulation by taking an expectation over all possible F from the available data and defining

$$\min_{\boldsymbol{x}} \theta(\boldsymbol{x}) := \mathbb{E}_F[\theta(\boldsymbol{x}|F)], \tag{3}$$

where the outcome is defined by integrating over empirical distribution itself. We can view the integration as the average loss under slight differences in the dataset available to us. The result is three nested expectations (from innermost to outer: on a single point, a set of points, and a distribution) instead of two, adding another layer of complication that resembles the nested simulation framework by Sun, Apley, and Staum (2011). We expect this new objective to be more resilient to the changes in the data leading to optimal choices for the prediction rule that are more robust. In Section 2 we survey existing estimations for each objective and propose a new nested bootstrap scheme that exhibits more promise in comparison.

1.3 Why do we care about robustness?

Having a robust design is to maintain a low bias and variance in the solutions and designs. Sanchez and Sanchez (2020) discuss different designs and highlight the merits of robust designs. In real-world problems, we may not always keep both bias and variance low, and often settle for a compromise between the two. One can evaluate robustness from two standpoints in an SO problem. The first is the estimator's stability, also referred to as robust statistics. The second is the robustness of the SO algorithm's solutions. In this paper, we focus on both evaluations as we compare different estimators.

One core component of robust statistics is the notion of influence functions (IF). IF's or nonparametric delta methods have recently gained a lot of attention in the machine learning communities, because of their nonparametric settings and ease of inference (Fisher and Kennedy 2020). Since IF based estimators are consistent with respect to small changes in the data or the model, they are considered doubly robust estimators. Using IF's provides a way to debias any estimator that is smooth with respect to changes in the data which we will describe in details in Section 3. In a numerical experimentation in Section 4 we test the effectiveness of our solution method focusing on estimation (the solver is a Genetic Algorithm that is suitable for our binary space search). We implement the methodology for feature selection problems on high-dimensional datasets that we simulate knowing the true contributing features.

2 SURVEY OF PREDICTION RULES' PERFORMANCE ESTIMATION

Evaluating a prediction rule on the same dataset that it is trained on will result in an "optimistic" estimate of the model performance, which is often called the apparent error or in-sample error. In-sample error or $\overline{err}(\boldsymbol{x}) := \hat{\mathbb{E}}_{\boldsymbol{Z}}[\hat{\mathbb{E}}_{Z_0}[q(Z_0, \boldsymbol{x}, \boldsymbol{Z})|\boldsymbol{Z} \sim F]]$ is not a good performance estimator for predictive models in machine learning and can be highly biased. This bias, $\theta(\boldsymbol{x}|F^c) - \overline{err}(\boldsymbol{x})$, is called the overfit or the optimism bias. The overfit bias is due to the fact that the data points on which we wish to make predictions are not accessible in the modeling phase. There are many sampling solutions to this problem in the literature that attempt to split the data in a way to get an almost unbiased estimate without sacrificing the robustness. Some of the well known sampling methods are $k-$fold cross validation, .632 bootstrapping, and leave-one-out bootstrap (Efron and Tibshirani 1997; Efron 1983), which we briefly describe in this section. $k-$fold cross validation (CVk) is one of the resampling methods that is widely used in practice. In CVk, the data is divided into k' non-overlapping folds. Setting each fold aside, a model is trained on the remaining $k-1$ folds, and then evaluated on the excluded fold. Hence, only one prediction is made for each observation. Its estimator can be achieved by $\hat{\theta}_{\text{CVk}}(\boldsymbol{x}) = \frac{1}{k}\sum_{j=1}^{k}\frac{1}{n/k}\sum_{i=1}^{n/k} q(z_{j_i}, \boldsymbol{x}, \boldsymbol{Z}^{(j)})$ where y_{j_i} refers to point i in fold j, and $\boldsymbol{Z}^{(j)}$ denotes all data excluding the j-th fold. Leave-one-out-CV (LOOCV) estimator, $\hat{\theta}_{\text{LOOCV}}(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} q(z_j, \boldsymbol{x}, \boldsymbol{Z}^{(j)})$, is a special case of CVk where $k = n$, with less bias since each training set contain $n-1$ points. Bengio and Grandvalet (2005) show that $\text{Var}(\hat{\theta}_{\text{CVk}}(\boldsymbol{x})) = \sigma^2(\boldsymbol{x})/n + \omega(\boldsymbol{x})\frac{n/k-1}{n} + \gamma(\boldsymbol{x})\frac{n-n/k}{n}$, where ω and γ are the covariances between two predicted values in the same fold and different folds, respectively, and $\sigma^2(\boldsymbol{x})$ is the variance of each observation. They also prove that there is no unbiased estimator for this quantity. In their experimental analysis, they show that as k increases, all variance terms decrease, therefore the total variance also decreases. Since the training sets in LOOCV are very similar to each other, with $n-2$ mutual observations out of $n-1$ points in each fold, the variance of the estimator will be close to $\sigma^2/n + \gamma$. To further reduce the variability of CVk, repeated CV was introduced which simply repeats the whole procedure multiple times and reports the overall average.

Bootstrapping (Efron 1979) is another resampling method that can be used for the evaluation of prediction rules. Bootstrapping was introduced as a way to evaluate estimators' variance and obtain narrower confidence intervals. In each of the $b = 1, \cdots, b'$ bootstraps, a model is built on n samples with replacement from the training data, $\boldsymbol{Z}^b$, and evaluated on the remaining points. Letting $I_i^{(b)} = \mathbb{I}(z_i \notin \boldsymbol{Z}^b)$ represent the points not in the training data and $J_i = \sum_{b=1}^{b'} I_i^{(b)}$ represent the number of unique predictions for point i, which equals the number of bootstraps that exclude point i, we summarize the calculation as

$$\hat{\theta}_{\text{Boot}}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\max\{1, J_i\}}\sum_{b=1}^{b'} I_i^{(b)} q(z_i, \boldsymbol{x}, \boldsymbol{Z}^b), \tag{4}$$

where $I_i^{(b)}$ discounts the function value for points inside the training data. The maximum operator is to avoid dividing by 0 if a point does not have any predictions at all. In (4) the two averages are estimators for the first and second expectation in (2), where the inner and outer averages estimate the outer and inner expectations, respectively. In the bootstrapping method, since we sample the training sets independently from each other, the correlation between their predictions is much smaller than one and therefore they provide a small variance estimator. In a more in-depth analysis and numerical experiments for variations of bootstrapping and cross validation, Efron (1983) shows that CVk and bootstrap both converge with the rate $\mathcal{O}(1/n^2)$. However, the bootstrap has additional terms in its estimator causing a downward bias, which can be partially overcome with Leave One Out Bootstrap (LOOBoot) or jackknife estimator. LOOBoot estimator is computed as $\hat{\theta}_{\text{LOOBoot}}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{b'}\sum_{b=1}^{b'} q(z_i, \boldsymbol{x}, \boldsymbol{Z}^b_{(i)})$, where $\boldsymbol{Z}^b_{(i)}$ is the b-th bootstrap from the training set that excludes the $i-$th data point. This estimation is more costly than $\hat{\theta}_{\text{Boot}}(\boldsymbol{x})$ as each point will have b' unique predictions. The downward bias in bootstrapping methods motivated the 632Boot estimator that measures a weighted average between the in-sample and the LOOBoot error with weights computed using asymptotic analysis. We write the 632Boot estimator as, $\hat{\theta}_{\text{632Boot}}(\boldsymbol{x}) = 0.368 \times \overline{err}(\boldsymbol{x}) + 0.632 \times \hat{\theta}_{\text{LOOBoot}}(\boldsymbol{x})$.

To further reduce the downward bias of bootstrapping methods, Leave-k-Out-Bootstrap (LkOBoot) (Efron and Tibshirani 1997), a smoothed version of CVk, divides the data into k folds. Then in each of k replications one fold is set aside and from the remaining folds b' bootstrap samples are drawn. Next a model is built on the bootstrap samples and their average performance on the left out fold is reported. The LkOBoot estimator is computed by $\hat{\theta}_{\text{LkOBoot}}(\boldsymbol{x}) = \frac{1}{nb'}\sum_{i=1}^{n}\sum_{j=1}^{k}\sum_{b=1}^{b'} I_i^{j,b} q(z_i, \boldsymbol{x}, \boldsymbol{Z}^{(j),b})$, where $I_i^{j,b} = \mathbb{I}\{z_i \in \boldsymbol{Z}^{j,b}\}$, and $\boldsymbol{Z}^{(j),b}$ denotes b-th bootstrap from the data that excludes the j-th fold. Note, each point still receives exactly b' predictions but the number of models reduces from $n \times b'$ to $k \times b'$.

Vahdat and Shashaani (2020) introduced a sampling method which combines the bootstrapping and cross validation characteristics. We refer to their method as LBootOBoot, short for leave bootstrap out bootstrap. LBootOBoot first takes b' bootstrap resamples to form the training sets, $\boldsymbol{Z}^b$. Then from the $\boldsymbol{Z}^{(b)}$, or those data points that do not appear in $\boldsymbol{Z}^b$, r' test sets are bootstrapped denoted by $\boldsymbol{Z}^{(b),r}$, $r = 1, 2, \cdots, r'$. The novelty of this estimator is in having multiple test and train resamples simultaneously, to reduce the variability. This is the first estimator that uses train and test pairs that do not add up to the whole data and therefore there is a nonzero probability that not all points get predicted. We summarize this estimator as,

$$\hat{\theta}_{\text{LBootOBoot}}(\boldsymbol{x}) = \frac{1}{\sum_{i=1}^{n} I_i}\sum_{i=1}^{n}\frac{1}{\max\{1, J_i\}}\sum_{b=1}^{b'}\frac{\sum_{r=1}^{r'} I_i^{(b),r} q(z_i, \boldsymbol{x}, \boldsymbol{Z}^b)}{\max\{1, \sum_{r=1}^{r'} I_i^{(b),r}\}}, \tag{5}$$

where $I_i^{(b),r} = \mathbb{I}\{z_i \in \boldsymbol{Z}^{(b),r}\}$, $J_i = \sum_{b=1}^{b'} \mathbb{I}(\sum_{r=1}^{r'} I_i^{(b),r} > 0)$ is the number of bootstraps with prediction for point i, and $I_i = \mathbb{I}(J_i > 0)$ specifies whether there is any prediction for point i. Observe that it is possible that $\sum_{i=1}^{n} I_i \neq n$ and plausible that $J_i \neq b'$ for any i and $\sum_{r=1}^{r'} I_i^{(b),r} \neq r'$ for any i and b. Otherwise (5) would simplify to $\hat{\theta}_{\text{LBootOBoot}}(\boldsymbol{x}) = (nb'r')^{-1}\sum_{i=1}^{n}\sum_{b=1}^{b'}\sum_{r=1}^{r'} I_i^{(b),r} q(z_i, \boldsymbol{x}, \boldsymbol{Z}^b)$.

2.1 Comparing Estimators

We compare the above estimators in terms of number of predictions per observation, number of models built, average squared bias, and variance. The number of models shows how time efficient the estimator is, as modeling takes up more time than predicting a data point. As the number of predictions per observation increases, the estimator becomes more stable. The bias and variance are evaluated over a simulated data with known true distribution and underlying relationship between the response and the predictors. Let $y = g(\boldsymbol{x}) + err$ be the true generating formula. Then the bias is defined as $\hat{g}(\boldsymbol{x}) - g(\boldsymbol{x})$ and the variance is the var$(\hat{g}(\boldsymbol{x}))$. Table 1 and Figure 1(a) summarize the comparisons with our proposed estimator, NestBoot, which we describe in the next subsection.

Table 1: We compare the estimators with bias and variance estimated over 100 macro-replications using common random numbers on a simulated dataset. The mean ± standard deviation of the squared bias and variance represent the spread of each estimator's reported measurement within 100 experiments; it is not the confidence interval. The repeated CV has 10 folds and 5 repeats, in the Boot method $b' = 50$ while $b' = 10$ in the other methods to keep roughly the same computation time across estimators.

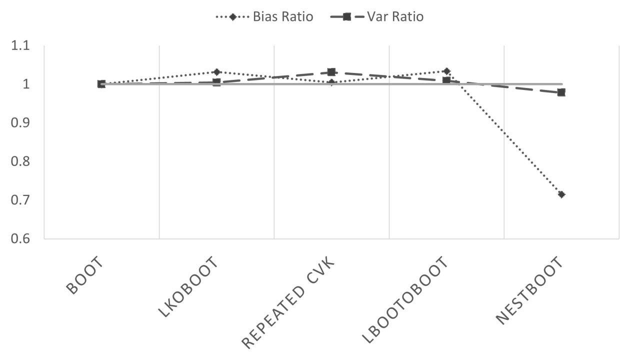
Estimator	**First Samples ...**	**Avg # Predictions**	**# Models**	**Avg Bias^2**	**Avg Variance**
Boot	training set	$\mathbb{E}[\sum_{i,b} I_i^b] \approx 0.37 \times b'n$	b'	852 ± 526	3,691 ± 1,009
LkOBoot	training set	$b'n$	$b'k$	879 ± 518	3,708 ± 1,000
repeated CVk	training set	$rep \times n$	$rep \times k$	856 ± 535	3,805 ± 1,082
LBootOBoot	training set	$\mathbb{E}[\sum_{i,b,r} I_i^{(b),r}]$	b'	881 ± 516	3,724 ± 1,015
NestBoot	test set	$\mathbb{E}[\sum_{i,b,r} M_i^{b,r}] = nb'r'$	$b'r'$	609 ± 408	3,611 ± 970

The Boot estimator requires a larger b' to perform well because its number of models equals the number of bootstraps. All estimators except for Boot and LBootOBoot have more models than bootstraps. The number of models for each estimator is always deterministic, but the number of predictions is stochastic for some of them. Boot, LBootOBoot, and NestBoot have a random number of predictions for each data point. We could not compute a closed-form expression for the average number of predictions for LBootOBoot but expect that to be less than $nb'r'$. In cross validation, we only have one prediction for each data point leading to high variance. On the other hand, bootstrap produces on average $(1 - 0.632) \times b'$ predictions for each data point, which results in more stability, as seen in the standard deviation of the squared bias and variance. We emphasize that Table 1 exhibits the distribution of the measurements that signals into how stable they can be. One can compute the half-widths with the given information. Doing so, e.g., for the squared biases, confirms that there is a statistically significant difference between NestBoot and the existing estimators. The common scheme in all the existing estimators is that a training set is selected first and then its model is evaluated on a test set. Our proposed estimator (NestBoot) reverses this process, successfully improving the stability and robustness while reducing the bias. As illustrated in Figure 1(b), there is less overlap between the training sets that are fewer data points drawn from different data bootstraps.

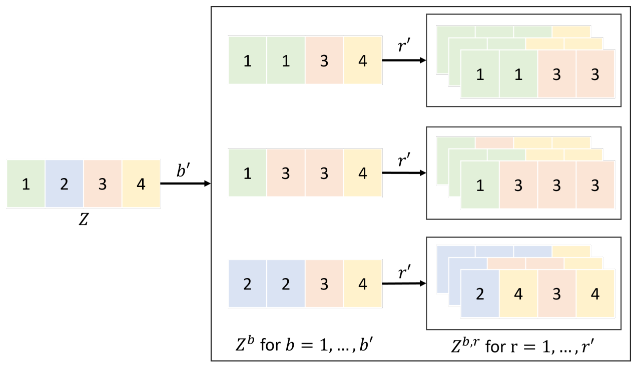
2.2 Proposed Sampling Scheme: Nested Bootstrap Method

Our proposed estimator uses two nested levels of bootstrapping. It first samples modeling sets with replacements and then samples pairs of test and training sets. Bootstraps provide estimates of the empirical distribution with F^b for $b = 1, \cdots, b'$ via sampling with replacement from F^a. In another view, bootstrapping can be considered as a simulation that generates scenarios from the empirical distribution and evaluate the desired functional over each scenario. We explain the details of the proposed sampling method in this section and in the following section we show ways to estimate its bias and variance. Define $M_i^{b,r}$ as the number of times data point i appears in bootstrap r that is resampled from bootstrap b, on which it appears M_i^b number of times. Therefore,

$$\hat{\theta}_{\text{NestBoot}}(\boldsymbol{x}) = \frac{1}{\sum_i^n I_i} \sum_i^n \frac{1}{\max\{1, J_i\}} \sum_b^{b'} \frac{\sum_r^{r'} M_i^{b,r} q(z_i, \boldsymbol{x}, \boldsymbol{Z}^{b,(r)})}{\max\{1, \sum_{r=1}^{r'} M_i^{b,r}\}}, \tag{6}$$



(a) bias and variance ratio in comparison with other methods.



(b) an example of NestBoot with $b' = r' = 3$ and $n = 4$.

Figure 1: NestBoot sampling scheme is the new proposed estimation method.

where $J_i = \sum_{b=1}^{b'} \mathbb{I}(\sum_{r=1}^{r'} M_i^{b,r} > 0)$ represents the number of first-level bootstraps that contain data point i in at least one of their nested bootstraps, $I_i = \mathbb{I}(J_i > 0)$ specifying whether there is any prediction for point i, $\boldsymbol{Z}^b$ is the b-th bootstrapped dataset, and $\boldsymbol{Z}^{b,r}$ is the r-th dataset bootstrapped from $\boldsymbol{Z}^b$. Specifically, $\boldsymbol{Z}^{b,r}$ is the test set and $\boldsymbol{Z}^{b,(r)}$ is the training set containing all data points in $\boldsymbol{Z}^b$ that were not at all selected for $\boldsymbol{Z}^{b,r}$. Each of the three averages in (6) estimates a corresponding expectation in (3). Figure 1(b) illustrates the NestBoot process, also summarized below:

- For $b = 1, \cdots, b'$:
  - Sample $\boldsymbol{M}^b$ randomly from F^a and create $\boldsymbol{Z}^b$.
  - For $r = 1, \cdots, r'$:
    - ∗ Sample $\boldsymbol{M}^{b,r}|\boldsymbol{M}^b$ randomly from F^b, and create $\boldsymbol{Z}^{b,r}$ and $\boldsymbol{Z}^{b,(r)}$.
    - ∗ Build model on $\boldsymbol{Z}^{b,(r)}$ and evaluate on $\boldsymbol{Z}^{b,r}$.

In the next two theorems we show that first sampling the test sets instead of the training sets increases the chance of having at least one prediction for each point.

Theorem 1 The NestBoot estimator makes the likelihood of a point being evaluated in the test set larger than its likelihood of being used in a training set to build a learner.

Proof. Given a data point z_i and an arbitrary b and r for the two-level-bootstraps,

$$\begin{aligned}\Pr\{z_i \in \boldsymbol{Z}^{b,r}\} &= \Pr\{M_i^{b,r} > 0\} \\ &= \sum_{m=1}^{n} \Pr\{M_i^{b,r} > 0 | M_i^b = m\} \Pr\{M_i^b = m\} = \sum_{m=1}^{n} (1 - (1 - m/n)^n) \Pr\{M_i^b = m\} \\ &= \mathbb{E}[1 - (1 - M_i^b/n)^n] \xrightarrow[n\to\infty]{} 1 - e^{e^{-1}-1} \approx 0.47, \end{aligned} \tag{7}$$

where we have used $\lim_{n\to\infty}(1 + k/n)^n = e^k$. However, following the same steps for $\Pr\{z_i \in \boldsymbol{Z}^{b,(r)}\} = \sum_{m=1}^{n} \Pr\{M_i^{b,r} = 0 | M_i^b = m\} \Pr\{M_i^b = m\}$, we observe that the last line of (7)'s counterpart becomes

$$\mathbb{E}[(1 - M_i^b/n)^n] - (1 - 1/n)^n \xrightarrow[n\to\infty]{} e^{e^{-1}-1} - e^{-1} \approx 0.16.$$

□

Next we show that by letting $b'r' \to \infty$, the likelihood of having at least one prediction for each point converges to 1. The convergence rate in this case is faster than the case with only one layer of bootstrapping, as it depends on the magnitude of $b'r'$ instead of b'.

Theorem 2 The likelihood of generating predictions for every point with the NestBoot estimator converges to 1 as the number of first level estimators b' tends to infinity.

Proof. First, we note $\Pr\{\sum_{i=1}^{n}\mathbb{I}_i = n\} = 1 - \Pr\{\sum_{b=1}^{b'}\sum_{r=1}^{r'} M_i^{b,r} = 0$ for at least one $i\}$. We now suppose that there exists an i for which $\sum_{b=1}^{b'}\sum_{r=1}^{r'} M_i^{b,r} = 0$ and show that the likelihood of this events drops off to zero with b' increasing. (For ease in readability, we simplify the summations' notations.) The proof is then complete since

$$\begin{aligned}\Pr\left\{\sum_b\sum_r M_i^{b,r}=0\right\} &= \Pr\left\{\sum_r M_i^{1,r}=0,\cdots,\sum_r M_i^{b',r}=0\right\}\\ &= \left(\Pr\left\{\sum_r M_i^{1,r}=0\right\}\right)^{b'} = \left(\Pr\left\{M_i^{1,1}=0\right\}\right)^{b'r'} \xrightarrow[b'r'\to\infty]{} 0, \qquad (8)\end{aligned}$$

due to non-negativity of $M_i^{b,r}$ and independence of first level estimators. □

Additionally, the average number of predictions for one point, knowing $\boldsymbol{M}^b \sim \text{Mult}(n, \boldsymbol{p}^a)$ and $\mathbb{E}[M_i^b] = 1$, can easily be computed as

$$\mathbb{E}\left[\sum_b\sum_r M_i^{b,r}\right] = b'r'\sum_{m=0}^{n}\mathbb{E}\left[M_i^{b,r}|M_i^b=m\right]\Pr\left\{M_i^b=m\right\} = b'r'\sum_{m=0}^{n} n\frac{m}{n}\Pr\left\{M_i^b=m\right\} = b'r'.$$

Recall that the main goal is to estimate $\theta(\boldsymbol{x}|F^c)$. Following Sun et al. (2011) and knowing (3), we can decompose each output as

$$q_i^{b,r}(\boldsymbol{x})M_i^{b,r} = \theta(\boldsymbol{x}|F^c) + (\theta(\boldsymbol{x}) - \theta(\boldsymbol{x}|F^c)) + \tau^b(\boldsymbol{x}) + \delta^{b,r}(\boldsymbol{x}) + \epsilon_i^{b,r}(\boldsymbol{x}), \qquad (9)$$

where $q_i^{b,r}(\boldsymbol{x}) = q(z_i, \boldsymbol{x}, \boldsymbol{Z}^{b,(r)})$. Next we define the random variables $N(\boldsymbol{x}) = \mathbb{E}_{Z_0}[q(Z_0, \boldsymbol{x}, \boldsymbol{Z})|\boldsymbol{Z} \sim F^{(0)}]$ and $L(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{Z}}\left[\mathbb{E}_{Z_0}\left[q(Z_0, \boldsymbol{x}, \boldsymbol{Z})|\boldsymbol{Z} \sim F^{(0)}\right]\right]$ (with F uncertain). Knowing these, the error terms become

$$\tau^b(\boldsymbol{x}) = L^b(\boldsymbol{x}) - \theta(\boldsymbol{x}),\ \delta^{b,r}(\boldsymbol{x}) = N^{b,r}(\boldsymbol{x}) - L^b(\boldsymbol{x}),\ \epsilon_i^{b,r}(\boldsymbol{x}) = W_i^{b,r}(\boldsymbol{x}) - N^{b,r}(\boldsymbol{x}), \qquad (10)$$

where $W_i^{b,r}(\boldsymbol{x}) = q_i^{b,r}(\boldsymbol{x})M_i^{b,r}$ for simplicity. We consider each output of the nested bootstrap to be $q_i^{b,r}(\boldsymbol{x})M_i^{b,r}$. Bootstrapping shuffles the density over all points and reassign probabilities to data points. This shuffling is one of the reasons for bootstrap's success in providing a stable and robust estimate of any desired statistic. Here we attempt to maintain the reassigns weights by multiplying $q_i^{b,r}(\boldsymbol{x})$ by the number of its repetition as our final output. Each error term introduced in (10) has mean zero and a constant variance and represents different sources of variabilities. The ϵ, δ, and τ denote the errors coming from estimation, modeling, and input data, respectively, and consequently their variances quantifies the variability of each source of uncertainty. We will discuss how to use the ANOVA (analysis of variance) method to estimate each of their variance, in the following section. The bias term can be written as $\theta(\boldsymbol{x}) - \theta(\boldsymbol{x}|F^c)$. Based on the bootstrap theorem (Efron 1979), we know that $\lim_{b'\to\infty}\theta(\boldsymbol{x}) - \theta(\boldsymbol{x}|F^c) = 0$, if n is large enough. However with limited simulation budget (b') and data points the bias can be quite significant. In section 3.2, we use von-Mises expansion (Fernholz 2012) with non-parametric probability distribution assumption to estimate the bias.

3 ANALYSIS OF VARIANCE AND BIAS IN NESTED BOOTSTRAPS

This section discusses the details of the proposed estimator along with its variance and bias estimation.

3.1 Analysis of Variance

Based on the previous section, one output of the NestBoot method is $W_i^{b,r}(\boldsymbol{x})$, and is defined in (9), where all error terms have zero means and estimable variances. Additionally, we assume each draw from $L(\boldsymbol{x})$ and $N(\boldsymbol{x})$ are i.i.d., and $\mathbb{E}\left[\tau^b(\boldsymbol{x})\delta^{b,r}(\boldsymbol{x})\right] = \mathbb{E}\left[\tau^b(\boldsymbol{x})\right]\mathbb{E}\left[\delta^{b,r}(\boldsymbol{x})\right] = \mathbb{E}\left[\epsilon_i^{b,r}(\boldsymbol{x})\tau^b(\boldsymbol{x})\right] = \mathbb{E}\left[\epsilon_i^{b,r}(\boldsymbol{x})\right]\mathbb{E}\left[\tau^b(\boldsymbol{x})\right] = \mathbb{E}\left[\epsilon_i^{b,r}(\boldsymbol{x})\delta^{b,r}(\boldsymbol{x})\right] = \mathbb{E}\left[\epsilon_i^{b,r}(\boldsymbol{x})\right]\mathbb{E}\left[\delta^{b,r}(\boldsymbol{x})\right] = 0$ which implies that all the covariance terms are zero. We use ANOVA to estimate the variance of our estimator. In the ANOVA, one need to first define sum of squares and find the relationship between them which then help in estimating the true variances. Taking averages from (9) with respect to points, test sets and data distributions we get

$$\begin{aligned}
W^{b,r}(\boldsymbol{x}) &= \sum_{i=1}^{n} W_i^{b,r}(\boldsymbol{x})/n = \theta(\boldsymbol{x}|F^c) + (\theta(\boldsymbol{x}) - \theta(\boldsymbol{x}|F^c)) + \tau^b(\boldsymbol{x}) + \delta^{b,r}(\boldsymbol{x}) + \epsilon^{b,r}(\boldsymbol{x}),\\
W^{b}(\boldsymbol{x}) &= \sum_{r=1}^{r'} W^{b,r}(\boldsymbol{x})/r' = \theta(\boldsymbol{x}|F^c) + (\theta(\boldsymbol{x}) - \theta(\boldsymbol{x}|F^c)) + \tau^b(\boldsymbol{x}) + \delta^{b}(\boldsymbol{x}) + \epsilon^{b}(\boldsymbol{x}),\\
\bar{W}(\boldsymbol{x}) &= \sum_{b=1}^{b'} W^{b}(\boldsymbol{x})/b' = \theta(\boldsymbol{x}|F^c) + (\theta(\boldsymbol{x}) - \theta(\boldsymbol{x}|F^c)) + \bar{\tau}(\boldsymbol{x}) + \bar{\delta}(\boldsymbol{x}) + \bar{\epsilon}(\boldsymbol{x}),
\end{aligned}$$

where $\epsilon^{b,r}(\boldsymbol{x}) = \sum_{i=1}^{n}\epsilon_i^{b,r}(\boldsymbol{x})/n$, $\epsilon^{b}(\boldsymbol{x}) = \sum_{r=1}^{r'}\epsilon^{b,r}(\boldsymbol{x})/r'$, $\bar{\epsilon}(\boldsymbol{x}) = \sum_{b=1}^{b'}\epsilon^{b}(\boldsymbol{x})/b'$, and similarly $\delta^{b}(\boldsymbol{x}) = \sum_{r=1}^{r'}\delta^{b,r}(\boldsymbol{x})/r'$, $\bar{\delta}(\boldsymbol{x}) = \sum_{b=1}^{b'}\delta^{b}(\boldsymbol{x})/b'$, $\bar{\tau}(\boldsymbol{x}) = \sum_{b=1}^{b'}\tau^{b}(\boldsymbol{x})/b'$. Having these we can now find the expected values for the proposed sum squares. So the desired sum squares can be defined as $SS_\tau(\boldsymbol{x}) = \sum_{b=1}^{b'}(W^b(\boldsymbol{x}) - \bar{W}(\boldsymbol{x}))^2$, $SS_\delta(\boldsymbol{x}) = \sum_{b=1}^{b'}\sum_{r=1}^{r'}(W^{b,r}(\boldsymbol{x}) - W^b(\boldsymbol{x}))^2$, $SS_\epsilon(\boldsymbol{x}) = \sum_{b=1}^{b'}\sum_{r=1}^{r'}\sum_{i=1}^{n}(W_i^{b,r}(\boldsymbol{x}) - W^{b,r}(\boldsymbol{x}))^2$. Starting from $SS_\epsilon(\boldsymbol{x})$ we can define its expectation as,

$$\begin{aligned}
\mathbb{E}\left[SS_\epsilon(\boldsymbol{x})\right] &= \sum_{b=1}^{b'}\sum_{r=1}^{r'}\sum_{i=1}^{n}\mathbb{E}\left[(W_i^{b,r}(\boldsymbol{x}) - W^{b,r}(\boldsymbol{x}))^2\right] = \sum_{b=1}^{b'}\sum_{r=1}^{r'}\sum_{i=1}^{n}\mathbb{E}\left[(\epsilon_i^{b,r}(\boldsymbol{x}) - \epsilon^{b,r}(\boldsymbol{x}))^2\right]\\
&= nb'r'\left(\sigma_\epsilon^2(\boldsymbol{x}) + \sigma_\epsilon^2(\boldsymbol{x})/n - 2\sigma_\epsilon^2(\boldsymbol{x})/n\right) = b'r'(n-1)\sigma_\epsilon^2(\boldsymbol{x}).
\end{aligned}$$

Similarly, the sum of squared errors at the modeling level (mid-level) can be achieved with,

$$\begin{aligned}
\mathbb{E}\left[SS_\delta(\boldsymbol{x})\right] &= \sum_{b=1}^{b'}\sum_{r=1}^{r'}\mathbb{E}\left[(W^{b,r}(\boldsymbol{x}) - W^{b}(\boldsymbol{x}))^2\right] = \sum_{b=1}^{b'}\sum_{r=1}^{r'}\mathbb{E}\left[((N^{b,r}(\boldsymbol{x}) - N^{b}(\boldsymbol{x})) + (\epsilon^{b,r}(\boldsymbol{x}) - \epsilon^{b}(\boldsymbol{x})))^2\right]\\
&= \sum_{b=1}^{b'}\sum_{r=1}^{r'}(\sigma_N^2(\boldsymbol{x}) - \sigma_N^2(\boldsymbol{x})/r') + (\frac{\sigma_\epsilon^2(\boldsymbol{x})}{n} - \frac{\sigma_\epsilon^2(\boldsymbol{x})}{nr'}) = b'(r'-1)\sigma_N^2(\boldsymbol{x}) + \frac{b'(r'-1)}{n}\sigma_\epsilon^2(\boldsymbol{x}).
\end{aligned}$$

Lastly, the expectation of the outer-level is similarly calculated as,

$$\mathbb{E}\left[SS_\tau(\boldsymbol{x})\right] = \sum_{b=1}^{b'}\mathbb{E}\left[(W^{b}(\boldsymbol{x}) - \bar{W}(\boldsymbol{x}))^2\right] = (b'-1)\sigma_L^2(\boldsymbol{x}) + \frac{b'-1}{r'}\sigma_N^2(\boldsymbol{x}) + \frac{b'-1}{nr'}\sigma_\epsilon^2(\boldsymbol{x}).$$

Hence we can estimate the variances as,

$$\begin{aligned}
&\hat{\sigma}_\epsilon^2(\boldsymbol{x}) = \frac{SS_\epsilon(\boldsymbol{x})}{b'r'(n-1)}, \quad \hat{\sigma}_N^2(\boldsymbol{x}) = \frac{SS_\delta(\boldsymbol{x}) - b'(r'-1)\hat{\sigma}_\epsilon^2(\boldsymbol{x})/n}{b'(r'-1)} = \frac{SS_\delta(\boldsymbol{x})}{b'(r'-1)} - \frac{SS_\epsilon(\boldsymbol{x})}{nb'r'(n-1)},\\
&\hat{\sigma}_L^2(\boldsymbol{x}) = \frac{SS_\tau(\boldsymbol{x})}{b'-1} - \frac{\hat{\sigma}_N^2(\boldsymbol{x})}{r'} - \frac{\hat{\sigma}_\epsilon^2(\boldsymbol{x})}{nr'} = \frac{SS_\tau(\boldsymbol{x})}{b'-1} - \frac{SS_\delta(\boldsymbol{x})}{b'r'(r'-1)}.
\end{aligned} \tag{11}$$

Note that the variance of the desired estimator is equivalent to $\sigma_L^2(\boldsymbol{x})$, and based on (11) is independent of estimation error. An interesting application of the variance estimates can be estimating the optimal b' and r', which we leave for the future research.

3.2 Bias Estimation Using Influence Functions

In practice, F^c is unknown so the bias term, i.e., $\hat{\theta}_{\text{NestBoot}}(\boldsymbol{x}) - \theta(\boldsymbol{x}|F^c)$ is not directly computable and needs to be estimated. We know using von-Mises expansion, which is similar to Taylor expansion but with the notion of Gateaux derivatives that

$$\theta(\boldsymbol{x}|F^c) = \theta(\boldsymbol{x}|F^b) + \int \text{IF}(\boldsymbol{z}, F^b) d(F^b(\boldsymbol{z}) - F^c(\boldsymbol{z})) + R_2(F^b, F^c), \tag{12}$$

where $\text{IF}(\boldsymbol{z}, F^b)$ is what is called the *influence function* and $R_2(F^b, F^c)$ is the second order approximation error between the two distributions, hence $R_2(F^b, F^c) = \mathcal{O}(\|F^b - F^c\|_2^2)$. Influence functions measures how much θ changes as the density of each point is slightly but properly (keeping the whole density in tact) upweighted. More precisely, $\text{IF}(\boldsymbol{z}, F^b) = \frac{\partial \theta(\boldsymbol{x}|F^b + \epsilon(\delta_{\boldsymbol{z}} - F^b))}{\partial \epsilon}|_{\epsilon=0}$.

Let $b_0 = n^n$ and $\mathcal{F}^n$ denote the *path* containing all b_0 possible distributions corresponding to the combinations of $\mathcal{M}^*$. Based on the bootstrap theory (Efron 1979), we claim that $\hat{\theta}_{\text{NestBoot}}(\boldsymbol{x}) - \theta(\boldsymbol{x}|\mathcal{F}^n) \to \theta(\boldsymbol{x}|\mathcal{F}^n) - \theta(\boldsymbol{x}|F^c)$, as $n \to \infty$ and $b' \to b_0$. Note that $\theta(\boldsymbol{x}|\mathcal{F}^n)$ is no longer random as it is the expectation over all possible F^b's. For the remainder of this paper we assume $\hat{\theta}_{\text{NestBoot}}(\boldsymbol{x})$ is smooth around the $\mathcal{F}^n$, since it is an average of b' bootstraps of the independent draws from the the ideal bootstrap or $\mathcal{F}^n$ (Efron 2014). Note that by "around the ideal distribution" we mean the pairwise distance between the two random vectors. Hence, the problem is reduced to estimating $\Delta(F^a, \mathcal{F}^n)$, and we can write

$$\begin{aligned}\Delta(\boldsymbol{x}, F^a, \mathcal{F}^n) = \hat{\theta}_{\text{NestBoot}}(\boldsymbol{x}) - \theta(\boldsymbol{x}|\mathcal{F}^n) =& \frac{\partial}{\partial \zeta}\theta(\boldsymbol{x}|F_\zeta)\bigg|_{\zeta=0} + \frac{1}{2}\frac{\partial^2}{\partial \zeta^2}\theta(\boldsymbol{x}|F_\zeta)\bigg|_{\zeta=0} + \mathcal{O}(\zeta^3) \\ =& \phi^{(1)}(\boldsymbol{x}) + \frac{1}{2}\phi^{(2)}(\boldsymbol{x}) + \mathcal{O}(\zeta^3),\end{aligned} \tag{13}$$

where $\phi^{(l)}(\boldsymbol{x})$ is the $l-$th order IF for $F_\zeta = \mathcal{F}^n + \zeta(F^a - \mathcal{F}^n)$ at $\zeta = 0$. Since ζ is between 0 and 1, the third term in (13) is negligible and can be omitted. Efron (2014) provides estimates of the first and the second order IF's for a simple estimation problem, where the estimator is a smooth functional of data. He suggests smoothing any estimator by taking an average over b' bootstraps of the available data. We build on his derivation to find an unbiased estimator for a prediction rule performance evaluation which is new in this paper.

Following Efron (2014), we can expand the expectations in $\hat{\theta}_{\text{NestBoot}}$ as summations over all possible pairs of M^b and $M^{b,r}$ ($b' = r' = n^n$), $\theta_{\text{NestBoot}} = \frac{1}{b'r'}\sum_b \sum_r \omega^{b,r} W^{b,r}(\boldsymbol{x})$, where

$$\omega^{b,r} = b'r'\mathbb{P}\{M^{b,r} = \mathbf{m}^{b,r} | M^b = \mathbf{m}^b\}\mathbb{P}\{M^b = \mathbf{m}^b\} = \prod_{i=1}^n \left(\frac{m_i^b}{n}\right)^{m_i^{b,r}} n^{m_i^{b,r}} p_i^{m_i^b} n^{m_i^b}.$$

Hence for a slight perturbation at point j, we define the resulting pmf as $\boldsymbol{p}_j(\zeta) = 1/n + \zeta(\boldsymbol{e}_j - 1/n)$, where $\boldsymbol{e}_j$ is a unit vector. Then

$$\begin{aligned}\omega^{b,r}(\boldsymbol{p}_j(\zeta)) &= \prod_{i=1}^n (m_i^b)^{m_i^{b,r}} \left(n(1/n + \zeta(\boldsymbol{e}_j - 1/n))\right)^{m_i^b} \approx \prod_{i=1}^n (n\|\boldsymbol{p}_j(\zeta)\boldsymbol{e}_i\|)^{m_i^{b,r}} (n\|\boldsymbol{p}_j(\zeta)\boldsymbol{e}_i\|)^{m_i^b} \\ &= \left(\prod_{i=1, i\neq j}^n (1-\zeta)^{m_i^{b,r}} (1-\zeta)^{m_i^b}\right) (1 + (n-1)\zeta)^{m_j^{b,r} + m_j^b} \approx 1 + n\zeta(m_j^b + m_j^{b,r} - 2),\end{aligned} \tag{14}$$

where the fourth approximation is due to the binomial expansion. Note that $\boldsymbol{p}_j(\zeta)$ is a valid pmf, since $\|\boldsymbol{p}_j(\zeta)\| = 1$ with positive elements between 0 and 1 (assuming $\zeta < 1$). Redefining our estimator under the perturbed distribution $F_j(\zeta)$ corresponding to $\boldsymbol{p}_j(\zeta)$,

$$\begin{aligned}\hat{\theta}(\boldsymbol{x}|F_j(\zeta)) &= \frac{1}{b'r'}\sum_b\sum_r (1+n\zeta(m_j^b+m_j^{b,r}-2))(W^{b,r}(\boldsymbol{x})-\theta(\boldsymbol{x})+\theta(\boldsymbol{x})) \\ &= \theta(\boldsymbol{x}) + \sum_b\sum_r (n\zeta(m_j^b+m_j^{b,r}-2))(W^{b,r}(\boldsymbol{x})-\theta(\boldsymbol{x}))/b'r' = \theta(\boldsymbol{x}) + n\zeta\mathrm{Cov}_j. \qquad (15)\end{aligned}$$

The second equality in (15) holds because of the strong law of large numbers. By replacing the estimators above in the following limits, we get $\phi_j^{(1)}(\boldsymbol{x}) = \lim_{\zeta\to 0}\zeta^{-1}\hat{\theta}(\boldsymbol{x}|F_j(\zeta)) - \theta(\boldsymbol{x}) = n\mathrm{Cov}_j(\boldsymbol{x})$.

To obtain the second order IF, we need to extend the approximation in (14) one step further, which results in $\omega^{b,r}(\boldsymbol{p}_j(\zeta)) \approx 1 + n\zeta(m_j^b+m_j^{b,r}-2) + n\zeta^2(m_j^b+m_j^{b,r}-2)^2/2$. Similar to (15) we get

$$\phi_j^{(2)}(\boldsymbol{x}) = \lim_{\zeta\to 0}\frac{\hat{\theta}(\boldsymbol{x}|1/n+\zeta(\delta_j-1/n)) - 2\theta(\boldsymbol{x}) + \hat{\theta}(\boldsymbol{x}|1/n-\zeta(\delta_j-1/n))}{\zeta^2} = n^2\mathrm{Cov}_j^*(\boldsymbol{x}),$$

where $\mathrm{Cov}_j^*(\boldsymbol{x}) = \sum_b\sum_r(m_j^b+m_j^{b,r}-2)^2(W^{b,r}(\boldsymbol{x})-\theta(\boldsymbol{x}))/b'r'$ is the covariance between the ratio of weights and simulation outputs, knowing $\mathbb{E}[m_j^b+m_j^{b,r}] = \mathbb{E}[\mathbb{E}[m_j^b+m_j^{b,r}|m_j^b]] = \mathbb{E}[2m_j^b] = 2$. We estimate $\theta(\boldsymbol{x})$ with $\overline{W}(\boldsymbol{x})$. Following chapter 6 of (Efron 1982),

$$\hat{\theta}_{\text{NestBoot}}(\boldsymbol{x}) - \theta(\boldsymbol{x}|F^c) = \frac{1}{2}\sum_{j=1}^{n}\frac{\phi_j^{(2)}(\boldsymbol{x})}{n^2} = \sum_b\sum_r\frac{(\sum_j(m_j^b+m_j^{b,r}-2)^2)(W^{b,r}(\boldsymbol{x})-\theta(\boldsymbol{x}))}{2b'r'} = \frac{\mathrm{Cov}^*}{2}.$$

Therefore, the debiased estimator can be computed as $\hat{\theta}_{\text{NestBoot}}(\boldsymbol{x}) - \mathrm{Cov}^*(\boldsymbol{x})/2$.

4 NUMERICAL EXPERIMENTS

To evaluate the proposed estimator and compare its performance with other methods, we experiment on a case study on the feature selection problem. We simulate two datasets with $n = 100$ rows for the training and an additional 100 points for out-of-sample validation. The first simulated dataset has 12 independent Gamma distributed features, with both shape and scale parameters set equal to 2. The second simulated dataset similarly has 12 variables, but their distribution is changed to Laplace. Both datasets follow equations introduced in (Van der Laan et al. 2007), with the addition of 23 standard normal noise features and 15 correlated (auto-regressive) features with a correlation value of 0.5. We refer to them as simulation 1 and simulation 2, respectively.

4.1 Case Study: Feature Selection (FS)

In machine learning, selecting the most informative and contributing features among the extensive set of independent variables is a challenging task. As a case study of the proposed work, we test our estimator within an optimization algorithm for evaluating each subset of features selected by the algorithm. This case study is a continuation of the simulation optimization-based FS algorithm introduced in (Vahdat and Shashaani 2020). In FS, the decision variable or $\boldsymbol{x}$ is a binary vector with size p, which equals the number of variables in a dataset. We employ a genetic algorithm (GA) (Goldberg and Holland 1988) search engine to go through different solution candidates and select the best. The GA hyperparameters were tuned using a Bayesian optimization framework. GA evaluates the candidate solution using an estimator at each search iteration and compares candidates with their evaluations. As a result, the estimator's robustness and stability play an essential role in guiding the search engine towards the best solution.

4.2 Results and Analysis

We perform an extensive comparison between different sampling methods and the nested bootstrap method proposed here. For each estimator, the average and standard deviation of the fitness function estimate ($\hat{\theta}$), along with average loss (ℓ) computing the mean squared difference between the predicted values using only the true features and those with the optimal features (capturing the robustness of the selected solution Sanchez and Sanchez (2020)), the number of optimal features (ν), and the average computation time.

Table 2: Summary results of FS case study on two simulated datasets where average and standard deviation of the performance measures are computed over 100 macro-replications with $b' = 10, r' = 5$, and $k = 10$. The average and standard deviation show the distribution of the corresponding method.

	Simulation 1				Simulation 2			
Method	$\hat{\theta}$	$\bar{\ell}$	$\bar{\nu}$	**Time**	$\hat{\theta}$	$\bar{\ell}$	$\bar{\nu}$	**Time**
Boot	12.4 ± 4.8	313	17.0 ± 2.8	1.6	71.7 ± 27.9	30.8	13.4 ± 3.5	1.5
632Boot	12.9 ± 4.9	344	19.2 ± 2.4	1.8	75.1 ± 27.7	31.3	16.4 ± 3.3	1.7
LkOBoot	12.9 ± 5.0	352	17.4 ± 2.9	1.7	71.9 ± 27.2	29.8	13.6 ± 3.3	1.9
repeated CV	13.4 ± 4.9	405	20.1 ± 3.0	1.7	78.0 ± 30.3	33.9	17.2 ± 3.7	1.6
LBootOBoot	13.4 ± 4.7	424	18.7 ± 2.9	1.1	74.6 ± 27.5	33.1	15.2 ± 3.9	1.1
NestBoot	11.6 ± 4.6	256	12.6 ± 1.6	1.6	66.7 ± 26.3	27.4	10.0 ± 2.0	1.6
Debiased NestBoot	12.4 ± 4.9	350	13.9 ± 1.6	1.6	67.6 ± 26.2	29.2	9.0 ± 2.1	1.5

The NestBoot has less MSE and loss in both datasets; hence it is more robust and accurate in our experiments without sacrificing time. The difference between NestBoot and other methods is tested using a χ^2 test with the significance level of 0.05. All the p-values lie below 0.05, showing the superiority of the proposed sampling method. However, the debiased NestBoot does not perform better than NestBoot, likely due to a sufficient amount of data in both simulations. LBootOBoot is the fastest among other methods because of its limited number of models (see Table 1), but with worse performance. Furthermore, we could bring the average number of selected features down to 13 from 17 in the best case of the existing methods in simulation 1 and to 10 from 14 in simulation 2. The NestBoot's success in identifying informative features is due to more robust and accurate estimates of the loss function values at each iteration.

5 CONCLUSION

This paper proposes a novel estimator, NestBoot, for evaluating predictive rules' performance in a simulation-optimization setting. The NestBoot is not the first method to use a second level of bootstrapping; however, nested bootstrapping samples the test set first to compute the weighted average as the output is the novelty of our method. The NestBoot is more robust against changes in the input data as it takes the input variability into account by conditioning. The estimator's variance calculated using ANOVA appears to be independent of estimation error. We employed the nonparametric delta method to loosely estimate the bias and deduct that from the proposed estimator; however, it does not improve the performance in our numerical experiments. Improving the approximations that lead to bias estimation and finding the optimum number of bootstraps at each level is our future focus.

REFERENCES

Barton, R. R., H. Lam, and E. Song. 2018. "Revisiting Direct Bootstrap Resampling for Input Model Uncertainty". In *Proceedings of the 2018 Winter Simulation Conference*, 1635–1645. Gothenburg, Sweden: Institute of Electrical and Electronics Engineers, Inc.

Bengio, Y., and Y. Grandvalet. 2005. *Bias in Estimating the Variance of K-Fold Cross-Validation*, 75–95. Boston, MA: Springer US.

Corlu, C. G., and B. Biller. 2015. “Subset Selection for Simulations Accounting for Input Uncertainty”. In *Proceedings of the 2015 Winter Simulation Conference*, 437–446: IEEE.

Efron, B. 1979. “Bootstrap Methods: Another Look at the Jackknife”. *The Annals of Statistics* 7(1):1–26.

Efron, B. 1982. *The Jackknife, the Bootstrap and Other Resampling Plans*. Society for Industrial and Applied Mathematics.

Efron, B. 1983. “Estimating the error rate of a prediction rule: improvement on cross-validation”. *Journal of the American statistical association* 78(382):316–331.

Efron, B. 2014. “Estimation and Accuracy After Model Selection”. *Journal of the American Statistical Association* 109(507):991–1007.

Efron, B., and R. Tibshirani. 1997. “Improvements on Cross-validation: The 632+ Bootstrap Method”. *Journal of the American Statistical Association* 92(438):548–560.

Fernholz, L. T. 2012. *Von Mises calculus for statistical functionals*, Volume 19. Springer Science & Business Media.

Fisher, A., and E. H. Kennedy. 2020. “Visually Communicating and Teaching Intuition for Influence Functions”. *The American Statistician* 0(0):1–11.

Goldberg, D. E., and J. H. Holland. 1988. “Genetic Algorithms and Machine Learning”. *Machine Learning* 3(2-3):95–99.

Lam, H. 2016. “Advanced Tutorial: Input Uncertainty and Robust Analysis in Stochastic Simulation”. In *Proceedings of the 2016 Winter Simulation Conference*, 178–192. Arlington, Virginia: Institute of Electrical and Electronics Engineers, Inc.

Lam, H., and E. Zhou. 2017. “The empirical likelihood approach to quantifying uncertainty in sample average approximation”. *Operations Research Letters* 45:301 – 307.

Morgan, L. E., B. L. Nelson, A. C. Titman, and D. Worthington. 2019. “Detecting bias due to input modelling in computer simulation”. *European Journal of Operational Research* 279(3):869 – 881.

Sanchez, S. M., and P. J. Sanchez. 2020. “Robustness Revisited: Simulation Optimization Viewed Through A Different Lens”. In *Proceedings of the 2015 Winter Simulation Conference*, 60–74.

Song, E., and B. L. Nelson. 2019. “Input–Output Uncertainty Comparisons for Discrete Optimization via Simulation”. *Operations Research* 67(2):562–576.

Sun, Y., D. W. Apley, and J. Staum. 2011. “Efficient Nested Simulation for Estimating the Variance of a Conditional Expectation”. *Operations Research* 59(4):998–1007.

Vahdat, K., and S. Shashaani. 2020. “Simulation Optimization Based Feature Selection, A Study on Data-Driven Optimization with Input Uncertainty”. In *Proceedings of the 2020 Winter Simulation Conference*, 2149–2160. IEEE.

Van der Laan, M. J., E. C. Polley, and A. E. Hubbard. 2007. “Super Learner”. *Statistical applications in genetics and molecular biology* 6(1).

AUTHOR BIOGRAPHIES

KIMIA VAHDAT is a third year Ph.D. student in the Edward P. Fitts Department of Industrial and System Engineering at North Carolina State University. Her main research is in the combination of stochastic optimization with data analysis. Her email address is kvahdat@ncsu.edu.

SARA SHASHAANI is an assistant professor in the Edward P. Fitts Department of Industrial and System Engineering at North Carolina State University. Her research interests are probabilistic data-driven modeling and simulation optimization. Her email address is sshasha2@ncsu.edu and her homepage is https://shashaani.wordpress.ncsu.edu/.