The Annals of Statistics
2019, Vol. 47, No. 2, 1088–1112
https://doi.org/10.1214/18-AOS1707

ESTIMATING THE ALGORITHMIC VARIANCE OF RANDOMIZED ENSEMBLES VIA THE BOOTSTRAP[1]

BY MILES E. LOPES

University of California, Davis

Although the methods of bagging and random forests are some of the most widely used prediction methods, relatively little is known about their algorithmic convergence. In particular, there are not many theoretical guarantees for deciding when an ensemble is "large enough"—so that its accuracy is close to that of an ideal infinite ensemble. Due to the fact that bagging and random forests are randomized algorithms, the choice of ensemble size is closely related to the notion of "algorithmic variance" (i.e., the variance of prediction error due only to the training algorithm). In the present work, we propose a bootstrap method to estimate this variance for bagging, random forests and related methods in the context of classification. To be specific, suppose the training dataset is fixed, and let the random variable ERR_t denote the prediction error of a randomized ensemble of size t. Working under a "first-order model" for randomized ensembles, we prove that the centered law of ERR_t can be consistently approximated via the proposed method as $t \to \infty$. Meanwhile, the computational cost of the method is quite modest, by virtue of an extrapolation technique. As a consequence, the method offers a practical guideline for deciding when the algorithmic fluctuations of ERR_t are negligible.

1. Introduction. Random forests and bagging are some of the most widely used prediction methods [Breiman (1996, 2001)], and over the course of the past fifteen years, much progress has been made in analyzing their statistical performance [Biau (2012), Biau, Devroye and Lugosi (2008), Bühlmann and Yu (2002), Hall and Samworth (2005), Scornet, Biau and Vert (2015)]. However, from a computational perspective, relatively little is understood about the *algorithmic convergence* of these methods, and in practice, ad hoc criteria are generally used to assess this convergence.

To clarify the idea of algorithmic convergence, recall that when bagging and random forests are used for classification, a large collection of t randomized classifiers is trained, and then new predictions are made by taking the plurality vote of the classifiers. If such a method is run several times on the same training data $\mathcal{D}$, the prediction error ERR_t of the ensemble will vary with each run, due to the randomized training algorithm. As the ensemble size increases ($t \to \infty$) with $\mathcal{D}$

Received October 2017; revised February 2018.
[1]Supported in part by NSF Grant DMS-1613218.
MSC2010 subject classifications. Primary 62F40; secondary 65B05, 68W20, 60G25.
Key words and phrases. Bootstrap, random forests, bagging, randomized algorithms.

held fixed, the random variable ERR_t typically decreases and eventually stabilizes at a limiting value $\mathrm{err}_\infty = \mathrm{err}_\infty(\mathcal{D})$. In this way, an ensemble reaches algorithmic convergence when its prediction error nearly matches that of an infinite ensemble trained on the same data.

Meanwhile, with regard to computational cost, larger ensembles are more expensive to train, to store in memory, and to evaluate on unlabeled points. For this reason, it is desirable to have a quantitative guarantee that an ensemble of a given size will perform nearly as well as an infinite one. This type of guarantee also prevents wasted computation, and assures the user that extra classifiers are unlikely to yield much improvement in accuracy.

1.1. *Contributions and related work.* To measure algorithmic convergence, we propose a new bootstrap method for approximating the distribution $\mathcal{L}(\sqrt{t}(\mathrm{ERR}_t - \mathrm{err}_\infty)|\mathcal{D})$ as $t \to \infty$. Such an approximation allows the user to decide when the algorithmic fluctuations of ERR_t around err_∞ are negligible. If particular, if we refer to the *algorithmic variance*

$$\sigma_t^2 := \mathrm{var}(\mathrm{ERR}_t|\mathcal{D}) = \mathbb{E}\big[\mathrm{ERR}_t^2|\mathcal{D}\big] - \big(\mathbb{E}[\mathrm{ERR}_t|\mathcal{D}]\big)^2,$$

as the variance of ERR_t due only the training algorithm, then the parameter σ_t is a concrete measure of convergence that can be estimated via the bootstrap. In addition, the computational cost of the method turns out to be quite modest, by virtue of an extrapolation technique, as described in Section 4.

Although the bootstrap is an established approach to distributional approximation and variance estimation, our work applies the bootstrap in a relatively novel way. Namely, the method is based on "bootstrapping an algorithm," rather than "bootstrapping data"—and in essence, we are applying an inferential method in order to serve a computational purpose. The opportunities for applying this perspective to other randomized algorithms can also be seen in the papers of Byrd et al. (2012), Lopes, Wang and Mahoney (2017, 2018), which deal with stochastic gradient methods, as well as randomized versions of matrix multiplication and least squares.

Bootstrap consistency. From a theoretical standpoint, our main result (Theorem 3.1) shows that the proposed method consistently approximates the distribution $\mathcal{L}(\sqrt{t}(\mathrm{ERR}_t - \mathrm{err}_\infty)|\mathcal{D})$ as $t \to \infty$ under a "first-order model" for randomized ensembles. The proof also offers a couple of theoretical contributions related to Hadamard differentiability and the functional delta method (van der Vaart and Wellner (1996), Chapter 3.9). The first ingredient is a lifting operator $\boldsymbol{L}$, which transforms a univariate empirical c.d.f. $\mathbb{F}_t : [0, 1] \to [0, 1]$ into a multivariate analogue $\boldsymbol{L}(\mathbb{F}_t) : \Delta \to \Delta$, where Δ is a simplex. In addition to having interesting properties in its own right, the lifting operator will allow us to represent ERR_t as a functional of an empirical process. The second ingredient is the calculation of

this functional's Hadamard derivative, which leads to a surprising connection with the classical *first variation formula* for smooth manifolds (Simon (1983), White (2016)).[2]

To briefly comment on the role of this formula in our analysis, consider the following informal statement of it. Let $\mathcal{M}$ be a smooth d-dimensional manifold contained in $\mathbb{R}^d$, and let $\{f_\delta\}_{\delta\in(-1,1)}$ be a one-parameter family of diffeomorphisms $f_\delta : \mathcal{M} \to f_\delta(\mathcal{M}) \subset \mathbb{R}^d$, satisfying $f_\delta \to \mathrm{id}_{\mathcal{M}}$ as $\delta \to 0$, where $\mathrm{id}_{\mathcal{M}}$ denotes the identity map on $\mathcal{M}$. Then

$$\frac{d}{d\delta}\,\mathrm{vol}\big(f_\delta(\mathcal{M})\big)\Big|_{\delta=0} = \int_{\mathcal{M}} \mathrm{div}(Z)(\theta)\,d\theta, \tag{1.1}$$

where $\mathrm{vol}(\cdot)$ is a volume measure, the symbol $\mathrm{div}(Z)$ denotes the divergence of the vector field $Z(\theta) := \frac{\partial}{\partial\delta} f_\delta(\theta)|_{\delta=0}$ and the symbol $d\theta$ is a volume element on $\mathcal{M}$. In our analysis, it is necessary to adapt this result to a situation where the maps f_δ are nonsmooth, the manifold $\mathcal{M}$ is a nonsmooth subset of Euclidean space and the vector field $Z(\cdot)$ is a nonsmooth Gaussian process. Furthermore, applying a version of Stokes' theorem to the right-hand side of equation (1.1) leads to a particular linear functional of $Z(\cdot)$, which turns out to be the Hadamard derivative relevant to understanding ERR_t. A more detailed explanation of this connection is given below equation (B.1) in Appendix B.

Related work. In the setting of binary classification, a few papers analyze the bias $\mathbb{E}[\mathrm{ERR}_t - \mathrm{err}_\infty\,|\mathcal{D}]$, and show that it converges at the fast rate of $1/t$ under various conditions (Cannings and Samworth (2017), Lopes (2016), Ng and Jordan (2001)). A couple of other works study alternative measures of convergence. For instance, the paper of Lam and Suen (1997) considers the probability that the majority vote commits an error at a fixed test point, and the paper of Hernández-Lobato, Martínez-Muñoz and Suárez (2013) provides an informal analysis of the probability that an ensemble of size t disagrees with an infinite ensemble at a random test point, but these approaches do not directly control ERR_t. In addition, some empirical studies of algorithmic convergence may be found in the papers of Latinne, Debeir and Decaestecker (2001), Oshiro, Perez and Baranauskas (2012).

Among the references just mentioned, the ones that are most closely related to the current paper are Lopes (2016) and Cannings and Samworth (2017). These works derive theoretical upper bounds on $\mathrm{var}(\mathrm{ERR}_t|\mathcal{D})$ or $\mathrm{var}(\mathrm{ERR}_{t,l}|\mathcal{D})$, where $\mathrm{ERR}_{t,l}$ is the error rate on a particular class l (cf. Section 4). The paper of Lopes (2016) also proposes a method to estimate the unknown parameters in such bounds. In relation to these works, the current paper differs in two significant ways. First, we offer an approximation to the full distribution $\mathcal{L}(\mathrm{ERR}_t - \mathrm{err}_\infty\,|\mathcal{D})$, and hence provide a *direct estimate* of algorithmic variance, rather than a bound. Second, the

[2]Further examples of problems where geometric analysis plays a role in understanding the performance of numerical algorithms may be found in the book of Bürgisser and Cucker (2013).

method proposed here is relevant to a wider range of problems, since it can handle any number of classes, whereas the analyses in Lopes (2016) and Cannings and Samworth (2017) are specialized to the binary setting. Moreover, the theoretical analysis of the bootstrap approach is entirely different from the previous techniques used in deriving variance bounds.

Outside of the setting of randomized ensemble classifiers, the papers of Arlot and Genuer (2014), Mentch and Hooker (2016), Scornet (2016a), Sexton and Laake (2009), Wager, Hastie and Efron (2014) look at the algorithmic fluctuations of ensemble regression functions at a fixed test point.

1.2. *Background and setup*. We consider the general setting of a classification problem with $k \geq 2$ classes. The set of training data is denoted $\mathcal{D} := \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, which is contained in a sample space $\mathcal{X} \times \mathcal{Y}$. The feature space $\mathcal{X}$ is arbitrary, and the space of labels $\mathcal{Y}$ has cardinality k. An ensemble of t classifiers is denoted by $Q_i : \mathcal{X} \to \mathcal{Y}$, with $i = 1, \ldots, t$.

Randomized ensembles. The key issue in studying the algorithmic convergence of bagging and random forests is randomization. In the method of bagging, randomization is introduced by generating random sets $\mathcal{D}_1^*, \ldots, \mathcal{D}_t^*$, each of size n, via sampling with replacement from $\mathcal{D}$. For each $i = 1, \ldots, t$, a classifier Q_i is trained on $\mathcal{D}_i^*$, with the same classification method being used each time. When each Q_i is trained with a decision tree method (such as CART [Breiman et al. (1984)]), the random forests procedure extends bagging by adding a randomized feature selection rule (Breiman (2001)).

It is helpful to note that the classifiers in bagging and random forests can be represented in a common way. Namely, there is a deterministic function, say g, such that for any fixed $x \in \mathcal{X}$, each classifier Q_i can be written as

$$Q_i(x) = g(x, \mathcal{D}, \xi_i), \tag{1.2}$$

where $\xi_1, \xi_2, \ldots$, is an i.i.d. sequence of random objects, independent of $\mathcal{D}$, that specify the "randomizing parameters" of the classifiers [cf. Breiman (2001), Definition 1.1]. For instance, in the case of bagging, the object ξ_i specifies the randomly chosen points in $\mathcal{D}_i^*$.

Beyond bagging and random forests, our proposed method will be generally applicable to ensembles that can be represented in the form (1.2), such as those in Bühlmann and Yu (2002), Dietterich (2000), Ho (1998). This representation should be viewed abstractly, and it is not necessary for the function g or the objects $\xi_1, \xi_2, \ldots$ to be explicitly constructed in practice. Some further examples include a recent ensemble method based on random projections (Cannings and Samworth (2017)), as well as the voting Gibbs classifier (Ng and Jordan (2001)), which is a Bayesian ensemble method based on posterior sampling. More generally, if the functions $Q_1, Q_2, \ldots$ are i.i.d. conditionally on $\mathcal{D}$, then the ensemble can be represented in the form (1.2), as long as the classifiers lie in a standard Borel space

[Kallenberg (2006), Lemma 3.22]. Lastly, it is important to note that the representation (1.2) generally does not hold for classifiers generated by boosting methods (Schapire and Freund (2012)), for which the analysis of algorithmic convergence is quite different.

Plurality vote. For any $x \in \mathcal{X}$, we define the ensemble's *plurality vote* as the label receiving the largest number of votes among $Q_1(x), \ldots, Q_t(x)$. In the exceptional case of a tie, it will simplify technical matters to define the plurality vote as a symbol not contained in $\mathcal{Y}$, so that a tie always counts as an error. We also use the labeling scheme, $\mathcal{Y} := \{\boldsymbol{e}_0, \ldots, \boldsymbol{e}_{k-1}\} \subset \mathbb{R}^{k-1}$, where $\boldsymbol{e}_0 := \mathbf{0}$, and $\boldsymbol{e}_l$ is the lth standard basis vector for $l \geq 1$. One benefit of this scheme is that the plurality vote is determined by the average of the labels, $\bar{Q}_t(x) := \frac{1}{t}\sum_{i=1}^{t} Q_i(x)$. For this reason, we denote the plurality vote as $\mathrm{V}(\bar{Q}_t(x))$.

Error rate. Let $\nu = \mathcal{L}(X, Y)$ denote the distribution of a test point (X, Y) in $\mathcal{X} \times \mathcal{Y}$, drawn independently of $\mathcal{D}$ and $Q_1, \ldots, Q_t$. Then, for a particular realization of the classifiers $Q_1, \ldots, Q_t$, trained with the given set $\mathcal{D}$, the prediction error rate is defined as

$$\mathrm{ERR}_t := \int_{\mathcal{X}\times\mathcal{Y}} 1\{\mathrm{V}(\bar{Q}_t(x)) \neq y\}\, d\nu(x, y) = \mathbb{P}(\mathrm{V}(\bar{Q}_t(X)) \neq Y \mid \mathcal{D}, \boldsymbol{\xi}_t), \tag{1.3}$$

where $\boldsymbol{\xi}_t := (\xi_1, \ldots, \xi_t)$. (Classwise error rates $\mathrm{ERR}_{t,l}$, with $l = 0, \ldots, k-1$ will also be addressed in Section 4.1.) Here, it is crucial to note that ERR_t is a random variable, since $\bar{Q}_t$ is a random function. Indeed, the integral above shows that ERR_t is a functional of $\bar{Q}_t$. Moreover, there are two sources of randomness to consider: the algorithmic randomness arising from $\boldsymbol{\xi}_t$, and the randomness arising from the training set $\mathcal{D}$. Going forward, we will focus on the algorithmic fluctuations of ERR_t due to $\boldsymbol{\xi}_t$, and our analysis will always be conditional on $\mathcal{D}$.

Algorithmic variance. At first sight, it might not be obvious how to interpret the algorithmic fluctuations of ERR_t when $\mathcal{D}$ is held fixed. These fluctuations are illustrated below in Figure 1. The left panel shows how ERR_t changes as decision trees are added incrementally during a single run of the random forests algorithm. For the purposes of illustration, if we run the algorithm repeatedly on $\mathcal{D}$ to train many ensembles, we obtain a large number of sample paths of ERR_t as a function of t, shown in the right panel. Averaging the sample paths at each value of t produces the red curve, representing $\mathbb{E}[\mathrm{ERR}_t|\mathcal{D}]$ with $\boldsymbol{\xi}_t$ averaged out. Furthermore, the blue envelope curves for the sample paths are obtained by plotting $\mathbb{E}[\mathrm{ERR}_t|\mathcal{D}] \pm 3\sqrt{\mathrm{var}(\mathrm{ERR}_t|\mathcal{D})}$ as a function of t.

Problem formulation. Recall that the value $\mathrm{err}_\infty = \mathrm{err}_\infty(\mathcal{D})$ represents the ideal prediction error of an infinite ensemble trained on $\mathcal{D}$. Hence, a natural way of defining algorithmic convergence is to say that it occurs when t is large enough so

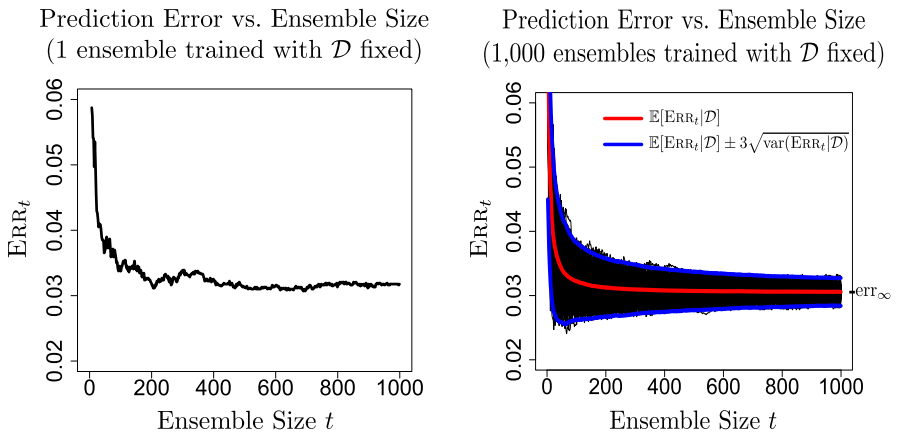

FIG. 1. *Left panel*: *The fluctuations of* ERR_t *for a single run of random forests on the "nursery data" (cf. Section 5). Right panel*: *The fluctuations of* ERR_t *for* 1000 *runs of random forests on the same data* (*i.e.*, 1000 *different realizations of the ensemble* $Q_1, \ldots, Q_t$). *For each ensemble, the value* ERR_t *was approximated by testing on the set of points denoted* $\mathcal{D}_{\text{ground}}$, *as described in Section 5.*

that the condition $|\mathrm{ERR}_t - \mathrm{err}_\infty| \le \epsilon$ holds with high probability, conditionally on $\mathcal{D}$, for some user-specified tolerance ϵ. However, the immediate problem we face is that it is not obvious how to check such a condition in practice.

From the right panel of Figure 1, we see that for most t, the inequality $|\mathrm{ERR}_t - \mathrm{err}_\infty| \le 3\sigma_t$ is highly likely to hold—and this observation can be formalized using Theorem 3.1 later on. For this reason, we propose to estimate σ_t as a route to measuring algorithmic convergence. It is also important to note that estimating the quantiles of $\mathcal{L}(\mathrm{ERR}_t - \mathrm{err}_\infty|\mathcal{D})$ would serve the same purpose, but for the sake of simplicity, we will focus on σ_t. In particular, there are at least two ways that an estimate $\widehat{\sigma}_t$ can be used in practice:

1. *Checking convergence for a given ensemble.* If an ensemble of a given size t_0 has been trained, then convergence can be checked by asking whether or not the observable condition $3\widehat{\sigma}_{t_0} \le \epsilon$ holds. Additional comments on possible choices for ϵ will be given shortly.

2. *Selecting t dynamically.* In order to make the training process as computationally efficient as possible, it is desirable to select the smallest t needed so that $|\mathrm{ERR}_t - \mathrm{err}_\infty| \le \epsilon$ is likely to hold. It turns out that this can be accomplished using an extrapolation technique, due to the fact that σ_t tends to scale like $1/\sqrt{t}$ (cf. Theorem 3.1). More specifically, if the user trains a small initial ensemble of size t_0 and computes an estimate $\widehat{\sigma}_{t_0}$, then "future" values of σ_t for $t \gg t_0$ can be estimated at no additional cost with the rescaled estimate $\sqrt{t_0/t}\,\widehat{\sigma}_{t_0}$. In other words, it is possible to look ahead and predict how many additional classifiers are needed to achieve $3\sigma_t \le \epsilon$. Additional details are given in Section 4.2.

Sources of difficulty in estimating σ_t. Having described the basic formulation of the problem, it is important to identify what challenges are involved in estimating σ_t. First, we must keep in mind that the parameter σ_t describes how ERR_t fluctuates over *repeated ensembles* generated from $\mathcal{D}$—and so it is not obvious that it is possible to estimate σ_t from the output of a *single ensemble*. Second, the computational cost to estimate σ_t should not outweigh the cost of training the ensemble, and consequently, the proposed method should be computationally efficient. These two obstacles will be described in Sections 3.2 and 4.2, respectively.

Remarks on the choice of error rate. Instead of analyzing the random variable ERR_t, a natural inclination would be to consider the "unconditional" error rate $\mathbb{E}[\mathrm{ERR}_t] = \mathbb{P}(\mathrm{V}(\bar{Q}_t(X)) \neq Y))$, which is likely more familiar, and reflects averaging over both $\mathcal{D}$ and the randomized algorithm. Nevertheless, there are a few reasons why the "conditional" error ERR_t may be more suitable for analyzing algorithmic convergence. First, the notion of algorithmic convergence typically describes how much computation should be applied to a given input—and in our context, the given input for the training algorithm is $\mathcal{D}$. Second, from an operational standpoint, once a user has trained an ensemble on a given dataset, their *actual* probability of misclassifying a future test point is ERR_t, rather than $\mathbb{E}[\mathrm{ERR}_t]$.

There is also a sense in which the convergence of $\mathbb{E}[\mathrm{ERR}_t]$ may be misleading. Existing theoretical results suggest that $\mathbb{E}[\mathrm{ERR}_t - \mathrm{err}_\infty]$ converges at the fast rate of $1/t$, and in the binary case, $k = 2$, it can be shown that $\mathbb{E}[\mathrm{ERR}_t - \mathrm{err}_\infty | \mathcal{D}] = \frac{1}{t}c(\mathcal{D}) + o(\frac{1}{t})$, for some number $c(\mathcal{D})$, under certain conditions (Cannings and Samworth (2017), Lopes (2016)). However, in Theorem 1 of Section 3, we show that conditionally on $\mathcal{D}$, the difference $\mathrm{ERR}_t - \mathrm{err}_\infty$ has fluctuations of order $1/\sqrt{t}$. In this sense, if the choice of t is guided only by $\mathbb{E}[\mathrm{ERR}_t]$ (rather than the fluctuations of ERR_t), then the user may be misled into thinking that algorithmic convergence occurs much faster than it really does—and this distinction is apparent from the red and blue curves in Figure 1.

Outline. Our proposed bootstrap method is described in Section 2, and our main consistency result is given in Section 3. Practical considerations are discussed in Section 4, numerical experiments are given in Section 5 and conclusions are stated in Section 6. The essential ideas of the proofs are explained in Appendices A and B, while the technical arguments are given Appendices C–E. Lastly, in Appendix F, we provide additional assessment of technical assumptions. All Appendices are in the Supplementary Material (Lopes (2019)).

2. Method. Based on the definition of ERR_t in equation (1.3), we may view ERR_t as a functional of $\bar{Q}_t$, denoted

$$\mathrm{ERR}_t = \varphi(\bar{Q}_t).$$

Algorithm 1 Bootstrap estimation of σ_t

For $b = 1, \dots, B$**:**

- Sample t classifiers $(Q_1^*, \dots, Q_t^*)$ with replacement from $(Q_1, \dots, Q_t)$.
- Compute $z_b := \widehat{\varphi}(\bar{Q}_t^*)$.

Return: the sample standard deviation of $z_1, \dots, z_B$, denoted $\widehat{\sigma}_t$.

From a statistical standpoint, the importance of this expression is that ERR_t is a functional of a sample mean, which makes it plausible that σ_t is amenable to bootstrapping, provided that φ is sufficiently smooth.

To describe the bootstrap method, let $(Q_1^*, \dots, Q_t^*)$ denote a random sample with replacement from the trained ensemble $(Q_1, \dots, Q_t)$, and put $\bar{Q}_t^*(\cdot) := \frac{1}{t}\sum_{i=1}^t Q_i^*(\cdot)$. In turn, it would be natural to regard the quantity

$$\mathrm{ERR}_t^* := \varphi\big(\bar{Q}_t^*\big) \tag{2.1}$$

as a bootstrap sample of ERR_t, but strictly speaking, this is an "idealized" bootstrap sample, because the functional φ depends on the unknown test point distribution $\nu = \mathcal{L}(X, Y)$. Likewise, in Section 2.1 below, we explain how each value $\varphi(\bar{Q}_t^*)$ can be estimated. So, in other words, if $\widehat{\varphi}$ denotes an estimate of φ, then an estimate of ERR_t would be written as

$$\widehat{\mathrm{ERR}}_t := \widehat{\varphi}(\bar{Q}_t),$$

and the corresponding bootstrap sample is

$$\widehat{\mathrm{ERR}}_t^* := \widehat{\varphi}\big(\bar{Q}_t^*\big).$$

Altogether, a basic version of the proposed bootstrap algorithm is summarized in Algorithm 1.

REMARK. While the above algorithm is conceptually simple, it suppresses most of the implementation details, and these are explained below. Also note that in order to approximate quantiles of $\mathcal{L}(\mathrm{ERR}_t - \mathrm{err}_\infty \,|\mathcal{D})$, rather than σ_t, it is only necessary to modify the last step, by returning the desired quantile of the centered values $z_1 - \bar{z}, \dots, z_B - \bar{z}$, with $\bar{z} = \frac{1}{B}\sum_{b=1}^B z_b$.

2.1. *Resampling algorithm with hold-out or "out-of-bag" points.* Here, we consider a version of Algorithm 1 where φ is estimated implicitly with hold-out points, and then later on, we will explain how hold-out points can be avoided using "out-of-bag" (OOB) points. To begin, suppose we have a hold-out set of size m, denoted $\mathcal{D}_{\mathrm{hold}} := \{(\tilde{X}_1, \tilde{Y}_1), \dots, (\tilde{X}_m, \tilde{Y}_m)\}$. Next, consider an array $\tilde{A}$ of size $t \times m$, whose ith row $\tilde{\mathbf{a}}_i$ is given by the predicted labels of Q_i on the hold-out points. That is,

$$\tilde{\mathbf{a}}_i := \big[Q_i(\tilde{X}_1), \dots, Q_i(\tilde{X}_m)\big], \tag{2.2}$$

Algorithm 2 Bootstrap estimation of σ_t with hold-out points

For $b = 1, \dots, B$**:**

- Draw a $t \times m$ array $\tilde{A}^*$ whose rows $(\tilde{\mathbf{a}}_1^*, \dots, \tilde{\mathbf{a}}_t^*)$ are sampled with replacement from $(\tilde{\mathbf{a}}_1, \dots, \tilde{\mathbf{a}}_t)$.
- Compute $z_b := \widehat{\mathrm{ERR}}(\tilde{A}^*)$.

Return: the sample standard deviation of $z_1, \dots, z_B$, denoted $\widehat{\sigma}_t$.

and

$$\tilde{A} := \begin{bmatrix} -\tilde{\mathbf{a}}_1- \\ \vdots \\ -\tilde{\mathbf{a}}_t- \end{bmatrix}. \tag{2.3}$$

The estimated error rate is easily computed as a function of this array, that is, $\widehat{\mathrm{ERR}}_t = \widehat{\mathrm{ERR}}_t(\tilde{A})$. To spell out the details, let $\tilde{A}_j = (Q_1(\tilde{X}_j), \dots, Q_t(\tilde{X}_j))$ denote the jth column of $\tilde{A}$, and with a slight abuse of our earlier notation, let $\mathbb{V}(\tilde{A}_j)$ denote the plurality vote of the labels in $\tilde{A}_j$. Then the estimated error rate is obtained from simple columnwise operations on $\tilde{A}$,

$$\widehat{\mathrm{ERR}}_t(\tilde{A}) := \frac{1}{m} \sum_{j=1}^{m} 1\{\mathbb{V}(\tilde{A}_j) \neq \tilde{Y}_j\}. \tag{2.4}$$

In other words, $\widehat{\mathrm{ERR}}_t(\tilde{A})$ is just the proportion of columns of $\tilde{A}$ for which plurality vote is incorrect. (Note that equation (2.4) is where φ is implicitly estimated.) Finally, since there is a one-to-one correspondence between the rows $\tilde{\mathbf{a}}_i$ and the classifiers Q_i, the proposed method is equivalent to resampling the rows $\tilde{\mathbf{a}}_i$, as given in Algorithm 2.

Extension to OOB points. Since the use of a hold-out set is often undesirable in practice, we instead consider OOB points—which are a special feature of bagging and random forests. To briefly review this notion, recall that each classifier Q_i is trained on a set of n points $\mathcal{D}_i^*$ obtained by sampling with replacement from $\mathcal{D}$. Consequently, each set $\mathcal{D}_i^*$ excludes approximately $(1 - \frac{1}{n})^n \approx 37\%$ of the points in $\mathcal{D}$, and these excluded points may be used as test points for the particular classifier Q_i. If a point X_j is excluded from $\mathcal{D}_i^*$, then we say "the point X_j is OOB for the classifer Q_i," and we write $i \in \mathrm{OOB}(X_j)$, where the set $\mathrm{OOB}(X_j) \subset \{1, \dots, t\}$ indexes the classifiers for which X_j is OOB.

In this notation, the error estimate $\widehat{\mathrm{ERR}}_t(\tilde{A})$ in Algorithm 2 can be given an analogous definition in terms of OOB points. Define a *new* $t \times n$ array A whose ith row is given by

$$\mathbf{a}_i := \big[Q_i(X_1), \dots, Q_i(X_n)\big].$$

Next, letting A_j be the jth column of A, define $\mathrm{V}_{\mathrm{O}}(A_j)$ to be the plurality vote of the set of labels $\{Q_i(X_j) \mid i \in \mathrm{OOB}(X_j)\}$. (If this set of labels is empty, then we treat this case as a tie, but this is unimportant, since it occurs with probability $[1-(1-\frac{1}{n})^n]^t \approx (0.63)^t$.) So, by analogy with $\widehat{\mathrm{ERR}}_t(\tilde{A})$, we define

$$\widehat{\mathrm{ERR}}_{t,\mathrm{O}}(A) := \frac{1}{n}\sum_{j=1}^{n} 1\{\mathrm{V}_{\mathrm{O}}(A_j) \neq Y_j\}. \tag{2.5}$$

Hence, the OOB version of Algorithm 2 may be implemented by simply interchanging $\tilde{A}$ and A, as well as $\widehat{\mathrm{ERR}}_t(\tilde{A}^*)$ and $\widehat{\mathrm{ERR}}_{t,\mathrm{O}}(A^*)$. The essential point to notice is that the sum in equation (2.5) is now over the training points in $\mathcal{D}$, rather than over the hold-out set $\mathcal{D}_{\mathrm{hold}}$, as in equation (2.4).

3. Main result. Our main theoretical goal is to prove that the bootstrap yields a consistent approximation of $\mathcal{L}(\sqrt{t}(\mathrm{ERR}_t - \mathrm{err}_\infty)|\mathcal{D})$ as t becomes large. Toward this goal, we will rely on two simplifications that are customary in analyses of bootstrap and ensemble methods. First, we will exclude the Monte Carlo error arising from the finite number of B bootstrap replicates, as well as the error arising from the estimation of ERR_t. For this reason, our results do not formally require the training or hold-out points to be i.i.d. copies of the test point (X, Y)—but from a practical standpoint, it is natural to expect that this type of condition should hold in order for Algorithm 2 (or its OOB version) to work well.

Second, we will analyze a simplified type of ensemble, which we will refer to as a *first-order model*. This type of approach has been useful in gaining theoretical insights into the behavior of complex ensemble methods in a variety of previous works [Arlot and Genuer (2014), Biau (2012), Biau, Devroye and Lugosi (2008), Genuer (2012), Lin and Jeon (2006), Scornet (2016a, 2016b)]. In our context, the value of this simplification is that it neatly packages the complexity of the base classifiers, and clarifies the relationship between t and quality of the bootstrap approximation. Also, even with such simplifications, the theoretical problem of proving bootstrap consistency still leads to considerable technical challenges. Lastly, it is important to clarify that the first-order model is introduced only for theoretical analysis, and our proposed method does not rely on this model.

3.1. *A first-order model for randomized ensembles.* Any randomized classifier $Q_1 : \mathcal{X} \to \{\boldsymbol{e}_0, \ldots, \boldsymbol{e}_{k-1}\}$ may be viewed as a stochastic process indexed by $\mathcal{X}$. From this viewpoint, we say that another randomized classifier $T_1 : \mathcal{X} \to \{\boldsymbol{e}_0, \ldots, \boldsymbol{e}_{k-1}\}$ is a *first-order model* for Q_1 if it has the same marginal distributions as Q_1, conditionally on $\mathcal{D}$, which means

$$\mathcal{L}\big(Q_1(x)|\mathcal{D}\big) = \mathcal{L}\big(T_1(x)|\mathcal{D}\big) \qquad \text{for all } x \in \mathcal{X}. \tag{3.1}$$

Since $Q_1(x)$ takes values in the finite set of binary vectors $\{\boldsymbol{e}_0, \ldots, \boldsymbol{e}_{k-1}\}$, the condition (3.1) is equivalent to

$$\mathbb{E}\big[Q_1(x)|\mathcal{D}\big] = \mathbb{E}\big[T_1(x)|\mathcal{D}\big] \qquad \text{for all } x \in \mathcal{X}, \tag{3.2}$$

where the expectation is only over the algorithmic randomness in Q_1 and T_1. A notable consequence of this matching condition is that the ensembles associated with Q_1 and T_1 have the *same* error rates on average. Indeed, if we let ERR'_t be the error rate associated with an ensemble of t independent copies of T_1, then it turns out that

$$\mathbb{E}[\mathrm{ERR}_t|\mathcal{D}] = \mathbb{E}\big[\mathrm{ERR}'_t|\mathcal{D}\big], \tag{3.3}$$

for all $t \geq 1$, where ERR_t is the error rate for $Q_1, \ldots, Q_t$, as before. (A short proof is given in Appendix E.) In this sense, a first-order model T_1 is a meaningful proxy for Q_1 with regard to statistical performance—even though the internal mechanisms of T_1 may be simpler.

3.1.1. *Constructing a first-order model.* Having stated some basic properties that are satisfied by any first-order model, we now construct a particular version that is amenable to analysis. Interestingly, it is possible to start with an *arbitrary* random classifier $Q_1 : \mathcal{X} \to \{\boldsymbol{e}_0, \ldots, \boldsymbol{e}_{k-1}\}$, and construct an associated T_1 in a relatively explicit way.

To do this, let $x \in \mathcal{X}$ be fixed, and consider the function

$$\vartheta(x) := \mathbb{E}\big[Q_1(x)|\mathcal{D}\big], \tag{3.4}$$

which takes values in the "full-dimensional" simplex $\Delta \subset \mathbb{R}^{k-1}$, defined by

$$\Delta := \big\{\theta \in [0,1]^{k-1} \mid \theta_1 + \cdots + \theta_{k-1} \leq 1\big\}.$$

For any fixed $\theta \in \Delta$, there is an associated partition of the unit interval into subintervals

$$I_0(\theta) \cup \cdots \cup I_{k-1}(\theta) = [0,1],$$

such that the width of interval $I_l(\theta)$ is equal to θ_l for $l \geq 1$. Namely, we put $I_1(\theta) := [0, \theta_1]$, and for $l = 2, \ldots, k-1$,

$$I_l(\theta) := \big((\theta_1 + \cdots + \theta_{l-1}), (\theta_1 + \cdots + \theta_l)\big].$$

Lastly, for I_0, we put

$$I_0(\theta) := \left(\sum_{l=1}^{k-1} \theta_l, 1\right].$$

Now, if we let $x \in \mathcal{X}$ be fixed, and let $U_1 \sim \mathrm{Uniform}[0,1]$, then we define $T_1(x) \in \{\boldsymbol{e}_0, \ldots, \boldsymbol{e}_{k-1}\}$ to have its lth coordinate equal to the following indicator variable:

$$\big[T_1(x)\big]_l := 1\big\{U_1 \in I_l\big(\vartheta(x)\big)\big\},$$

where $l = 1, \ldots, k-1$. It is simple to check that the first-order matching condition (3.2) holds, and so T_1 is indeed a first-order model of Q_1. Furthermore, given that

T_1 is defined in terms of a single random variable $U_1 \sim \text{Uniform}[0,1]$, we obtain a corresponding "first-order ensemble" $T_1, \dots, T_t$ via an i.i.d. sample of uniform variables $U_1, \dots, U_t$, which are independent of $\mathcal{D}$. (The lth coordinate of the ith classifier T_i is given by $[T_i(x)]_l = 1\{U_i \in I_l(\vartheta(x))\}$.) Hence, with regard to the representation $Q_i(x) = g(x, \mathcal{D}, \xi_i)$ in equation (1.2), we may make the identification

$$\xi_i = U_i,$$

when the first-order model holds with $Q_i = T_i$.

REMARK. To mention a couple of clarifications, the variables $U_1, \dots, U_t$ are only used for the construction of a first-order model, and they play no role in the proposed method. Also, even though the "randomizing parameters" $U_1, \dots, U_t$ are independent of $\mathcal{D}$, the classifiers $T_1, \dots, T_t$ still depend on $\mathcal{D}$ through the function $\vartheta(x) = \mathbb{E}[Q_1(x)|\mathcal{D}]$.

3.1.1.1. *Interpretation of first-order model.* To understand the statistical meaning of the first-order model, it is instructive to consider the simplest case of binary classification, $k = 2$. In this case, $T_1(x)$ is a Bernoulli random variable, where $T_1(x) = 1\{U_1 \le \vartheta(x)\}$. Since $\bar{Q}_t(x) \to \vartheta(x)$ almost surely as $t \to \infty$ (conditionally on $\mathcal{D}$), the majority vote of an infinite ensemble has a similar form, that is, $1\{\frac{1}{2} \le \vartheta(x)\}$. Hence, the classifiers $\{T_i\}$ can be viewed as "random perturbations" of the asymptotic majority vote arising from $\{Q_i\}$. Furthermore, if we view the number $\vartheta(x)$ as score to be compared with a threshold, then the variable U_i plays the role of a random threshold whose expected value is $\frac{1}{2}$. Lastly, even though the formula $T_1(x) = 1\{U_1 \le \vartheta(x)\}$ might seem to yield a simplistic classifier, the complexity of T_1 is actually wrapped up in the function ϑ. Indeed, the matching condition (3.2) allows for the function ϑ to be arbitrary.

3.2. *Bootstrap consistency.* We now state our main result, which asserts that the bootstrap "works" under the first-order model. To give meaning to bootstrap consistency, we first review the notion of conditional weak convergence.

Conditional weak convergence. Let λ_0 be a probability distribution on $\mathbb{R}$, and let $\{\lambda_{\boldsymbol{\xi}_t}\}_{t \ge 1}$ be a sequence of probability distributions on $\mathbb{R}$ that depend on the randomizing parameters $\boldsymbol{\xi}_t = (\xi_1, \dots, \xi_t)$. Also, let d_{BL} be the bounded Lipschitz metric for distributions on $\mathbb{R}$ (van der Vaart and Wellner (1996), Section 1.12) and let $\mathbb{P}_{\boldsymbol{\xi}}$ be the joint distribution of $(\xi_1, \xi_2, \dots)$. Then, as $t \to \infty$, we say that $\lambda_{\boldsymbol{\xi}_t} \xrightarrow{w} \lambda_0$ in $\mathbb{P}_{\boldsymbol{\xi}}$-probability if the sequence $\{d_{\text{BL}}(\lambda_{\boldsymbol{\xi}_t}, \lambda_0)\}_{t \ge 1}$ converges to 0 in $\mathbb{P}_{\boldsymbol{\xi}}$-probability.

REMARK. If a test point X is drawn from class $Y = \boldsymbol{e}_l$, then we denote the distribution of the random vector $\vartheta(X)$, conditionally on $\mathcal{D}$, as

$$\mu_l := \mathcal{L}\big(\vartheta(X)|\mathcal{D}, Y = \boldsymbol{e}_l\big),$$

which is a distribution on the simplex $\Delta \subset \mathbb{R}^{k-1}$. Since this distribution plays an important role in our analysis, it is worth noting that the properties of μ_l are *not affected* by the assumption of a first-order model, since $\vartheta(x) = \mathbb{E}[T_1(x)|\mathcal{D}] = \mathbb{E}[Q_1(x)|\mathcal{D}]$ for all $x \in \mathcal{X}$. We will also assume that the measures μ_l satisfy the following extra regularity condition.

ASSUMPTION 1. For the given set $\mathcal{D}$, and each $l = 0, \ldots, k-1$, the distribution μ_l has a density $f_l : \Delta \to [0, \infty)$ with respect to Lebesgue measure on Δ, and f_l is continuous on Δ. Also, if Δ° denotes the interior of Δ, then for each l, the density f_l is C^1 on Δ°, and $\|\nabla f_l\|_2$ is bounded on Δ°.

To interpret this assumption, consider a situation where the classwise test point distributions $\mathcal{L}(X|Y = \boldsymbol{e}_l)$ have smooth densities on $\mathcal{X} \subset \mathbb{R}^p$ with respect to Lebesgue measure. In this case, the density f_l will exist as long as ϑ is sufficiently smooth (cf. Appendix F, Proposition F.1). Still, Assumption 1 might seem unrealistic in the context of random forests, because ϑ is obtained by averaging over all decision trees that can be generated from $\mathcal{D}$, and strictly speaking, this is a finite average of nonsmooth functions. However, due to the bagging mechanism in random forests, the space of trees that can be generated from $\mathcal{D}$ is very large, and consequently, the function ϑ represents a very fine-grained average. Indeed, the idea that bagging is actually a "smoothing operation" on nonsmooth functions has received growing attention over the years (Bühlmann and Yu (2002), Buja and Stuetzle (2000), Buja and Stuetzle (2006), Efron (2014)), and the recent paper of Efron (2014) states that bagging is "also known as bootstrap smoothing." In Appendix F, we provide additional assessment of Assumption 1 in terms of both theoretical and empirical examples.

THEOREM 3.1 (Bootstrap consistency). *Suppose that the first-order model $Q_i = T_i$ holds for all $i \geq 1$, and that Assumption 1 holds. Then, for the given set $\mathcal{D}$, there are numbers $\mathrm{err}_\infty = \mathrm{err}_\infty(\mathcal{D})$ and $\sigma = \sigma(\mathcal{D})$ such that as $t \to \infty$,*

$$\mathcal{L}\big(\sqrt{t}(\mathrm{Err}_t - \mathrm{err}_\infty)|\mathcal{D}\big) \xrightarrow{w} N\big(0, \sigma^2\big), \tag{3.5}$$

and furthermore,

$$\mathcal{L}\big(\sqrt{t}(\mathrm{Err}_t^* - \mathrm{Err}_t)|\mathcal{D}, \boldsymbol{\xi}_t\big) \xrightarrow{w} N\big(0, \sigma^2\big) \qquad \text{in } \mathbb{P}_{\boldsymbol{\xi}}\text{-probability.}$$

REMARKS. In a nutshell, the proof of Theorem 3.1 is composed of three pieces: showing that ERR_t can be represented as a functional of an empirical process (Appendix A.1), establishing the smoothness of this functional (Appendix A.2) and employing the functional delta method (Appendix A.3). With regard to theoretical techniques, there are two novel aspects of the proof. The problem of deriving this functional is solved by introducing a certain lifting operator, while the problem of showing smoothness is based on a nonsmooth instance of the first-variation formula, as well as some special properties of Bernstein polynomials. Lastly, it is worth mentioning that the core technical result of the paper is Theorem A.1.

To mention some consequences of Theorem 3.1, the fact that the limiting distribution of $\mathcal{L}(\mathrm{ERR}_t - \mathrm{err}_\infty | \mathcal{D})$ has mean 0 shows that the fluctuations of ERR_t have more influence on algorithmic convergence than the bias $\mathbb{E}[\mathrm{ERR}_t - \mathrm{err}_\infty | \mathcal{D}]$ (as illustrated in Figure 1). Second, the limiting distribution motivates a convergence criterion of the form $3\sigma_t \leq \epsilon$, since asymptotic normality it indicates that the event $|\mathrm{ERR}_t - \mathrm{err}_\infty| \leq 3\sigma_t$ should occur with high probability when t is large, and again, this is apparent in Figure 1. Lastly, the theorem implies that the quantiles of $\mathcal{L}(\mathrm{ERR}_t - \mathrm{err}_\infty | \mathcal{D})$ agree asymptotically with their bootstrap counterparts. This is of interest, because quantiles allow the user to specify a bound on $\mathrm{ERR}_t - \mathrm{err}_\infty$ that holds with a tunable probability. Quantiles also provide an alternative route to estimating algorithmic variance, because if r_t^* denotes the interquartile range of $\mathcal{L}(\mathrm{ERR}_t^* - \mathrm{ERR}_t | \mathcal{D}, \boldsymbol{\xi}_t)$, then the theorem implies $\frac{\sqrt{t}}{c} r_t^* \longrightarrow \sigma$ in $\mathbb{P}_{\boldsymbol{\xi}}$-probability, where $c = \Phi^{-1}(3/4) - \Phi^{-1}(1/4)$.

4. Practical considerations. In this section, we discuss some considerations that arise when the proposed method is used in practice, such as the choice of error rate, the computational cost and the choice of a stopping criterion for algorithmic convergence.

4.1. *Extension to classwise error rates.* In some applications, classwise error rates may be of greater interest than the total error rate ERR_t. For any $l = 0, \ldots, k-1$, let $\nu_l = \mathcal{L}(X | Y = \boldsymbol{e}_l)$ denote the distribution of the test point X given that it is drawn from class l. Then the error rate on class l is defined as

$$\begin{aligned} \mathrm{ERR}_{t,l} &:= \int_{\mathcal{X}} 1\{\mathbb{V}(\bar{Q}_t(x)) \neq \boldsymbol{e}_l\} \, d\nu_l(x) \\ &= \mathbb{P}(\mathbb{V}(\bar{Q}_t(X)) \neq \boldsymbol{e}_l \mid \mathcal{D}, \boldsymbol{\xi}_t, Y = \boldsymbol{e}_l), \end{aligned} \tag{4.1}$$

and the corresponding algorithmic variance is

$$\sigma_{t,l}^2 := \mathrm{var}(\mathrm{ERR}_{t,l} | \mathcal{D}).$$

In order to estimate $\sigma_{t,l}$, Algorithm 2 can be easily adapted using either hold-out or OOB points from a particular class. Our theoretical analysis also extends immediately to the estimation of $\sigma_{t,l}$ (cf. Section A.1).

4.2. *Computational cost and extrapolation.* A basic observation about Algorithm 2 is that it only relies on the array of predicted labels $\tilde{A}$ (or alternatively A). Consequently, the algorithm does not require any retraining of the classifiers. Also, with regard to computing the arrays $\tilde{A}$ or A, at least one of these is *typically computed anyway* when evaluating an ensemble's performance with hold-out or OOB points—and so the cost of obtaining $\tilde{A}$ or A will typically not be an added expense of Algorithm 2. Third, the algorithm can be parallelized, since the bootstrap replicates can be computed independently. Lastly, the cost of the algorithm is *dimension-free* with respect to the feature space $\mathcal{X}$, since all operations are on the arrays $\tilde{A}$ or A, whose sizes do not depend on the number of features.

To measure the cost of Algorithm 2 in terms of floating point operations, it is simple to check that at each iteration $b = 1, \ldots, B$, the cost of evaluating $\widehat{\mathrm{ERR}}_t(\tilde{A}^*)$ is of order $t \cdot m$, since $\tilde{A}$ has m columns, and each evaluation of the plurality voting function has cost $\mathcal{O}(t)$. Hence, if the arrays $\tilde{A}$ or A are viewed as given, and if $m = \mathcal{O}(n)$, then the cost of Algorithm 2 is $\mathcal{O}(B \cdot t \cdot n)$, for either the hold-out or OOB versions. Below, we describe how this cost can be reduced using a basic form of extrapolation (Bickel and Yahav (1988), Brezinski and Zaglia (2013), Sidi (2003)).

Saving on computation with extrapolation. To explain the technique of extrapolation, the first step produces an inexpensive estimate $\widehat{\sigma}_{t_0}$ by applying Algorithm 2 to a small initial ensemble of size t_0. The second step then rescales $\widehat{\sigma}_{t_0}$ so that it approximates σ_t for $t \gg t_0$. This rescaling relies on Theorem 3.1, which leads to the approximation, $\sigma_t \approx \frac{\sigma}{\sqrt{t}}$. Consequently, we define the extrapolated estimate of σ_t as

$$\widehat{\sigma}_{t,\mathrm{extrap}} := \frac{\sqrt{t_0}}{\sqrt{t}} \cdot \widehat{\sigma}_{t_0}. \tag{4.2}$$

In turn, if the user desires $3\sigma_t \leq \epsilon$ for some $\epsilon \in (0, 1)$, then t should be chosen so that

$$3\widehat{\sigma}_{t,\mathrm{extrap}} \leq \epsilon, \tag{4.3}$$

which is equivalent to $t \geq (\frac{3\sqrt{t_0}}{\epsilon} \cdot \widehat{\sigma}_{t_0})^2$.

In addition to applying Algorithm 2 to a smaller ensemble, a second computational benefit is that extrapolation allows the user to "look ahead" and dynamically determine how much extra computation is needed so that σ_t is within a desired range. In Section 5, some examples are given showing that σ_{1000} can be estimated well via extrapolation when $t_0 = 200$.

Comparison with the cost of training a random forest. Given that one of the main uses of Algorithm 2 is to control the size of a random forest, one would hope that the cost of Algorithm 2 is less than or similar to the cost of training a

single ensemble. In order to simplify this comparison, suppose that each tree in the ensemble is grown so that all nodes are split into exactly 2 child nodes (except for the leaves), and that all trees are grown to a common depth $d \geq 2$. Furthermore, suppose that $\mathcal{X} \subset \mathbb{R}^p$, and that random forests uses the default rule of randomly selecting from $\lceil \sqrt{p} \rceil$ features during each node split. Under these conditions, it is known that the cost of training a random forest with t trees via CART is at least of order $t \cdot \sqrt{p} \cdot d \cdot n$ (Breiman et al. (1984), page 166). Additional background on the details of random forests and decision trees may be found in the book of Hastie, Tibshirani and Friedman (2001).

Based on the reasoning just given, the cost of running Algorithm 2 does not exceed the cost of training t trees, provided that

$$B = \mathcal{O}\left(\frac{t}{t_0} \cdot \sqrt{p} \cdot d\right),$$

where the factor $\frac{t}{t_0}$ arises from the extrapolation speedup described earlier. Moreover, with regard to the selection of B, our numerical examples in Section 5 show that the modest choice $B = 50$ allows Algorithm 2 to perform well on a variety of datasets.

The choice of the threshold ϵ. When using a criterion such as (4.3) in practice, the choice of the threshold ϵ will typically be unique to the user's goals. For instance, if the user desires that ERR_t is within 0.5% of err_∞, then the choice $\epsilon = 0.005$ would be appropriate. Another option is to choose ϵ from a relative standpoint, depending on the scale of the error. If the error is high (say $\mathbb{E}[\mathrm{ERR}_t|\mathcal{D}]$ is 40%), then it may not be worth paying a large computational price to ensure that $3\sigma_t$ is less than 0.5%. Conversely, if $\mathbb{E}[\mathrm{ERR}_t|\mathcal{D}]$ is 2%, then it may be worthwhile to train a very large ensemble so that σ_t is a fraction of 2%. In either of these cases, the size of σ_t could be controlled in a relative sense by selecting t when $\widehat{\sigma}_t \leq \eta \widehat{m}_t$, where $\widehat{m}_t$ is an estimate of $\mathbb{E}[\mathrm{ERR}_t|\mathcal{D}]$ obtained from a hold-out set, and $\eta \in (0, 1)$ is a user-specified constant that measures the balance between computational cost and accuracy. But regardless of the user's preference for ϵ, the more basic point to keep in mind is that the proposed method makes it possible for the user to have direct control over the relationship between t and σ_t, and this type of control has not previously been available.

5. Numerical experiments. To illustrate our proposed method, we describe experiments in which the random forests method is applied to natural and synthetic datasets (6 in total). More specifically, we consider the task of estimating the parameter $3\sigma_t = 3\sqrt{\mathrm{var}(\mathrm{ERR}_t|\mathcal{D})}$, as well as $3\sigma_{t,l} = 3\sqrt{\mathrm{var}(\mathrm{ERR}_{t,l}|\mathcal{D})}$. Overall, the main purpose of the experiments is to show that the bootstrap can indeed produce accurate estimates of these parameters. A second purpose is to demonstrate the value of the extrapolation technique from Section 4.2.

5.1. *Design of experiments*. Each of the 6 datasets were partitioned in the following way. First, each dataset was evenly split into a training set $\mathcal{D}$ and a "ground truth" set $\mathcal{D}_{\text{ground}}$, with nearly matching class proportions in $\mathcal{D}$ and $\mathcal{D}_{\text{ground}}$. (The reason that a substantial portion of data was set aside for $\mathcal{D}_{\text{ground}}$ was to ensure that ground truth values of σ_t and $\sigma_{t,l}$ could be approximated using this set.) Next, a smaller set $\mathcal{D}_{\text{hold}} \subset \mathcal{D}_{\text{ground}}$ with cardinality satisfying $|\mathcal{D}_{\text{hold}}|/(|\mathcal{D}_{\text{hold}}| + |\mathcal{D}|) \approx 1/6$ was used as the hold-out set for implementing Algorithm 2. As before, the class proportions in $\mathcal{D}_{\text{hold}}$ and $\mathcal{D}$ were nearly matching. The smaller size of $\mathcal{D}_{\text{hold}}$ was chosen to illustrate the performance of the method when hold-out points are limited.

Ground truth values. After preparing $\mathcal{D}$, $\mathcal{D}_{\text{ground}}$, and $\mathcal{D}_{\text{hold}}$, a collection of 1000 ensembles was trained on $\mathcal{D}$ by repeatedly running the random forests method. Each ensemble contained a total of 1000 trees, trained under default settings from the package `randomForest` (Liaw and Wiener (2002)). Also, we tested each ensemble on $\mathcal{D}_{\text{ground}}$ to approximate a corresponding sample path of ERR_t (like the ones shown in Figure 1). Next, in order to obtain "ground truth" values for σ_t with $t = 1, \ldots, 1000$, we used the sample standard deviation of the 1000 sample paths at each t. (Ground truth values for each $\sigma_{t,l}$ were obtained analogously.)

Extrapolated estimates. With regard to our methodology, we applied the hold-out and OOB versions of Algorithm 2 to each of the ensembles—yielding 1000 realizations of each type of estimate of σ_t. In each case, the number of bootstrap replicates was set to $B = 50$, and we applied the extrapolation rule, starting from $t_0 = 200$. If we let $\widehat{\sigma}_{200,\text{H}}$ and $\widehat{\sigma}_{200,\text{O}}$ denote the initial hold-out and OOB estimators, then the corresponding extrapolated estimators for $t \geq 200$ are given by

$$\widehat{\sigma}_{t,\text{H},\text{extrap}} := \frac{\sqrt{200}}{\sqrt{t}}\widehat{\sigma}_{200,\text{H}} \quad \text{and} \quad \widehat{\sigma}_{t,\text{O},\text{extrap}} := \frac{\sqrt{200}}{\sqrt{t}}\widehat{\sigma}_{200,\text{O}}. \tag{5.1}$$

Next, as a benchmark, we considered an enhanced version of the hold-out estimator, for which the entire ground truth set $\mathcal{D}_{\text{ground}}$ was used in place of $\mathcal{D}_{\text{hold}}$. In other words, this benchmark reflects a situation where a much larger hold-out set is available, and it is referred to as the "ground estimate" in the plots. Its value based on $t_0 = 200$ is denoted $\widehat{\sigma}_{200,\text{G}}$, and for $t \geq 200$, we use

$$\widehat{\sigma}_{t,\text{G},\text{extrap}} := \frac{\sqrt{200}}{\sqrt{t}}\widehat{\sigma}_{200,\text{G}} \tag{5.2}$$

to refer to its extrapolated version. Lastly, classwise versions of all extrapolated estimators were computed in an analogous way.

5.2. *Description of datasets*. The following datasets were each partitioned into $\mathcal{D}$, $\mathcal{D}_{\text{hold}}$ and $\mathcal{D}_{\text{ground}}$, as described above.

Census income data. A set of census records for 48,842 people were collected with 14 socioeconomic features (continuous and discrete) (Lichman (2013)). Each record was labeled as 0 or 1, corresponding to low or high income. The proportions of the classes are approximately $(0.76, 0.24)$. As a preprocessing step, we excluded three features corresponding to work-class, occupation and native country, due to a high proportion of missing values.

Connect-4 data. The observations represent 67,557 board positions in the two-person game "connect-4" [Lichman (2013)]. For each position, a list of 42 categorical features are available, and each position is labeled as a draw $l = 0$, loss $l = 1$ or win $l = 2$ for the first player, with the class proportions being approximately $(0.10, 0.25, 0.65)$.

Nursery data. This dataset was prepared from a set of 12,960 applications for admission to a nursery school (Lichman (2013)). Each application was associated with a list of 8 (categorical) socioeconomic features. Originally, each application was labeled as one of five classes, but in order to achieve reasonable label balance, the last three categories were combined. This led to approximate class proportions $(1/3, 1/3, 1/3)$.

Online news data. A collection of 39,797 news articles from the website mashable.com were associated with 60 features (continuous and discrete). Each article was labeled based on the number of times it was shared: fewer than 1000 shares ($l = 0$), between 1000 and 5000 shares ($l = 1$), and greater than 5000 shares ($l = 2$), with approximate class proportions $(0.28, 0.59, 0.13)$.

Synthetic continuous data. Two classes of data points in $\mathbb{R}^{100}$, each of size 10,000, were obtained by drawing samples from the multivariate normal distributions $N(\boldsymbol{\mu}_0, \Sigma)$ and $N(\boldsymbol{\mu}_1, \Sigma)$ with a common covariance matrix Σ. The first mean vector was chosen to be $\boldsymbol{\mu}_0 = \mathbf{0} \in \mathbb{R}^{100}$, and the second mean vector was constructed to be a sparse vector in the following way. Specifically, we sampled 10 numbers $(i_1, \ldots, i_{10})$ without replacement from $\{1, \ldots, 100\}$, and the coordinates of $\boldsymbol{\mu}_1$ indexed by $(i_1, \ldots, i_{10})$ were set to the value .05 (with all other coordinates were set to 0). Letting $U \Lambda U^\top$ denote the spectral decomposition of Σ, we selected the matrix of eigenvectors U by sampling from the uniform (Haar) distribution on 100×100 orthogonal matrices. The eigenvalues were chosen as $\Lambda = \operatorname{diag}(\frac{1}{1^2}, \frac{1}{2^2}, \ldots, \frac{1}{100^2})$.

Synthetic discrete data. Two classes of data points in $\mathbb{R}^{100}$, each of size 10,000, were obtained by drawing samples from the discrete distributions Multinomial$(N_0, \boldsymbol{p}_0)$ and Multinomial$(N_1, \boldsymbol{p}_1)$, where N_l refers to the number of balls in 100 cells, and the cell probabilities are specified by $\boldsymbol{p}_l \in \mathbb{R}^{100}$. Specifically, we set $N_0 = N_1 = 100$, and $\boldsymbol{p}_0 = (\frac{1}{100}, \ldots, \frac{1}{100})$. The vector $\boldsymbol{p}_1$ was obtained

by perturbing $\boldsymbol{p}_0$ and then normalizing it. Namely, letting $\boldsymbol{z} \in \mathbb{R}^{100}$ be a vector of i.i.d. $N(0,1)$ variables, we defined the vector $\boldsymbol{p}_1 = |\boldsymbol{p}_0 + \frac{1}{300}\boldsymbol{z}|/\|\boldsymbol{p}_0 + \frac{1}{300}\boldsymbol{z}\|_1$, where $|\cdot|$ refers to coordinatewise absolute value.

5.3. *Numerical results.*

Interpreting the plots. For each dataset, we plot the ground truth value $3\sigma_t$ as a function of $t = 1, \ldots, 1000$, where the y-axis is expressed in units of %, so that a value $3\sigma_t = 0.01$ is marked as 1%. Alongside each curve for $3\sigma_t$, we plot the averages of $3\widehat{\sigma}_{t,\mathrm{O},\mathrm{extrap}}$ (green), $3\widehat{\sigma}_{t,\mathrm{H},\mathrm{extrap}}$ (purple) and $3\widehat{\sigma}_{t,\mathrm{G},\mathrm{extrap}}$ (orange) over their 1000 realizations, with error bars indicating the spread between the 10th and 90th percentiles of the estimates. Here, the error bars are only given to illustrate the variance of the estimates, conditionally on $\mathcal{D}$, and they are not proposed as confidence intervals for σ_t. (Indeed, our main focus is on the fluctuations of ERR_t, rather than the fluctuations of variance estimates.) Lastly, we plot results for the classwise parameters $\sigma_{t,l}$ in the same manner, but in order to keep the number of plots manageable, we only display the class l with the highest value of $3\sigma_{t,l}$ at $t = 1000$. This is reflected in the plots, since the values of $3\sigma_{t,l}$ for the chosen class l are generally larger than $3\sigma_t$.

Walking through an example (*Figure* 2). To explain the plots from the user's perspective, suppose the user trains an initial ensemble of $t_0 = 200$ classifiers with the 'census income' data. (The following considerations will apply in the same way to the other datasets in Figures 3–7.) At this stage, the user may compute either of the estimators $\widehat{\sigma}_{200,\mathrm{O}}$ or $\widehat{\sigma}_{200,\mathrm{H}}$. In turn, the user may follow the definitions (5.1) to plot the extrapolated estimators for all $t \geq 200$ at no additional cost. These curves

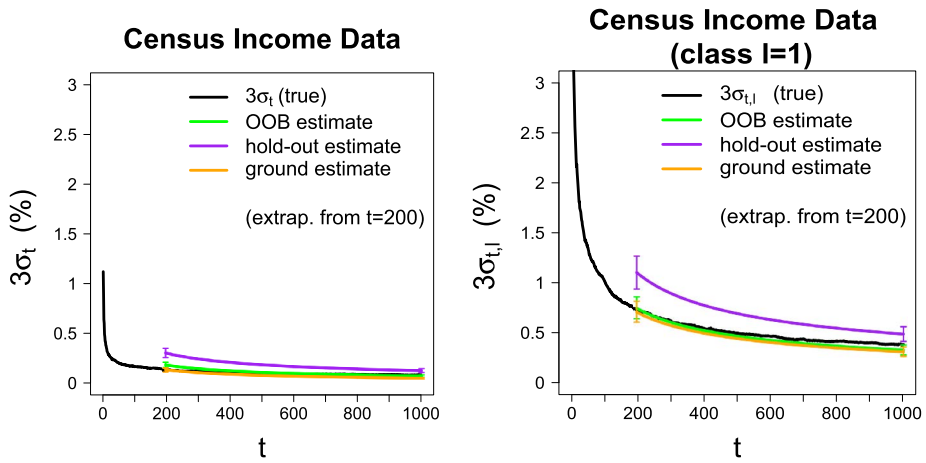

FIG. 2. *Results for census income data.*

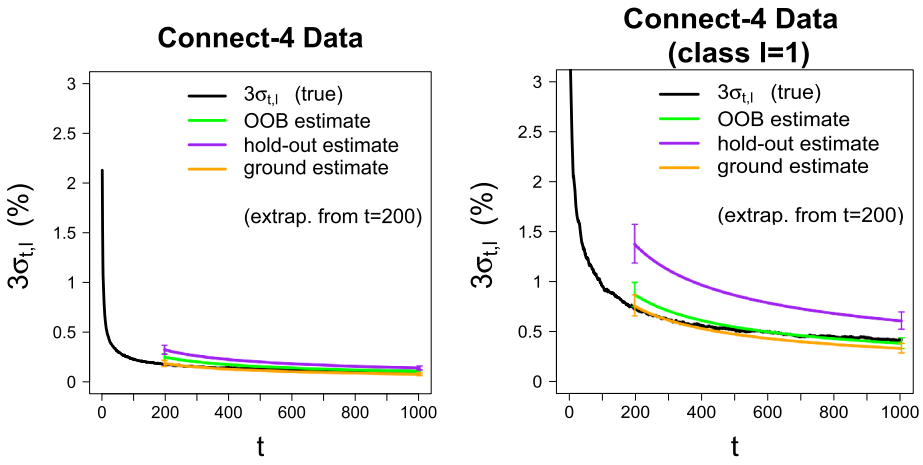

FIG. 3. *Results for connect-4 data.*

will look like the purple or green curves in the left panel of Figure 2, up to a small amount of variation indicated by the error bars.

If the user wants to select t so that $3\sigma_t$ is at most, say 0.5%, then the purple or green curves in the left panel of Figure 2 would tell the user that 200 classifiers are already sufficient, and no extra classifiers are needed (which is correct in this particular example). Alternatively, if the user happens to be interested in the class-wise error rate for $l = 1$, and if the user wants $3\sigma_{t,1}$ to be at most 0.5%, then the curve for the OOB estimator accurately predicts that approximately 600 total (i.e., 400 extra) classifiers are needed. By contrast, the hold-out method is conservative, and indicates that approximately 1000 total (i.e., 800 extra) classifiers should be

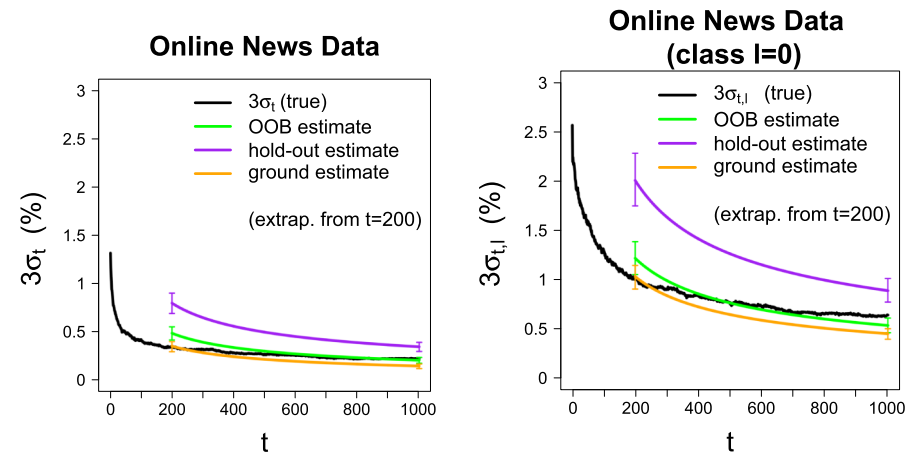

FIG. 4. *Results for online news data.*

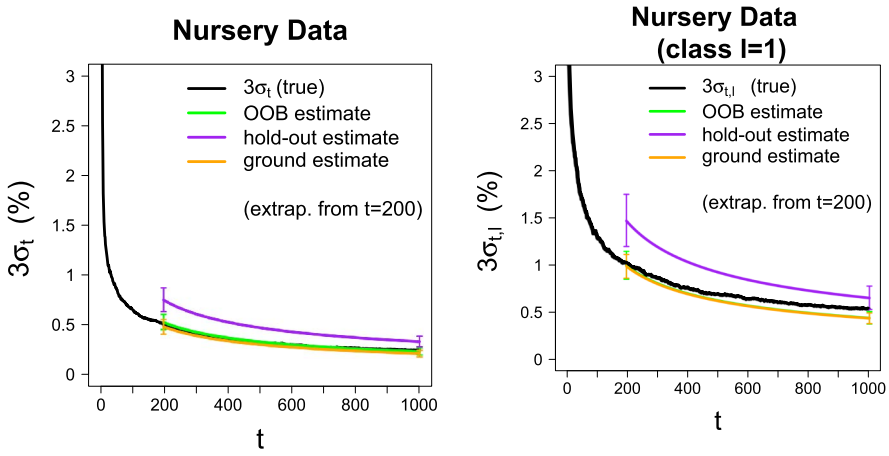

FIG. 5. *Results for nursery data.*

trained. So, in other words, the hold-out estimator would still provide the user with the desired outcome, but at a higher computational cost.

Comments on numerical results. Considering all of the datasets collectively, the plots show that the extrapolated OOB and ground estimators are generally quite accurate. Meanwhile, the hold-out estimator tends to be conservative, due to an upward bias. Consequently, the OOB method should be viewed as preferable, since it is both more accurate, and does not require data to be held out. Nevertheless, when considering the hold-out estimator, it is worth noting that the effect of the bias actually diminishes with extrapolation, and even if the initial value $3\widehat{\sigma}_{t_0,\mathrm{H},\mathrm{extrap}}$ has noticeable bias at $t_0 = 200$, it is possible for the extrapolated value $3\widehat{\sigma}_{t,\mathrm{H},\mathrm{extrap}}$ to have relatively small bias at $t = 1000$.

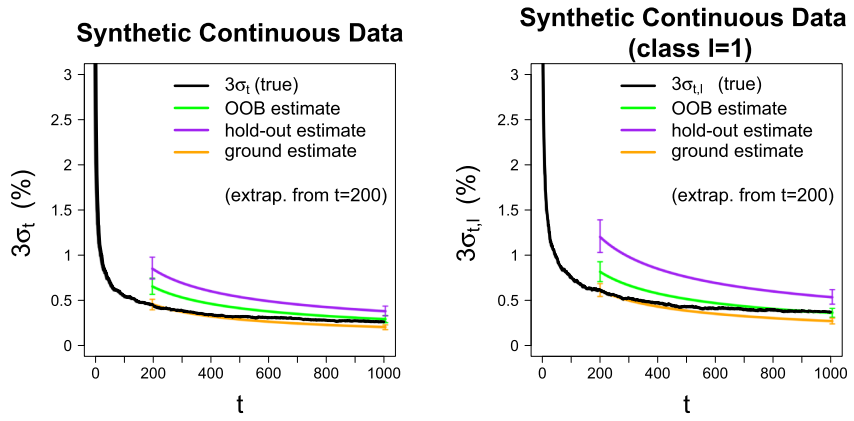

FIG. 6. *Results for synthetic continuous data.*

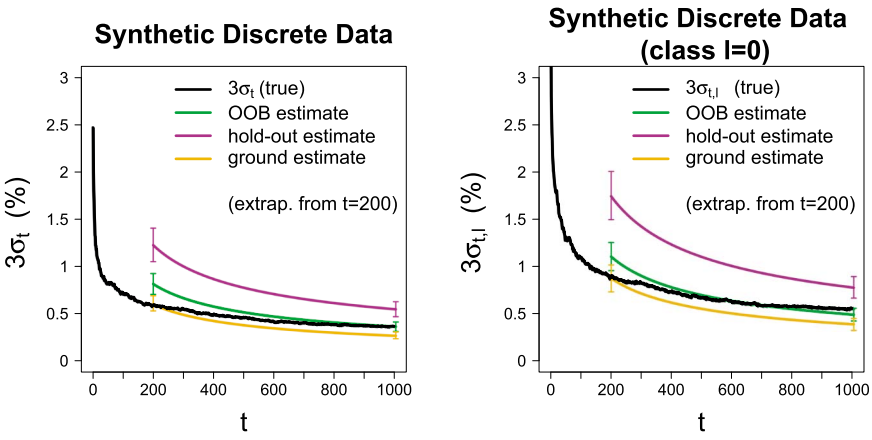

FIG. 7. *Results for synthetic discrete data.*

To understand where the bias of the hold-out estimator comes from, imagine that two ensembles have been trained on the same data, and suppose their accuracy is compared on a small hold-out set. In this situation, it is possible for their observed error rates on the hold-out set to noticeably differ—even if the true error rates are very close. For this reason, the small size of $\mathcal{D}_{\text{hold}}$ leads to greater variation among the estimated values $\widetilde{\text{ERR}}(\tilde{A}^*)$ generated in the hold-out version of Algorithm 2, which leads to an inflated estimate of σ_t. By contrast, the ground estimator suffers less from this bias because it relies on the much larger ground truth set $\mathcal{D}_{\text{ground}}$ in place of $\mathcal{D}_{\text{hold}}$. Similarly, the OOB estimate of σ_t is less susceptible to this bias, because it will typically use every point in the larger training set $\mathcal{D}$ as an "effective test point", preventing excess variation among the values $\widetilde{\text{ERR}}(A^*)$. (Even for small choices of t_0, all training points are likely to be an OOB point for at least one classifier.)

One last point to mention is that many of the datasets have discrete features, which may violate the theoretical conditions in Assumption 1. Nevertheless, the presence or absence of discrete features does not seem to substantially affect on the performance of the estimators. So, to this extent, the bootstrap does not seem to depend too heavily on Assumption 1. (See Appendix F.2 for further empirical assessment of that assumption.)

6. Conclusion. We have studied the notion of algorithmic variance $\sigma_t^2 = \text{var}(\text{ERR}_t|\mathcal{D})$ as a criterion for deciding when a randomized ensemble will perform nearly as well as an infinite one (trained on the same data). To estimate this parameter, we have developed a new bootstrap method, which allows the user to directly measure the convergence of randomized ensembles with a guarantee that has not previously been available.

With regard to practical considerations, we have shown that our bootstrap method can be enhanced in two ways. First, the use of a hold-out set can be avoided

with the OOB version of Algorithm 2, and our numerical results show that the OOB version is preferable when hold-out points are scarce. Second, the extrapolation technique substantially reduces the cost of bootstrapping. Furthermore, we have analyzed the cost of the method in terms of floating point operations to show that it compares favorably with the cost of training a single ensemble via random forests.

From a theoretical standpoint, we have analyzed the proposed method within the framework of a first-order model for randomized ensembles. In particular, for a generic ensemble whose classifiers are conditionally i.i.d. given $\mathcal{D}$, there is a corresponding first-order model that matches the generic ensemble with respect to its average error rate $\mathbb{E}[\mathrm{ERR}_t|\mathcal{D}]$. Under this setup, our main result shows that the proposed method consistently approximates $\mathcal{L}(\sqrt{t}(\mathrm{ERR}_t - \mathrm{err}_\infty)|\mathcal{D})$ as $t \to \infty$. Some extensions of this result could include generalizations to other models of randomized ensembles [e.g., along the lines of Biau (2012), Biau, Devroye and Lugosi (2008), Scornet (2016b), Scornet, Biau and Vert (2015)], as well as corresponding results in the context of regression ensembles, which we hope to pursue in future work. More generally, due to the versatility of bootstrap methods, our approach may also be relevant to measuring the convergence of other randomized algorithms, as in Lopes, Wang and Mahoney (2017, 2018).

Acknowledgments. The author thanks Peter Bickel, Philip Kegelmeyer and Debashis Paul for helpful discussions. In addition, the author thanks the Editors and referees for their valuable feedback, which significantly improved the paper.

SUPPLEMENTARY MATERIAL

Supplement: Supplementary Material for "Estimating the algorithmic variance of randomized ensembles via the bootstrap" (DOI: 10.1214/18-AOS1707SUPP; .pdf). The supplement contains proofs of all theoretical results, and an assessment of technical assumptions.

REFERENCES

ARLOT, S. and GENUER, R. (2014). Analysis of purely random forests bias. arXiv:1407.3939.

BIAU, G. (2012). Analysis of a random forests model. *J. Mach. Learn. Res.* **13** 1063–1095. MR2930634

BIAU, G., DEVROYE, L. and LUGOSI, G. (2008). Consistency of random forests and other averaging classifiers. *J. Mach. Learn. Res.* **9** 2015–2033. MR2447310

BICKEL, P. J. and YAHAV, J. A. (1988). Richardson extrapolation and the bootstrap. *J. Amer. Statist. Assoc.* **83** 387–393. MR0971364

BREIMAN, L. (1996). Bagging predictors. *Mach. Learn.* **24** 123–140.

BREIMAN, L. (2001). Random forests. *Mach. Learn.* **45** 5–32.

BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. A. and STONE, C. J. (1984). *Classification and Regression Trees*. Wadsworth Advanced Books and Software, Belmont, CA. MR0726392

BREZINSKI, C. and ZAGLIA, M. R. (2013). *Extrapolation Methods*: *Theory and Practice*. North-Holland, Amsterdam.

BÜHLMANN, P. and YU, B. (2002). Analyzing bagging. *Ann. Statist.* **30** 927–961. MR1926165

BUJA, A. and STUETZLE, W. (2000). Smoothing effects of bagging. Preprint, AT&T Labs-Research, Florham Park, NJ.

BUJA, A. and STUETZLE, W. (2006). Observations on bagging. *Statist. Sinica* **16** 323–351. MR2267238

BÜRGISSER, P. and CUCKER, F. (2013). *Condition*: *The Geometry of Numerical Algorithms*. *Grundlehren der Mathematischen Wissenschaften* [*Fundamental Principles of Mathematical Sciences*] **349**. Springer, Heidelberg. MR3098452

BYRD, R. H., CHIN, G. M., NOCEDAL, J. and WU, Y. (2012). Sample size selection in optimization methods for machine learning. *Math. Program.* **134** 127–155. MR2947555

CANNINGS, T. I. and SAMWORTH, R. J. (2017). Random-projection ensemble classification. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **79** 959–1035. With discussions and a reply by the authors. MR3689307

DIETTERICH, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach. Learn.* **40** 139–157.

EFRON, B. (2014). Estimation and accuracy after model selection. *J. Amer. Statist. Assoc.* **109** 991–1007. MR3265671

GENUER, R. (2012). Variance reduction in purely random forests. *J. Nonparametr. Stat.* **24** 543–562. MR2968888

HALL, P. and SAMWORTH, R. J. (2005). Properties of bagged nearest neighbour classifiers. *J. R. Stat. Soc. Ser. B. Stat. Methodol.* **67** 363–379. MR2155343

HASTIE, T., TIBSHIRANI, R. and FRIEDMAN, J. (2001). *The Elements of Statistical Learning*: *Data Mining*, *Inference*, *and Prediction*. Springer, New York. MR1851606

HERNÁNDEZ-LOBATO, D., MARTÍNEZ-MUÑOZ, G. and SUÁREZ, A. (2013). How large should ensembles of classifiers be? *Pattern Recognit.* **46** 1323–1336.

HO, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.* **20** 832–844.

KALLENBERG, O. (2006). *Foundations of Modern Probability*. Springer, Berlin.

LAM, L. and SUEN, C. Y. (1997). Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Trans. Syst. Man Cybern., Part A, Syst. Hum.* **27** 553–568.

LATINNE, P., DEBEIR, O. and DECAESTECKER, C. (2001). Limiting the number of trees in random forests. In *Multiple Classifier Systems* (*Cambridge*, 2001). *Lecture Notes in Computer Science* **2096** 178–187. Springer, Berlin. MR2043269

LIAW, A. and WIENER, M. (2002). Classification and regression by randomForest. *R News* **2** 18–22.

LICHMAN, M. (2013). UCI machine learning repository.

LIN, Y. and JEON, Y. (2006). Random forests and adaptive nearest neighbors. *J. Amer. Statist. Assoc.* **101** 578–590. MR2256176

LOPES, M. E. (2016). A sharp bound on the computation-accuracy tradeoff for majority voting ensembles. Preprint, arXiv:1303.0727.

LOPES, M. E. (2019). Supplement to "Estimating the algorithmic variance of randomized ensembles via the bootstrap." DOI:10.1214/18-AOS1707SUPP.

LOPES, M. E., WANG, S. and MAHONEY, M. W. (2017). A bootstrap method for error estimation in randomized matrix multiplication. Preprint, arXiv:1708.01945.

LOPES, M. E., WANG, S. and MAHONEY, M. W. (2018). Error estimation for randomized least-squares algorithms via the bootstrap. In *Proceedings of the* 35*th International Conference on Machine Learning* **80** 3217–3226.

MENTCH, L. and HOOKER, G. (2016). Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *J. Mach. Learn. Res.* **17** Paper No. 26, 41. MR3491120

NG, A. Y. and JORDAN, M. I. (2001). Convergence rates of the Voting Gibbs classifier, with application to Bayesian feature selection. In *Proceedings of the* 18*th International Conference on Machine Learning* 377–384.

OSHIRO, T. M., PEREZ, P. S. and BARANAUSKAS, J. A. (2012). How many trees in a random forest? In *Machine Learning and Data Mining in Pattern Recognition* 154–168. Springer, Berlin.

SCHAPIRE, R. E. and FREUND, Y. (2012). *Boosting*: *Foundations and Algorithms*. MIT Press, Cambridge, MA. MR2920188

SCORNET, E. (2016a). On the asymptotics of random forests. *J. Multivariate Anal.* **146** 72–83. MR3477650

SCORNET, E. (2016b). Random forests and kernel methods. *IEEE Trans. Inform. Theory* **62** 1485–1500. MR3472261

SCORNET, E., BIAU, G. and VERT, J.-P. (2015). Consistency of random forests. *Ann. Statist.* **43** 1716–1741. MR3357876

SEXTON, J. and LAAKE, P. (2009). Standard errors for bagged and random forest estimators. *Comput. Statist. Data Anal.* **53** 801–811. MR2654590

SIDI, A. (2003). *Practical Extrapolation Methods*: *Theory and Applications*. *Cambridge Monographs on Applied and Computational Mathematics* **10**. Cambridge Univ. Press, Cambridge. MR1994507

SIMON, L. (1983). *Lectures on Geometric Measure Theory*. *Proceedings of the Centre for Mathematical Analysis*, *Australian National University* **3**. Australian National Univ., Centre for Mathematical Analysis, Canberra. MR0756417

VAN DER VAART, A. W. and WELLNER, J. A. (1996). *Weak Convergence and Empirical Processes*: *With Applications to Statistics*. Springer, New York. MR1385671

WAGER, S., HASTIE, T. and EFRON, B. (2014). Confidence intervals for random forests: The jackknife and the infinitesimal jackknife. *J. Mach. Learn. Res.* **15** 1625–1651. MR3225243

WHITE, B. (2016). Introduction to minimal surface theory. In *Geometric Analysis*. *IAS/Park City Math. Ser.* **22** 387–438. Amer. Math. Soc., Providence, RI. MR3524221

DEPARTMENT OF STATISTICS
UNIVERSITY OF CALIFORNIA, DAVIS
MATHEMATICAL SCIENCES BUILDING 4118
399 CROCKER LANE
ONE SHIELDS AVENUE
DAVIS, CALIFORNIA 95616
USA
E-MAIL: melopes@ucdavis.edu