The role of algorithmic stability in distribution-free inference

Rina Foygel Barber*

October 3, 2025

Abstract. In statistics and machine learning, a model fitting algorithm is said to be stable if its output is not substantially altered by randomly resampling or removing a small number of data points from the training set. Algorithmic stability plays a crucial role in many results in statistics, particularly in the assumption-lean or distribution-free regime, which aims to avoid placing assumptions on the distribution of the data. A related question of interest is whether it is possible to certify that an algorithm is stable, or to construct an algorithm that is guaranteed to be stable, again without distributional assumptions. This article presents some results that characterize the benefits and challenges of the algorithmic stability framework in a distribution-free setting.

1 Introduction In many modern applications, machine learning algorithms are trained on high-dimensional data and return highly complex models, where both the distribution of the data, and the exact nature of the training process, are not known or not fully understood. These practical settings often lie far beyond the scope of classical statistical methods (such as parametric models), which leads to substantial challenges in performing inference: for instance, can we quantify our confidence in the fitted model's predictions, in the estimates of effect sizes, or in the results of tests that aim to detect the presence or absence of some effect or some property of the data?

In this article, our focus is primarily on the problem of predictive inference—quantifying our confidence, or our uncertainty, in a fitted model's predictions. To make this questions more concrete, suppose that we have trained a model $\hat{f}$ for predicting a response $Y \in \mathcal{Y}$ based on features $X \in \mathcal{X}$. Can we determine our confidence in the prediction $\hat{f}(X)$ on a new test point X—for instance, by computing a margin of error around $\hat{f}(X)$ to provide a prediction interval, or, by estimating some measure of risk or error of the predictions produced by $\hat{f}$? If we wish to avoid placing assumptions on the data distribution, a standard approach to such questions is to use a holdout set: we use a portion of the available data to train the model $\hat{f}$, and use the remaining data (the holdout set) to evaluate its performance. However, this approach incurs a statistical cost, due to splitting the data, since the sample size used for training the model must necessarily be substantially smaller in order to leave sufficient data for the holdout set.

We may therefore prefer cross-validation type approaches, to make more efficient use of the limited available data. For instance, in 10-fold cross validation, after partitioning the training data into 10 (equal-sized) batches, we train models $\hat{f}^{(k)}$ on the 90% of the data that remains after deleting batch k, for each $k = 1, \ldots, 10$. We then examine the performance of each $\hat{f}^{(k)}$ on its own held-out batch (i.e., the performance of $\hat{f}^{(k)}(X_i)$ for predicting Y_i, for each data point (X_i, Y_i) that is in the kth batch, and therefore, was *not* used in the training process for $\hat{f}^{(k)}$), and based on these errors, we can attempt to provide prediction intervals or estimates of model accuracy for $\hat{f}$, the model trained on the entire data set. While this approach is intuitively reasonable and empirically often performs well, its theoretical validity generally relies on certain regularity conditions, and may only yield asymptotic guarantees—which is unsatisfying if we wish to provide inference whose validity holds at a finite sample size n, and does not rely on untestable assumptions.

The framework of *algorithmic stability* provides an appealing alternative for analyzing this problem. A model training algorithm is said to be stable if the predictions returned by a fitted model $\hat{f}$, which is trained on a large data set $(X_1, Y_1), \ldots, (X_n, Y_n)$, remain approximately the same under small random perturbations of the training data set—specifically, if we randomly delete or resample one, or a small number, of the n data points. In this article, we will explore several results at the intersection of algorithmic stability and distribution-free inference:

*Department of Statistics, University of Chicago (rina@uchicago.edu).

- First, in Section 2, we will see that the assumption of algorithmic stability is sufficient to ensure nonasymptotic guarantees for predictive inference via cross-validation, without any additional assumptions on the distribution of the data.
- Since stability appears to hold for many algorithms used in practice, this suggests that algorithmic stability may provide a route for obtaining assumption-free predictive inference guarantees. But, perhaps surprisingly, a hardness result presented in Section 3 shows that in the "black-box" setting, where the complexity of the training algorithm forces us to study its properties empirically rather than theoretically, the problem of certifying an algorithm as stable is impossible without assumptions.
- Motivated by the hardness of testing whether an existing algorithm is stable, in Section 4 we show that by using bagging, where we average multiple fitted models produced by running an algorithm on multiple subsets resampled from the data, algorithmic stability can be guaranteed to hold without any assumptions on either the algorithm or the data.
- Finally, in Section 5, we combine these results to provide a fully assumption-free approach towards predictive inference via cross-validation, and discuss additional connections and open questions raised by this work.

The results presented in this article are based on joint work in collaboration with Emmanuel Candès, Aaditya Ramdas, and Ryan Tibshirani [7], for the results in Section 2; with Byol Kim [27] and with Yuetian Luo [38], for the results in Section 3; and with Jake Soloff and Rebecca Willett [49, 50], for the results in Section 4.

1.1 Defining algorithmic stability Throughout this article, we will assume that the response Y is real-valued, $\mathcal{Y} \subseteq \mathbb{R}$. We will use the notation $\mathcal{A}$ to denote a model training algorithm, which inputs a data set of any size (that is, any number of (X, Y) pairs), and returns a fitted model $\hat{f} : \mathcal{X} \to \mathbb{R}$ that predicts Y from X—for instance, linear regression, or a neural network. More formally, $\mathcal{A}$ is a map from $\bigcup_{n\geq 0}(\mathcal{X} \times \mathbb{R})^n$ (i.e., data sets of any size) to the space of measurable functions $\mathcal{X} \to \mathbb{R}$, with

$$\hat{f} = \mathcal{A}\big((X_1, Y_1), \ldots, (X_n, Y_n)\big)$$

denoting the fitted model $\hat{f} : \mathcal{X} \to \mathbb{R}$ produced by running the training algorithm $\mathcal{A}$ on the data set $(X_1, Y_1), \ldots, (X_n, Y_n)$. We will often assume that $\mathcal{A}$ is *symmetric*, meaning that $\mathcal{A}$ is invariant to the order of the data points on which it is trained—that is, $\mathcal{A}((X_1, Y_1), \ldots, (X_n, Y_n)) = \mathcal{A}((X_{\sigma(1)}, Y_{\sigma(1)}), \ldots, (X_{\sigma(n)}, Y_{\sigma(n)}))$, for any data set and any permutation σ on $[n] := \{1, \ldots, n\}$. Some algorithms, such as stochastic gradient descent methods, are randomized, meaning that a training data set $(X_1, Y_1), \ldots, (X_n, Y_n)$ can return different fitted models $\hat{f}$ on different calls to $\mathcal{A}$. We can express this by adding an additional argument, a random seed $\xi \in [0, 1]$, to the input of $\mathcal{A}$, so that the fitted model is now given by $\hat{f} = \mathcal{A}((X_1, Y_1), \ldots, (X_n, Y_n); \xi)$. (A symmetric algorithm is now defined in a distributional sense: essentially, $\mathcal{A}$ is symmetric if the distribution of its predictions is unchanged when we permute the training data points.) The results presented in this paper apply regardless of whether $\mathcal{A}$ is deterministic or randomized, but for simplicity we generally will use the notation of a deterministic algorithm $\mathcal{A}$.

Intuitively, we say that an algorithm $\mathcal{A}$ is stable if a small change in the training data set generally does not lead to large changes in its output (the fitted model $\hat{f}$). However, there are many ways to formalize this intuition: how should we quantify the amount of "change" in the input data, and in the output model? A wide range of varying definitions appear in the literature, many of which are qualitatively very different from each other; see [12] for an overview of some of the most important formulations of stability.

In this work, our primary focus will be the following definition:

DEFINITION 1.1. *We say that an algorithm $\mathcal{A}$ satisfies (ε, δ)-stability with respect to a distribution P on $\mathcal{X} \times \mathbb{R}$ and a sample size $n \geq 1$ (or, equivalently, we say that $(\mathcal{A}, P, n)$ is (ε, δ)-stable) if*

$$\mathbb{P}\left(|\hat{f}(X) - \hat{f}_{-i}(X)| > \varepsilon\right) \leq \delta,$$

for all $i \in [n]$, where $(X_1, Y_1), \ldots, (X_n, Y_n), (X, Y) \overset{\text{iid}}{\sim} P$, $\hat{f} = \mathcal{A}((X_j, Y_j)_{j\in[n]})$, and $\hat{f}_{-i} = \mathcal{A}((X_j, Y_j)_{j\in[n]\setminus\{i\}})$.

This type of stability condition appears in many classical works in the algorithmic stability literature, e.g., [26, 12], and is sometimes called "hypothesis stability". A related definition, which is quite similar in flavor, instead uses an ℓ_2 type measure of the perturbation:

DEFINITION 1.2. *We say that an algorithm $\mathcal{A}$ satisfies β-ℓ_2-stability with respect to a distribution P on $\mathcal{X} \times \mathbb{R}$ and a sample size $n \geq 1$ (or, equivalently, we say that $(\mathcal{A}, P, n)$ is β-ℓ_2-stable) if*

$$\mathbb{E}\left[(\hat{f}(X) - \hat{f}_{-i}(X))^2\right] \leq \beta^2,$$

for all $i \in [n]$, where $(X_1, Y_1), \ldots, (X_n, Y_n), (X, Y) \overset{\text{iid}}{\sim} P$, $\hat{f} = \mathcal{A}((X_j, Y_j)_{j \in [n]})$, and $\hat{f}_{-i} = \mathcal{A}((X_j, Y_j)_{j \in [n] \setminus \{i\}})$.

These two definitions are closely related: indeed, if $(\mathcal{A}, P, n)$ is β-ℓ_2-stable then it is also $(\varepsilon, (\beta/\varepsilon)^2)$-stable for any $\varepsilon > 0$ (by Markov's inequality), and conversely if $(\mathcal{A}, P, n)$ is (ε, δ)-stable, and the fitted models produced by $\mathcal{A}$ always return predictions lying in some bounded range $[a, b]$, then $(\mathcal{A}, P, n)$ is $\sqrt{\varepsilon^2 + (b-a)^2\delta}$-$\ell_2$-stable. (In the case of a randomized algorithm $\mathcal{A}$, these definitions should be interpreted as computing probability or expectation with respect to the random draw of both the data points (X_i, Y_i) and the random seed ξ [20].) While the definitions here examine the change in the prediction when a single data point is deleted, other similar definitions in the literature instead consider resampling (rather than removing) the ith data point.

In a certain sense, both of the above definitions describe a fairly weak notion of stability. They require that, for i.i.d. data, deleting one training point (X_i, Y_i) from the training set is unlikely to lead to a large change in the prediction at a new test point $X \in \mathcal{X}$. Since the data points are assumed to be i.i.d., effectively we are deleting one data point *at random* from the training set of size n. Indeed, it may be the case that, in a training set of size n, a few of the data points are highly influential—but if we delete one data point at random, it is unlikely that we have deleted one of the most influential points.

At the other extreme, we might consider a uniform, or worst-case, notion of stability (stated here only for the case of a deterministic $\mathcal{A}$, for simplicity):

DEFINITION 1.3. *We say that an algorithm $\mathcal{A}$ satisfies β-uniform-stability with respect to a sample size $n \geq 1$ (or, equivalently, we say that $(\mathcal{A}, n)$ is β-uniformly-stable) if*

$$|\hat{f}(X) - \hat{f}_{-i}(X)| \leq \beta$$

for all $i \in [n]$ and all data values $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathcal{X} \times \mathbb{R}$ and $X \in \mathcal{X}$, where $\hat{f} = \mathcal{A}((X_j, Y_j)_{j \in [n]})$ and $\hat{f}_{-i} = \mathcal{A}((X_j, Y_j)_{j \in [n] \setminus \{i\}})$.

In contrast to the earlier definitions, here we require that $\hat{f}$ is essentially unchanged when a data set is perturbed by adding or removing *any* data point. Under this stronger notion of stability, then, no one training point is allowed to be strongly influential. Definition 1.3 is sometimes referred to as the *sensitivity* (specifically, the sensitivity of the function $(X_i, Y_i)_{i \in [n]} \mapsto \hat{f}(X)$, mapping a training data set to a prediction at the test point), and plays a crucial role in the construction of differentially private statistical methods [18].

1.2 Algorithmic stability in statistics and learning theory Algorithmic stability has been extensively studied over the past 50 years in the statistics and learning theory literature. A primary focus of much of the work in this area has been the connection between algorithmic stability and generalization—see, e.g., [29, 12, 21, 43, 39, 47]. At a high level, these works show that uniform stability type conditions (as in Definition 1.3) ensure that generalization holds in a strong sense, meaning that a fitted model's training error is a good estimate of its test performance, but weaker notions of stability (as in Definitions 1.1 or 1.2) are nonetheless sufficient to ensure that cross-validation type estimates of test error are accurate. Moreover, stability is closely connected to learnability, i.e., whether there exists any algorithm that can produce a fitted model whose risk is asymptotically optimal as $n \to \infty$ [47]; see also [56] for connections between stability, learnability, and privacy. More recent works [25, 6, 9, 8] establish generalization guarantees and central limit theorem results for estimating model risk via cross-validation, under stability assumptions. See [32] for an overview of many recent results in these areas.

In light of these important downstream implications of stability, we may wish to ensure that the machine learning algorithms we use are indeed stable. But, non-asymptotically, the algorithms for which stability has been theoretically shown to hold are fairly limited: for instance, k-nearest neighbors and other "k-local" methods [45, 17], ridge regression and other methods with strongly convex regularization [12, 57], and stochastic gradient descent type methods under certain smoothness conditions [23]. Beyond such simple examples, many algorithms are too complex to permit a precise theoretical analysis of their stability properties. Indeed, in some cases, certain types of algorithms are known to be unavoidably unstable—for instance, [58] show that any algorithm

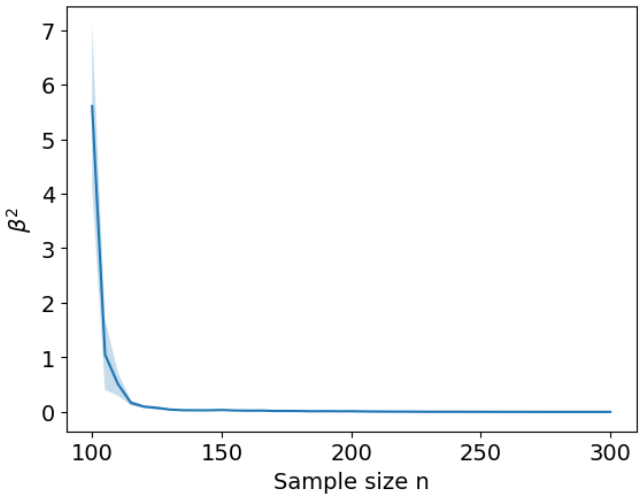

Figure 1.1: This figure illustrates an empirical estimate of the parameter β^2, for least squares regression to satisfy β-ℓ_2-stability (as in Definition 1.2). The data points are i.i.d., with $X_i \sim \mathcal{N}(0, \mathbf{I}_d)$ (for dimension $d = 95$), and $Y_i = X_i^\top \theta + \mathcal{N}(0, 1)$, where $\theta = \sqrt{10} \cdot u$ for a uniformly drawn unit vector $u \in \mathbb{R}^d$. For each sample size n, the figure plots $(\hat{f}(X) - \hat{f}_{-i}(X))^2$ (with notation defined as in Definition 1.2), showing the empirical mean over 50 independent trials (with shaded area representing the standard error). We see high instability when $n \approx d$.

that constrained to obey sparsity cannot be stable. Even classical algorithms such as least squares may exhibit severe instability, as illustrated in Figure 1.1. Nonetheless, many algorithms in common use do appear to exhibit stability properties empirically, and stability is widely considered to be less stringent assumption as compared to assuming restrictive properties of the data distribution (such as a parametric model or a smoothness condition).

2 Predictive inference via cross-validation: guarantees under stability In this section, we consider the problem of predictive inference: given a trained model $\hat{f}$, how can we quantify our uncertainty around the predictions returned by this model, for future test points? If the training and test data are i.i.d. from a common distribution, but the properties of this distribution are completely unknown, we cannot rely on classical methods such as parametric models to provide valid predictive intervals. We will see that, when constructing a prediction interval for data drawn from an unknown distribution, using a stable algorithm for training the model $\hat{f}$ allows for coverage guarantees without any assumptions on the distribution of the data.

2.1 The problem of distribution-free predictive inference We begin by defining some notation to formalize the problem. Let $(X_1, Y_1), \ldots, (X_n, Y_n), (X_{n+1}, Y_{n+1}) \overset{\text{iid}}{\sim} P$, for a distribution P on $\mathcal{X} \times \mathbb{R}$. Here data points $i = 1, \ldots, n$ comprise the available training data, while (X_{n+1}, Y_{n+1}) denotes a test point, for which the feature X_{n+1} is observed while Y_{n+1} is unobserved (and our task is to provide predictive inference for this unseen response). We emphasize that this setting is distribution-free in the sense that we are unwilling to place any assumptions on P; nonetheless, we do assume that the data points are i.i.d., meaning that we are not allowing for challenges such as distribution shift between the training and test data.

Writing $\mathcal{A}$ to denote a model training algorithm as before, let

$$\hat{f} = \mathcal{A}\big((X_1, Y_1), \ldots, (X_n, Y_n)\big) \tag{2.1}$$

be the fitted model produced by using all n available data points as the training set. Given the test feature X_{n+1}, we would like to construct a prediction interval around the prediction $\hat{f}(X_{n+1})$,

$$\mathcal{C}(X_{n+1}) = \Big[\hat{f}(X_{n+1}) - q, \hat{f}(X_{n+1}) + q\Big]$$

for some margin of error q, in such a way that ensures coverage with some desired probability $1 - \alpha$, i.e., we would like to guarantee that

$$\mathbb{P}(Y_{n+1} \in \mathcal{C}(X_{n+1})) \geq 1 - \alpha \text{ for any distribution } P. \tag{2.2}$$

How can we compute a margin of error q such that this coverage probability is guaranteed to hold, without placing any assumptions on P?

In general, using the errors on the training data—that is, the residuals $|Y_i - \hat{f}(X_i)|$, for $i \in [n]$—cannot provide an accurate estimate of the likely size of the test point's residual, $|Y_{n+1} - \hat{f}(X_{n+1})|$, for all but the simplest algorithms. This is because high-dimensional models often overfit to the training data, meaning that if we choose a margin of error based on the empirical distribution of the training errors, we will likely produce a prediction interval that is too narrow to achieve the desired coverage level. Instead, it has long been common to use a holdout set: partitioning the sample into two portions, $n = n_0 + n_1$, we train the model on the first n_0 data points, $\hat{f}_0 = \mathcal{A}((X_1, Y_1), \ldots, (X_{n_0}, Y_{n_0}))$, and then use the remaining data $\{(X_i, Y_i)\}_{i=n_0+1,\ldots,n}$ as a holdout set to estimate the test error distribution of our model.

In the distribution-free prediction literature, this type of approach is known as *split conformal prediction* (also known as inductive conformal prediction): choosing $\hat{q}_1 = \text{Quantile}_{(1-\alpha)(1+1/n_1)}(|Y_{n_0+1} - \hat{f}_0(X_{n_0+1})|, \ldots, |Y_n - \hat{f}_0(X_n)|)$ as the appropriate quantile of the residuals on the test point, the resulting prediction set $\mathcal{C}_{\text{split}}(X_{n+1}) = [\hat{f}_0(X_{n+1}) - \hat{q}_1, \hat{f}_0(X_{n+1}) + \hat{q}_1]$ offers a distribution-free coverage guarantee as in (2.2) [42, 54, 33]. While this is an appealing guarantee, unfortunately split conformal prediction (and any other holdout set based method) comes with a statistical cost: the model $\hat{f}_0$ is trained on only a portion of the available n data points, and therefore is often less accurate than $\hat{f}$, and necessitates a wider prediction interval. An alternative method is *full conformal prediction* [54], which avoids the data splitting step (and thus often returns more precise prediction intervals), but comes at a steep computational cost: it requires refitting the model for each potential value of the test point, namely, computing $\mathcal{A}((X_1, Y_1), \ldots, (X_n, Y_n), (X_{n+1}, y))$ for every possible value y for Y_{n+1}. In some settings this is straightforward—for instance, if $\mathcal{Y} = \{0, 1\}$, for a binary response. But for a real-valued response, this is computationally impossible and can only be approximated via discretization [15], aside from certain special cases such as ridge regression [41], the Lasso [31], or algorithms satisfying uniform stability type conditions [40]. See [54, 46, 5, 4] for further background on full conformal prediction and related methods.

2.2 Constructing a prediction interval via cross-validation If we do not want to incur the statistical cost of data splitting, but also cannot afford the computational burden of full conformal prediction, an intermediate option is to use cross-validation in order to construct the prediction interval. Cross-validation type methods date back many decades in the statistical literature [52, 19]. Here, we will focus on leave-one-out cross-validation: in order to define a margin of error for achieving coverage at the desired level $1 - \alpha$, we compute the leave-one-out residuals for each of the n data points, and take the $(1 - \alpha)$-quantile of these residuals:

$$\begin{aligned} \hat{q} = \text{Quantile}_{1-\alpha}\big(|Y_1 - \hat{f}_{-1}(X_1)|, \ldots, |Y_n - \hat{f}_{-n}(X_n)|\big) \\ \text{where } \hat{f}_{-i} = \mathcal{A}\big((X_1, Y_1), \ldots, (X_{i-1}, Y_{i-1}), (X_{i+1}, Y_{i+1}), \ldots, (X_n, Y_n)\big). \end{aligned} \tag{2.3}$$

The resulting leave-one-out CV prediction interval is then given by

$$\mathcal{C}_{\text{CV}}(X_{n+1}) = \left[\hat{f}(X_{n+1}) - \hat{q}, \hat{f}(X_{n+1}) + \hat{q}\right].$$

This approach requires refitting the model n many times (that is, n many calls to the algorithm $\mathcal{A}$), which is a compromise between the holdout set approach (which requires only one call to $\mathcal{A}$, but at the statistical cost of a much less accurate model), and the full conformal framework (which requires, potentially, infinitely many calls to $\mathcal{A}$, but offers greater statistical efficiency). Of course, in practice, we may want to further reduce the computational cost by using K-fold cross-validation, rather than leave-one-out (i.e., n-fold) cross-validation, but we do not consider this extension here. This leave-one-out construction is sometimes referred to as the *jackknife* interval in the predictive inference literature.

Intuitively, we would expect this method to be very effective in practice: the leave-one-out residuals $|Y_i - \hat{f}_{-i}(X_i)|$ should provide a good estimate of the predictive accuracy of the fitted model $\hat{f}$ on new test points, since the ith data point (X_i, Y_i) was not used for training $\hat{f}_{-i}$ and is therefore effectively an independent test point. But as the following example shows, without any further assumptions it is not possible to provide a distribution-free guarantee of coverage.

EXAMPLE 2.1. *Suppose $\mathcal{A}$ is the following model fitting algorithm: given a training set with m data points, $\mathcal{A}$ returns a constant function that always predicts the value m. That is, for any $m \geq 1$ and any data points*

$(X_1, Y_1), \ldots, (X_m, Y_m)$, the fitted model $\mathcal{A}((X_1, Y_1), \ldots, (X_m, Y_m))$ is given by the constant function $x \mapsto m$. Now let P be a distribution on $\mathcal{X} \times \mathbb{R}$ such that $Y = 0$ holds almost surely under P. Then, almost surely, we will have leave-one-out residuals $|Y_i - \hat{f}_{-i}(X_i)| = |0 - (n-1)| = n-1$, for each $i \in [n]$, and therefore the prediction interval will be given by $\mathcal{C}_{\mathrm{CV}}(X_{n+1}) = [\hat{f}(X_{n+1}) - \hat{q}, \hat{f}(X_{n+1}) + \hat{q}] = [n-(n-1), n+(n-1)] = [1, 2n-1]$. Since $Y_{n+1} = 0$ almost surely, we will therefore have zero coverage, $\mathbb{P}(Y_{n+1} \in \mathcal{C}_{\mathrm{CV}}(X_{n+1})) = 0$.

Of course, in this example, the algorithm $\mathcal{A}$ is highly degenerate, and does not resemble any procedure we might ever consider using in practice. But this issue is not merely a theoretical one, arising in degenerate examples; for instance, [7] show that severe undercoverage may occur for least squares regression in dimension d, even when run on data generated from the linear model, if we are in the regime $n \approx d$.

2.3 A coverage guarantee As we have seen in Example 2.1 above, for a highly degenerate algorithm, it is possible for the CV prediction interval to have an arbitrarily high loss of coverage. The algorithm used in the construction is highly unstable, which suggests that a stability assumption might be sufficient for proving a coverage guarantee. The same conjecture is suggested by the example of least squares regression, which can also exhibit undercoverage when $n \approx d$ [7], since least squares in dimension d is known to be unstable when $n \approx d$ due to a large condition number of the covariate matrix (as illustrated in Figure 1.1).

The following result verifies that this is indeed the case: if the CV prediction interval is constructed using a stable algorithm $\mathcal{A}$, then, with a small inflation to the interval, predictive coverage is guaranteed to achieve nearly the nominal level $1 - \alpha$.

THEOREM 2.2 (Adapted from [7, Theorem 5]). *Let P be any distribution on $\mathcal{X} \times \mathbb{R}$, and let $(X_1, Y_1), \ldots, (X_n, Y_n), (X_{n+1}, Y_{n+1}) \overset{\mathrm{iid}}{\sim} P$. Let $\mathcal{A}$ be a symmetric algorithm, and assume that $\mathcal{A}$ satisfies (ε, δ)-stability with respect to distribution P at sample size n (Definition 1.1). Define the prediction interval*

$$\mathcal{C}^{\varepsilon}_{\mathrm{CV}}(X_{n+1}) = \left[\hat{f}(X_{n+1}) - (\hat{q} + \varepsilon), \hat{f}(X_{n+1}) + (\hat{q} + \varepsilon)\right],$$

where $\hat{f}$ and $\hat{q}$ are defined as in (2.1) and (2.3)—this is a slight inflation of the CV prediction interval. Then

$$\mathbb{P}\left(Y_{n+1} \in \mathcal{C}^{\varepsilon}_{\mathrm{CV}}(X_{n+1})\right) \geq (1-\alpha)\left(1 - \frac{1}{n+1}\right) - \sqrt{2\delta}.$$

Many related results appear in the literature as well, including asymptotic analyses of the coverage properties for CV predictive intervals under stability [51], and results that guarantee stronger forms of coverage (namely, coverage conditional on the training data) under stability conditions [34, 2].

This result tells us that, as long as the algorithm $\mathcal{A}$ can be assumed to be stable, the CV prediction interval should essentially obtain the desired coverage level. (The slight inflation of the interval required in the theorem can generally be ignored, since for a continuously distributed response, the probability that Y_{n+1} would be covered by $\mathcal{C}^{\varepsilon}_{\mathrm{CV}}(X_{n+1})$ but not by $\mathcal{C}_{\mathrm{CV}}(X_{n+1})$ would likely be small.) This is an encouraging finding, because it offers a distribution-free coverage guarantee that allows us to find a compromise between the statistical and computational costs and benefits of split conformal prediction and full conformal prediction. But unlike the coverage guarantees for the conformal prediction methods, here we rely on a stability assumption, even though the distribution P of the data may be arbitrary. (Some methods in the literature—cross-conformal prediction [53, 55] and the jackknife+ and CV+ methods [7]—remove this assumption by constructing the prediction interval in a different way, but here we will restrict our attention to the CV prediction interval.)

For the CV prediction interval, should we view Theorem 2.2 as a universal guarantee—or have we essentially replaced one type of unverifiable assumption (i.e., assumptions on P) with another (i.e., the stability assumption on $\mathcal{A}$)? In order to address this question, in the next section we will ask whether it is possible to empirically certify that an algorithm $\mathcal{A}$ is stable, in order to be confident that the coverage guarantees of Theorem 2.2 apply in a given setting.

2.4 Proof of Theorem 2.2 To complete this section, we present a proof of the coverage guarantee stated in Theorem 2.2. First, by definition of the prediction interval $\mathcal{C}^{\varepsilon}_{\mathrm{CV}}(X_{n+1})$, we have

$$Y_{n+1} \notin \mathcal{C}^{\varepsilon}_{\mathrm{CV}}(X_{n+1}) \iff |Y_{n+1} - \hat{f}(X_{n+1})| > \mathrm{Quantile}_{1-\alpha}\big((|Y_i - \hat{f}_{-i}(X_i)|)_{i \in [n]}\big) + \varepsilon. \tag{2.4}$$

Next we will consider models that are allowed to incorporate the test point (X_{n+1}, Y_{n+1}) into training. For each $i \in [n+1]$, define a model

$$\tilde{f}_{-i} = \mathcal{A}\big((X_j, Y_j)_{j \in [n+1] \setminus \{i\}}\big),$$

and note that in the case $i = n+1$, we have $\tilde{f}_{-(n+1)} = \hat{f}$, by construction. Moreover, for each $i \neq j \in [n+1]$, define a model

$$\tilde{f}_{-i,j} = \mathcal{A}\big((X_k, Y_k)_{k \in [n+1] \setminus \{i,j\}}\big),$$

and note that in the case $j = n+1$, we have $\tilde{f}_{-i,(n+1)} = \tilde{f}_{-(n+1),i} = \hat{f}_{-i}$, by construction. Next we define a matrix of residuals $R \in \mathbb{R}^{(n+1)\times(n+1)}$, with entries given by

$$R_{i,j} = \begin{cases} |Y_i - \tilde{f}_{-i,j}(X_i)|, & i \neq j, \\ |Y_i - \tilde{f}_{-i}(X_i)|, & i = j. \end{cases}$$

Under stability, we would expect $R_{i,j} \approx R_{i,i}$ for all $i \neq j$, since these values are each the residual of the same data point (X_i, Y_i), under trained models $\tilde{f}_{-i,j}$ and $\tilde{f}_{-i}$ (which should output similar predictions, due to stability).

Examining all these definitions, we can see that the equivalence in (2.4) can be expressed as

$$Y_{n+1} \notin \mathcal{C}^{\varepsilon}_{\mathrm{CV}}(X_{n+1}) \iff R_{n+1,n+1} > \mathrm{Quantile}_{1-\alpha}\big((R_{i,n+1})_{i\in[n]}\big) + \varepsilon. \tag{2.5}$$

By the assumption of the exchangeability of the data, along with the symmetry of the algorithm $\mathcal{A}$, the distribution of this matrix of residuals R is unchanged if we permute the order of the data points—that is, we can verify that

$$\Pi R \Pi^\top \overset{\mathrm{d}}{=} R$$

for any permutation matrix $\Pi \in \{0,1\}^{(n+1)\times(n+1)}$, where $\overset{\mathrm{d}}{=}$ denotes equality in distribution. Therefore,

$$\mathbb{P}\left(R_{n+1,n+1} > \mathrm{Quantile}_{1-\alpha}\big((R_{i,n+1})_{i\in[n]}\big) + \varepsilon\right) = \mathbb{P}\left(R_{j,j} > \mathrm{Quantile}_{1-\alpha}\big((R_{i,j})_{i\in[n+1]\setminus\{j\}}\big) + \varepsilon\right),$$

for each $j \in [n+1]$. As we know from (2.5), the quantity on the left-hand side is the probability that $Y_{n+1} \notin \mathcal{C}^{\varepsilon}_{\mathrm{CV}}(X_{n+1})$, i.e., the inflated interval *fails* to cover when the test point is (X_{n+1}, Y_{n+1}) and when the training data set is $(X_i, Y_i)_{i\in[n]}$. Analogously, the quantity on the right-hand side can therefore be interpreted as the probability that the inflated interval fails to cover, in an alternative scenario when the test point is instead (X_j, Y_j), and the training data set is $(X_i, Y_i)_{i\in[n+1]\setminus\{j\}}$.

By taking an average of the above equality over $j \in [n+1]$, we then have

$$\mathbb{P}\left(R_{n+1,n+1} > \mathrm{Quantile}_{1-\alpha}\big((R_{i,n+1})_{i\in[n]}\big) + \varepsilon\right) = \mathbb{E}\left[\frac{1}{n+1}\sum_{j=1}^{n+1} \mathbf{1}\left\{R_{j,j} > \mathrm{Quantile}_{1-\alpha}\big((R_{i,j})_{i\in[n+1]\setminus\{j\}}\big) + \varepsilon\right\}\right].$$

Next we make a deterministic claim: by Lemma A.1 below, it holds surely that

$$\frac{1}{n+1}\sum_{j=1}^{n+1} \mathbf{1}\left\{R_{j,j} > \mathrm{Quantile}_{1-\alpha}\big((R_{i,j})_{i\in[n+1]\setminus\{j\}}\big) + \varepsilon\right\} \leq \alpha + \sqrt{2\Delta_\varepsilon(R)} + \frac{1-\alpha}{n+1}, \tag{2.6}$$

where

$$\Delta_\varepsilon(R) = \frac{1}{(n+1)^2}\sum_{i=1}^{n+1}\sum_{j=1}^{n+1} \mathbf{1}\left\{|R_{i,i} - R_{i,j}| > \varepsilon\right\}.$$

We then have

$$\mathbb{P}\left(R_{n+1,n+1} > \mathrm{Quantile}_{1-\alpha}\big((R_{i,n+1})_{i\in[n]}\big) + \varepsilon\right) \leq \alpha + \mathbb{E}\left[\sqrt{2\Delta_\varepsilon(R)}\right] + \frac{1-\alpha}{n+1} \leq \alpha + \sqrt{2\delta} + \frac{1-\alpha}{n+1},$$

where the last step holds by the stability assumption, since $\mathbb{P}(|R_{i,i} - R_{i,j}| > \varepsilon) \leq \delta$ for all i, j and so $\mathbb{E}[\Delta_\varepsilon(R)] \leq \delta$. Recalling (2.5), this proves the desired bound on the coverage probability.

3 Hardness of testing algorithmic stability While many algorithms appear to exhibit stable behavior in empirical settings, certifying that stability properties hold for an algorithm $\mathcal{A}$ can be challenging: observing that $\mathcal{A}$ leads only to small changes in predictions when we randomly delete a data point from the available data set, does not imply that this is necessarily likely to occur on a different draw of data. Intuitively, if we would like to test whether $\mathcal{A}$ is stable at sample size n for some distribution P, this is certainly possible if we have vast amounts of available data sampled from P: given $K(n+1)$ many data points, we could simply evaluate $|\hat{f}(X_{n+1}) - \hat{f}_{-i}(X_{n+1})|$, K many times independently. But if we only have $\approx n$ available data points, is it possible to test this property?

As we have seen above in Section 2, this question has important practical applications: if we use the algorithm $\mathcal{A}$, along with n available training points, to construct a CV prediction interval for predictive inference on test data, can we be confident that $\mathcal{A}$ satisfies the stability conditions needed for this predictive interval to be valid? Aside from certain simple examples of algorithms that are known to be stable, we would need to certify this based on empirical tests of $\mathcal{A}$'s performance. In this section, we examine the question of whether it is possible to certify the stability of $(\mathcal{A}, P, n)$, and will establish hardness results for this question in the regime where limited data is available. For simplicity, we restrict our attention to symmetric algorithms $\mathcal{A}$ in this section. Throughout, we will write N to denote the number of data points sampled from P that are available for us to test $\mathcal{A}$, and n to denote the sample size at which we wish to certify stability of $\mathcal{A}$ (i.e., we are testing whether $(\mathcal{A}, P, n)$ is stable). We emphasize that in practice, given N data points, we are most interested in testing this question for $n = N$ or $n \approx N$: if we have N available training points, we are most interested in the downstream implications of stability (e.g., implications for generalization or for predictive inference) for a model that is trained on the entire available data set, not on some substantially smaller portion of the data.

We will write $T(\mathcal{A}, \mathcal{D}_N)$ to denote the data-dependent test, where we return $T(\mathcal{A}, \mathcal{D}_N) = 1$ to indicate that we are confident that stability holds, or $T(\mathcal{A}, \mathcal{D}_N) = 0$ otherwise.

3.1 Defining the black-box testing framework Our goal in this section is to determine whether stability can be certified empirically, rather than theoretically. Towards this aim, then, we need to define what it means to study the algorithm $\mathcal{A}$'s properties only through empirical observation. In particular, we cannot allow the test $T(\mathcal{A}, \mathcal{D}_N)$ to depend arbitrarily on $\mathcal{A}$—to take an extreme example, if our test is given by $T(\mathcal{A}, \mathcal{D}_N) = \mathbf{1}\{\mathcal{A}$ satisfies β-uniform-stability$\}$ (recalling Definition 1.3), this test is relying on theoretical properties of $\mathcal{A}$ rather than any empirical observations, and therefore may not be possible to evaluate for complex algorithms $\mathcal{A}$.

Instead, we need to require that T treats $\mathcal{A}$ as a "black box", meaning that it is possible to run $\mathcal{A}$ on various inputs and observe the outputs, but not to examine the actual definition of $\mathcal{A}$ to study its theoretical properties. To express this in a different way, if $\mathcal{A}$ is implemented in code, we may run the code but we may not read the code. We formalize this with the following definition.

DEFINITION 3.1. *Let $T(\mathcal{A}, \mathcal{D}_N)$ be a (possibly randomized) function. We call T a* black-box test *if it can be expressed in the form of the following iterative procedure, for some choice of functions $g, g^{(1)}, g^{(2)}, \ldots$. Sample random seeds $\zeta, \zeta^{(1)}, \zeta^{(2)}, \ldots \overset{\text{iid}}{\sim} \text{Unif}[0,1]$, and at each iteration $r \geq 1$,*

- *Generate a new data set and a new random seed, based on all information we have observed so far, and run the algorithm: at time $r = 1$ we generate*

$$(\mathcal{D}^{(1)}, \xi^{(1)}) = g^{(1)}(\mathcal{D}_N, \zeta^{(1)})$$

  and define $\hat{f}^{(1)} = \mathcal{A}(\mathcal{D}^{(1)}, \xi^{(1)})$, while at any time $r \geq 2$ we may also use past data sets and past trained models,

$$(\mathcal{D}^{(r)}, \xi^{(r)}) = g^{(r)}\big(\mathcal{D}_N, (\mathcal{D}^{(s)})_{1\leq s<r}, (\hat{f}^{(s)})_{1\leq s<r}, (\xi^{(s)})_{1\leq s<r}, (\zeta^{(s)})_{1\leq s\leq r}\big)$$

  and define $\hat{f}^{(r)} = \mathcal{A}(\mathcal{D}^{(r)}, \xi^{(r)})$.

- *If some stopping criterion is reached, exit the loop.*

Stopping at some finite (but possibly data-dependent) iteration $\hat{r}$, we then return

$$T(\mathcal{A}, \mathcal{D}_N) = g\big(\mathcal{D}_N, (\mathcal{D}^{(s)})_{1\leq s\leq \hat{r}}, (\hat{f}^{(s)})_{1\leq s\leq \hat{r}}, (\xi^{(s)})_{1\leq s\leq \hat{r}}, (\zeta^{(s)})_{1\leq s\leq \hat{r}}, \zeta\big).$$

This definition permits any type of empirical exploration of the behavior of $\mathcal{A}$. For example, we might choose to run $\mathcal{A}$ on subsets of the available data $\mathcal{D}_N$ and/or on random simulated data sets, and evaluate the resulting models with and without a deleted data point in the training set. Note that this definition does not place any computational constraints on the test—in principle, $T(\mathcal{A}, \mathcal{D}_N)$ may call $\mathcal{A}$ an unlimited number of times before determining its answer. (The hardness result that we will establish below holds even with this unlimited computational budget; we will comment on this more later on.)

Our next task is to define what it means for a test to be valid, in a distribution-free sense. Recalling the motivation of Section 2, our goal is to be able to certify an algorithm $\mathcal{A}$ as stable (with respect to P and n), so that we can then use cross-validation methods for predictive inference with models trained by $\mathcal{A}$. In particular, this means that there is a high cost to incorrectly certifying $\mathcal{A}$ as stable if in fact it does not satisfy stability: this type of error would lead to false confidence in our predictive inference method. With this in mind, given some desired error tolerance α we will say a test T is *distribution-free valid* (or simply "valid") if it holds that

$$\begin{aligned} &\mathbb{P}\big(T(\mathcal{A}, \mathcal{D}_N) = 1\big) \leq \alpha \text{ for any symmetric } \mathcal{A} \text{ and} \\ &\qquad \text{any distribution } P \text{ on } \mathcal{X} \times \mathcal{Y} \text{ such that } (\mathcal{A}, P, n) \text{ is not } (\varepsilon, \delta)\text{-stable.} \end{aligned} \tag{3.1}$$

Here the probability is taken with respect to a data set $\mathcal{D}_N$ drawn i.i.d. from P, as well as with respect to any auxiliary randomness in running the test (i.e., any random seeds used for implementing a randomized procedure). In other words, if we think of declaring $(\mathcal{A}, P, n)$ to be stable (i.e., $T(\mathcal{A}, \mathcal{D}_N) = 1$) as a detection, or a positive result, then we are enforcing a uniform bound on the probability of a false positive. Any test T that is a black-box test as in Definition 3.1, and is distribution-free valid as in (3.1), will be referred to as a *valid black-box test*.

3.2 A simple example: the Binomial test Next, we will define a simple test that satisfies the conditions above. Our goal is to provide a simple example that fits into the black-box framework, to provide a baseline against which other approaches may be compared.

For simplicity, assume that N is a multiple of $n+1$, so that $N = K(n+1)$ for some positive integer K. The N available data points can then be split into K batches of size $n+1$,

$$\mathcal{D}_N = \big(\underbrace{(X_1^{(1)}, Y_1^{(1)}), \ldots, (X_{n+1}^{(1)}, Y_{n+1}^{(1)})}_{\text{batch } 1}, \ \cdots, \ \underbrace{(X_1^{(K)}, Y_1^{(K)}), \ldots, (X_{n+1}^{(K)}, Y_{n+1}^{(K)})}_{\text{batch } K} \big).$$

We then define K many Bernoulli random variables,

$$B_k = \mathbf{1}\left\{ |\hat{f}^{(k)}(X_{n+1}^{(k)}) - \hat{f}_{-n}^{(k)}(X_{n+1}^{(k)})| > \varepsilon \right\} \text{ for } k = 1, \ldots, K,$$

where

$$\hat{f}^{(k)} = \mathcal{A}\big((X_1^{(k)}, Y_1^{(k)}), \ldots, (X_n^{(k)}, Y_n^{(k)})\big), \quad \hat{f}_{-n}^{(k)} = \mathcal{A}\big((X_1^{(k)}, Y_1^{(k)}), \ldots, (X_{n-1}^{(k)}, Y_{n-1}^{(k)})\big).$$

Finally, we observe that if $(\mathcal{A}, P, n)$ is not (ε, δ)-stable, then by definition we must have $\sum_{k=1}^K B_k \sim \text{Binomial}(K, \delta')$ for some $\delta' > \delta$. Therefore, to certify stability, it suffices to compare $\sum_{k=1}^K B_k$ against the $\text{Binomial}(K, \delta)$ distribution. Consequently, we define our (randomized) test via a one-tailed test of a Binomial proportion: choosing $\ell_* = \min\{\ell : \mathbb{P}(\text{Binomial}(K, \delta) \leq \ell) \geq \alpha\}$, define

$$T_{\text{Binom}}(\mathcal{A}, \mathcal{D}_N) = \begin{cases} 1, & \sum_{k=1}^K B_k < \ell_*, \\ \mathbf{1}\left\{ \zeta \leq \frac{\alpha - \mathbb{P}(\text{Binom}(K,\delta) < \ell_*)}{\mathbb{P}(\text{Binom}(K,\delta) = \ell_*)} \right\}, & \sum_{k=1}^K B_k = \ell_*, \\ 0, & \sum_{k=1}^K B_k > \ell_*, \end{cases} \tag{3.2}$$

for a random seed $\zeta \sim \text{Unif}[0,1]$ providing auxiliary randomness.

THEOREM 3.2 (Adapted from [27, Theorem 1]). *Assume N is a multiple of $n+1$, and let T_{Binom} be defined as in (3.2). Then T_{Binom} is a valid black-box test of (ε, δ)-stability. Moreover, if $(1-\delta)^{N/(n+1)} \geq \alpha$, then*

$$\mathbb{P}(T_{\text{Binom}}(\mathcal{A}, \mathcal{D}_N) = 1) = \frac{\alpha}{(1-\delta)^{N/(n+1)}}$$

for any algorithm $\mathcal{A}$ that is ε-uniformly-stable.

This theorem ensures that the test T_{Binom} offers a distribution-free way to certify that $\mathcal{A}$ is stable. But of course, the construction of T_{Binom} is extremely simple, and we are clearly not extracting all possible information from the available data. For example, in each batch k, instead of comparing $\hat{f}^{(k)}$ to all n possible leave-one-out models, we only compare to a single one. Moreover, we might hope that more sophisticated strategies, such as running $\mathcal{A}$ on resampled or perturbed versions of the data set, might reveal more about the behavior of $\mathcal{A}$ than this naive construction. However, as we will see next, this simple Binomial test is essentially optimal in a certain sense; the hardness of the problem of certifying stability prevents us from achieving better power than this naive test.

3.3 A hardness result We are now ready to state our hardness result for the problem of certifying algorithmic stability.

THEOREM 3.3 (Adapted from [27, Theorem 3] and [38, Theorem 1]). *Fix any $\varepsilon \geq 0$, $\delta, \alpha \in (0,1)$, and $n, N \geq 1$. Assume $|\mathcal{X} \times \mathcal{Y}| = \infty$, and let T be a valid black-box test of (ε, δ)-stability. Then, for any algorithm $\mathcal{A}$ and any distribution P on $\mathcal{X} \times \mathcal{Y}$, the power of the test T is bounded as*

$$\mathbb{P}(T(\mathcal{A}, \mathcal{D}_N) = 1) \leq \frac{\alpha}{(1-\delta)^{N/(n-1)}}.$$

This result reveals the limitations of the black-box distribution-free framework, by proving a bound on our power to certify any algorithm $\mathcal{A}$ as stable. In particular, if $N \approx n$ and δ is small, then this upper bound on the power of the test is not much higher than α—that is, our test is, at best, not much better than random. Moreover, comparing to Theorem 3.2, we can see that in certain regimes, the power of the naive Binomial test is essentially as high as the power of *any* valid black-box test—the only difference lies in the exponents, $\frac{N}{n+1}$ versus $\frac{N}{n-1}$, which are nearly identical when n is large.

Before proceeding to the proof, we briefly mention some extensions and alternative formulations of this result, which we have not stated formally here. First, [27] show that this hardness result holds even if we constrain the fitted models $\hat{f}$ (which are returned by $\mathcal{A}$) to lie in a very restricted class—for instance, even if $\mathcal{A}$ is only allowed to return constant functions. The intuition here is that the hardness result arises from possible discontinuous behavior in $\mathcal{A}$, the algorithm that maps data to predictors $\hat{f}$, rather than in the predictor $\hat{f}$ itself. Second, the hardness result above assumes that $\mathcal{X} \times \mathcal{Y}$ is an infinite space; this is because, with no computational constraints on our definition of a black-box algorithm (Definition 3.1), a finite data space would permit us to construct a test that simply checks for instability via exhaustive search. [38] extend the above hardness result to examine the tradeoff between the size of the (potentially finite) data space $\mathcal{X} \times \mathcal{Y}$ with a computational budget placed on the construction of the test T.

3.4 Proof of Theorem 3.2 First suppose $(\mathcal{A}, P, n)$ is not (ε, δ)-stable. Then for each batch k,

$$\mathbb{P}(B_k = 1) = \mathbb{P}\left(|\hat{f}^{(k)}(X_{n+1}^{(k)}) - \hat{f}_{-n}^{(k)}(X_{n+1}^{(k)})| > \varepsilon\right) > \delta$$

by definition of stability, and $B_1, \ldots, B_K$ are mutually independent since the data points are i.i.d. Therefore, $\sum_{k=1}^K B_k$ stochastically dominates the distribution $\text{Binom}(K, \delta)$. We can also calculate

$$\begin{aligned}
&\mathbb{P}\left(T_{\text{Binom}}(\mathcal{A}, \mathcal{D}_N) = 1\right) = \mathbb{P}\left(\sum_{k=1}^K B_k < \ell_*\right) + \mathbb{P}\left(\sum_{k=1}^K B_k = \ell_*, \zeta \leq \frac{\alpha - \mathbb{P}(\text{Binom}(K,\delta) < \ell_*)}{\mathbb{P}(\text{Binom}(K,\delta) = \ell_*)}\right) \\
&= \mathbb{P}\left(\sum_{k=1}^K B_k < \ell_*\right) + \frac{\alpha - \mathbb{P}(\text{Binom}(K,\delta) < \ell_*)}{\mathbb{P}(\text{Binom}(K,\delta) = \ell_*)} \cdot \mathbb{P}\left(\sum_{k=1}^K B_k = \ell_*\right) \\
&\leq \mathbb{P}(\text{Binom}(K,\delta) < \ell_*) + \frac{\alpha - \mathbb{P}(\text{Binom}(K,\delta) < \ell_*)}{\mathbb{P}(\text{Binom}(K,\delta) = \ell_*)} \cdot \mathbb{P}\left(\text{Binom}(K,\delta) = \ell_*\right) = \alpha,
\end{aligned}$$

where the last inequality uses the fact that $\sum_{k=1}^K B_k$ stochastically dominates $\text{Binom}(K, \delta)$. This verifies the validity of the test. Finally, if $(1-\delta)^K \geq \alpha$, we have $\ell_* = 0$ and $\mathbb{P}(\text{Binom}(K,\delta) = \ell_*) = (1-\delta)^K$, and so

$$T(\mathcal{A}, \mathcal{D}_N) = \mathbf{1}\left\{\sum_{k=1}^K B_k = 0 \text{ and } \zeta \leq \frac{\alpha}{(1-\delta)^K}\right\}.$$

If $\mathcal{A}$ is ε-uniformly-stable, $\sum_{k=1}^K B_k = 0$ almost surely, which completes the proof since $\mathbb{P}(\zeta \leq \frac{\alpha}{(1-\delta)^K}) = \frac{\alpha}{(1-\delta)^K}$.

3.5 Proof of Theorem 3.3 The idea behind the proof is that we construct a distribution P' similar to P, and an algorithm $\mathcal{A}'$ similar to $\mathcal{A}$, so that $(\mathcal{A}', P', n)$ is *not* stable, but nonetheless it is nearly impossible to distinguish between $(\mathcal{A}', P', n)$ and $(\mathcal{A}, P, n)$.

First, let $\gamma > 0$, and fix any point $(x_*, y_*) \in \mathcal{X} \times \mathcal{Y}$ with $\mathbb{P}\left((x_*, y_*) \in \mathcal{D}_N \cup \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(\hat{r})}\right) < \gamma$ (the existence of such a point is ensured by Lemma A.2). Let P' be the mixture distribution given by

$$P' = (1-c) \cdot P + c \cdot \delta_{(x_*, y_*)},$$

where $\delta_{(x_*,y_*)}$ denotes the point mass at (x_*, y_*), and where $c \in (0,1)$ is chosen to satisfy $(1-c) < (1-\delta)^{\frac{1}{n-1}}$. Let $\mathcal{A}'$ be defined as

$$\mathcal{A}'(\mathcal{D}) = \begin{cases} f_0, & |\mathcal{D}| = n \text{ and } (x_*, y_*) \in \mathcal{D}, \\ f_1, & |\mathcal{D}| = n-1 \text{ and } (x_*, y_*) \in \mathcal{D}, \\ \mathcal{A}(\mathcal{D}), & \text{otherwise}, \end{cases}$$

where f_0 and f_1 are constant functions, with $f_0(x) = c_0$ for all x and $f_1(x) = c_1$ for all x, for some constants c_0, c_1 with $|c_0 - c_1| > \varepsilon$.

Now let $\mathcal{D}_N$ be sampled i.i.d. from P, while $\mathcal{D}'_N$ is sampled i.i.d. from P'. By definition of P', we can construct a coupling of $\mathcal{D}_N$ and $\mathcal{D}'_N$ so that $\mathbb{P}(\mathcal{D}'_N = \mathcal{D}_N \mid \mathcal{D}_N) \geq (1-c)^N$, i.e., this is describing the event that all of the N data points in $\mathcal{D}'_N$ are drawn from the first component in the mixture. Therefore,

$$\mathbb{P}(T(\mathcal{A}', \mathcal{D}'_N) = 1) \geq (1-c)^N \mathbb{P}(T(\mathcal{A}', \mathcal{D}_N) = 1).$$

Next, if $(x_*, y_*) \notin \mathcal{D}_N \cup \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(\hat{r})}$, then the models returned by $\mathcal{A}'$ are all identical to the ones returned by $\mathcal{A}$. That is, if we run the black-box test initialized at $\mathcal{D}_N$, and (x_*, y_*) does not appear in this original data set or in any data set $\mathcal{D}^{(r)}$ arising at any stage of the process, then the outcome is identical whether we are calling algorithm $\mathcal{A}$ or algorithm $\mathcal{A}'$ when running the black-box test. Therefore,

$$\begin{aligned} \mathbb{P}(T(\mathcal{A}', \mathcal{D}_N) = 1) &\geq \mathbb{P}(T(\mathcal{A}', \mathcal{D}_N) = 1,\ (x_*, y_*) \notin \mathcal{D}_N \cup \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(\hat{r})}) \\ &= \mathbb{P}(T(\mathcal{A}, \mathcal{D}_N) = 1,\ (x_*, y_*) \notin \mathcal{D}_N \cup \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(\hat{r})}) \\ &\geq \mathbb{P}(T(\mathcal{A}, \mathcal{D}_N) = 1) - \mathbb{P}((x_*, y_*) \in \mathcal{D}_N \cup \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(\hat{r})}) \\ &\geq \mathbb{P}(T(\mathcal{A}, \mathcal{D}_N) = 1) - \gamma. \end{aligned}$$

Combining everything,

$$\mathbb{P}(T(\mathcal{A}, \mathcal{D}_N) = 1) \leq (1-c)^{-N} \mathbb{P}(T(\mathcal{A}', \mathcal{D}'_N) = 1) + \gamma.$$

Now we check the stability of $(\mathcal{A}', P', n)$. For $(X_1, Y_1), \ldots, (X_n, Y_n) \overset{\text{iid}}{\sim} P'$, we have

$$\mathbb{P}_{P'}(|\hat{f}(X_{n+1}) - \hat{f}_{-i}(X_{n+1})| > \varepsilon) \geq \mathbb{P}_{P'}(\hat{f} = f_0, \hat{f}_{-i} = f_1) \geq \mathbb{P}_{P'}\left((x_*, y_*) \in \{(X_j, Y_j)_{j \in [n] \setminus \{i\}}\}\right) \geq 1 - (1-c)^{n-1} > \delta,$$

where the last step holds since c is chosen to satisfy $(1-c) < (1-\delta)^{\frac{1}{n-1}}$. Thus $(\mathcal{A}', P', n)$ is *not* stable, and so we must have $\mathbb{P}(T(\mathcal{A}', \mathcal{D}'_N) = 1) \leq \alpha$, by validity of the test T. Therefore,

$$\mathbb{P}(T(\mathcal{A}, \mathcal{D}_N) = 1) \leq (1-c)^{-N} \alpha + \gamma.$$

Since this holds for any $\gamma > 0$, and any c satisfying $(1-c) < (1-\delta)^{\frac{1}{n-1}}$, this completes the proof.

4 A distribution-free stability guarantee via bagging

As we have seen in Section 3, it is impossible to construct a universally valid test that can reliably certify a black-box algorithm $\mathcal{A}$ as stable (in the regime where the available data is limited). Nonetheless, in practice we do not want to restrict ourselves to only using simple algorithms such as k-nearest neighbors for the sake of their theoretical stability guarantees. Is there any way to get the "best of both worlds"—can we use complex state-of-the-art algorithms that are not known theoretically to be stable, but nonetheless benefit from the positive implications of stability, such as generalization guarantees and validity of predictive inference?

In this section, we will prove that using *bagging*, i.e., averaging models that are fitted to random subsamples of the data, offers a stability guarantee regardless of the choice of base algorithm. In combination with the hardness result presented in Section 3, this stability guarantee provides an interesting perspective: while it is impossible to *certify* that a black-box algorithm $\mathcal{A}$ is stable, it is instead possible to *modify* $\mathcal{A}$ in order to ensure that stability holds, by replacing $\mathcal{A}$ with its bagged version.

4.1 Bagging an algorithm The method of bagging has been used for several decades to improve the performance of an algorithm, by increasing its stability and reducing variance. Bagging was proposed by [13], and the name is shorthand for "bootstrapped aggregation", where new data sets are sampled *with replacement* from the available data $(X_i, Y_i)_{i\in[n]}$, and the resulting fitted models are then aggregated (e.g., averaged) to produce a final bagged model. Any algorithm may be used as the base algorithm in a bagging procedure; we can think of bagging as a post-processing step, applied to an existing algorithm. A common application of bagging is the random forests method [13], which is the bagged version of a regression tree (or a decision tree, in a classification task). Another variant is where the resampled data sets are drawn *without* replacement from the n training points; this variant is sometimes called "subbagging" or "subagging" [3], to refer to the fact that it uses subsets of the training data. To illustrate the potential benefits of bagging, Figure 4.1 displays the stability of a base algorithm $\mathcal{A}$, and its bagged counterpart, in the case where $\mathcal{A}$ is given by logistic regression. We can see that, in this example, bagging has achieved the desired performance: in the higher-dimensional regime where $\mathcal{A}$ is highly unstable, its bagged version exhibits much stronger stability, while in the lower-dimensional regime where $\mathcal{A}$ is already stable, its bagged version returns predictions that are similar to those returned by $\mathcal{A}$. Indeed, bagging was initially proposed in [13] with exactly this aim: we would like to stabilize an unstable algorithm, and hopefully observe improved accuracy if this is the case, but if $\mathcal{A}$ is already stable then we can only hope that its performance is not degraded by bagging.

To formalize the method, consider any distribution over data sets resampled from the training data. Specifically, we will write $\hat{f}_{\mathbf{r}} = \mathcal{A}((X_i, Y_i)_{i\in\mathbf{r}})$ to denote the trained model obtained by running $\mathcal{A}$ on the resampled data set $(X_i, Y_i)_{i\in\mathbf{r}}$. Here $\mathbf{r}$ denotes a sequence (of any length) of indices in $[n]$. In particular, $\mathbf{r}$ may contain a subset of $[n]$, with no repeated values (if data points are sampled without replacement from the training data), or, $\mathbf{r}$ may contain repeated indices (if data points are sampled with replacement). We then define the bagged version of $\mathcal{A}$ as follows: for a fixed data set $(X_i, Y_i)_{i\in[n]}$, define the fitted model

$$\hat{f} = \mathcal{A}_{\text{bag}}((X_i, Y_i)_{i\in[n]}) \text{ where } \hat{f}(x) = \mathbb{E}\left[\hat{f}_{\mathbf{r}}(x)\right] \text{ and } \hat{f}_{\mathbf{r}} = \mathcal{A}((X_i, Y_i)_{i\in\mathbf{r}}).$$

In particular, in the case of subbagging, consider the distribution over $\mathbf{r}$ obtained by drawing m indices uniformly at random without replacement. Then

$$\hat{f} = \mathcal{A}_{\text{bag}}((X_i, Y_i)_{i\in[n]}) \text{ where } \hat{f}(x) = \frac{1}{\binom{n}{m}} \sum_{I\subseteq[n],|I|=m} \hat{f}_I(x) \text{ and } \hat{f}_I = \mathcal{A}((X_i, Y_i)_{i\in I}) \tag{4.1}$$

(or more generally, if $\mathcal{A}$ is not necessarily symmetric, then we average over all *ordered* subsets of m distinct indices). Of course, in practice, for large n and m it is not feasible to refit the model $\binom{n}{m}$ many times. Instead, we would simply average over a large number B of randomly drawn bags:

$$\hat{f} = \mathcal{A}_{\text{bag},B}((X_i, Y_i)_{i\in[n]}) \text{ where } \hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_{I_b}(x), \tag{4.2}$$

where now we average over bags $I_1, \ldots, I_B \subseteq [n]$, which are i.i.d. subsets of size m, each drawn uniformly without replacement from $[n]$.

4.2 Theoretical guarantee We are now ready to state a theoretical guarantee for the stability properties of any bagged algorithm.

THEOREM 4.1 (Adapted from [49, Theorem 8] and [50, Theorem 6]). *Let $\mathcal{A}$ be any algorithm, which returns functions whose outputs lie in $[-1, 1]$, and let $\mathcal{A}_{\text{bag}}$ be its bagged counterpart, obtained by subbagging m out of n data points and averaged over all possible subsamples, as in (4.1). Then for any training data set $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathcal{X} \times \mathbb{R}$ and any test point $X \in \mathcal{X}$, it holds that*

$$\frac{1}{n}\sum_{i=1}^{n} \left(\hat{f}(X) - \hat{f}_{-i}(X)\right)^2 \leq \frac{1}{n-1} \cdot \frac{p}{1-p},$$

where $\hat{f} = \mathcal{A}_{\text{bag}}((X_j, Y_j)_{j\in[n]})$ and $\hat{f}_{-i} = \mathcal{A}_{\text{bag}}((X_j, Y_j)_{j\in[n]\setminus\{i\}})$, and where $p = m/n$. If we instead use $\mathcal{A}_{\text{bag},B}$,

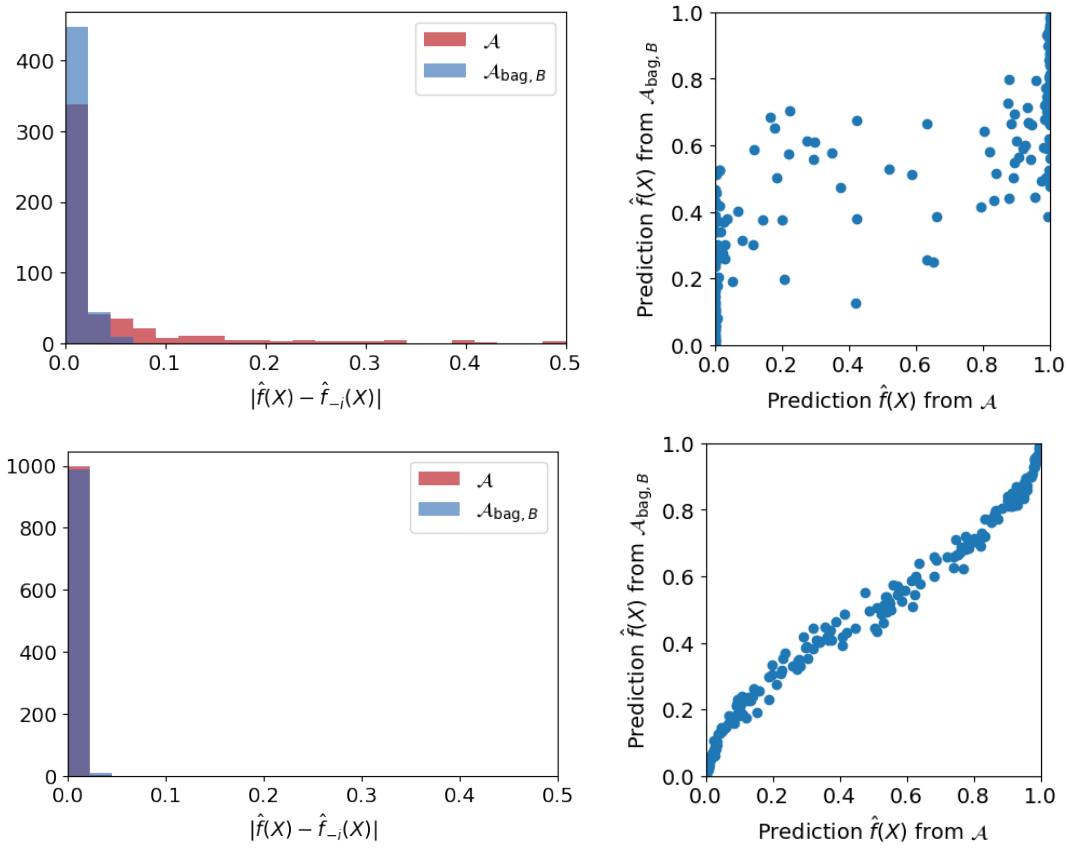

Figure 4.1: This figure displays the stability properties of the logistic regression algorithm, for data generated as $X_i \sim \mathcal{N}(0, \mathbf{I}_d)$, and $Y_i \sim \text{Bernoulli}(1/(1+e^{-X_i^\top \theta}))$, for $\theta = (0.1, \ldots, 0.1)$. We run the simulation with $d = 200, n = 500$ (top row) and $d = 200, n = 1000$ (bottom row). The figures on the left show the distribution of the leave-one-out perturbation, $|\hat{f}(X) - \hat{f}_{-i}(X)|$, for the algorithm $\mathcal{A}$ (logistic regression with a small ℓ_2 penalty) and its subbagged counterpart $\mathcal{A}_{\text{bag},B}$ (with $B = 10000$), on one test point. We can see that $\mathcal{A}$ may be stable or unstable, depending on d and n, but its bagged counterpart exhibits good stability in both cases. The figures on the right plot the prediction $\hat{f}(X)$ returned by $\mathcal{A}$ versus by $\mathcal{A}_{\text{bag},B}$, for 200 test points X. Predictions from $\mathcal{A}_{\text{bag},B}$ are similar to those from $\mathcal{A}$, when $\mathcal{A}$ was already stable (as in the bottom row).

which averages over B randomly drawn subsamples as in (4.2), we instead have the bound

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[\left(\hat{f}(X) - \hat{f}_{-i}(X)\right)^2\right] \leq \frac{1}{n-1} \cdot \frac{p}{1-p} + \frac{4}{B},$$

where now $\hat{f} = \mathcal{A}_{\text{bag},B}((X_j, Y_j)_{j\in[n]})$ and $\hat{f}_{-i} = \mathcal{A}_{\text{bag},B}((X_j, Y_j)_{j\in[n]\setminus\{i\}})$, and where the expected value is taken over the randomly sampled bags.

We remark that the assumption of boundedness for the predictions is primarily for convenience; related versions of this result can be obtained without assuming bounded outputs by instead adapting to the range of the predictions, but the calculations are more involved [49].

Comparing this result to our earlier definitions of stability, we can see that the bagged algorithm, $\mathcal{A}_{\text{bag}}$, satisfies a notion of stability that is strictly stronger than Definition 1.2—because the above bound holds for $\mathcal{A}_{\text{bag}}$ applied to *any* data set, while Definition 1.2 only requires a bound to hold on average over data sampled i.i.d. from some distribution P. In particular, setting $\beta^2 = \frac{1}{n-1} \cdot \frac{p}{1-p}$, we see that Theorem 4.1 implies that $\mathcal{A}_{\text{bag}}$ must be β-ℓ_2-stable with respect to any P (and an analogous bound, with a larger β, holds for bagging with B many

bags). Moreover, if we choose m to be not too small (e.g., $m = 0.9n$, or even, $m = n - \mathrm{o}(n)$), the accuracy of the bagged algorithm $\mathcal{A}_{\mathrm{bag}}$ intuitively should not be substantially worse than that of $\mathcal{A}$ since $\mathcal{A}_{\mathrm{bag}}$ is constructed from models trained on sample size m (which is nearly as large as n), and may be substantially better in settings where $\mathcal{A}$ suffers from instability. Earlier results in the literature have also examined the stability properties of bagging [29, 44, 20, 14], but in contrast to the guarantee presented here, prior results either require $\mathcal{A}$ to already satisfy some stability properties (and then establish that bagging improves the stability of the algorithm), or hold only for the setting $m = \mathrm{o}(n)$ (where models are fitted to substantially smaller subsamples of the data—and so we might lose accuracy in settings where the sample size is moderate).

While the result stated here pertains to bagging in the specific setting of computing a real-valued prediction, bagging can be applied more generally for aggregating data-dependent estimates in many other contexts, as well: for example, we might be estimating the density of a distribution, or, a vector of probabilities over a finite set of possible labels, in a classification task. The results of Theorem 4.1 are extended to this more general setting in [50]. Another related direction is in the setting of discrete output, such as classification or ranking tasks, where the returned output is a predicted label or predicted ranking. Notions of stability that are suited for such settings include the work of [1, 30, 22]. Further work by [48, 35] develops a framework that incorporates bagging in order to achieve distribution-free stability guarantees in these discrete problems.

4.3 Proof of Theorem 4.1 We now turn to the proof of our stability guarantee. First consider subbagging via $\mathcal{A}_{\mathrm{bag}}$, i.e., taking an average over all possible bags. Define the ith difference term as $d_i = \hat{f}(X) - \hat{f}_{-i}(X)$, for each $i \in [n]$. Our goal is to bound $\|d\|_2^2$, where $d = (d_1, \ldots, d_n)$. First we begin with a basic calculation: since $p = m/n = \mathbb{P}(i \in \mathbf{r})$, where $\mathbf{r} \subseteq [n]$ denotes a random subset of size m sampled uniformly without replacement, we have

$$\hat{f}(X) = \mathbb{E}[\hat{f}_{\mathbf{r}}(X)] = \mathbb{E}[\hat{f}_{\mathbf{r}}(X) \cdot \mathbf{1}\{i \in \mathbf{r}\}] + \mathbb{E}[\hat{f}_{\mathbf{r}}(X) \cdot \mathbf{1}\{i \notin \mathbf{r}\}] = \mathbb{E}[\hat{f}_{\mathbf{r}}(X) \cdot \mathbf{1}\{i \in \mathbf{r}\}] + (1-p) \cdot \hat{f}_{-i}(X),$$

since $\hat{f}_{-i}(X) = \mathbb{E}[\hat{f}_{\mathbf{r}}(X) \mid i \notin \mathbf{r}]$. Here and throughout the proof, all expectations and probabilities are computed with respect to the random draw of the bag $\mathbf{r}$ (and we emphasize that the data $(X_1, Y_1), \ldots, (X_n, Y_n)$ and test point X are treated as fixed).

Therefore, rearranging terms,

$$(1-p)\big(\hat{f}(X) - \hat{f}_{-i}(X)\big) = \mathbb{E}[\hat{f}_{\mathbf{r}}(X) \cdot \mathbf{1}\{i \in \mathbf{r}\}] - p\hat{f}(X) = \mathbb{E}[(\hat{f}_{\mathbf{r}}(X) - \hat{f}(X)) \cdot \mathbf{1}\{i \in \mathbf{r}\}],$$

by again using the fact that $p = \mathbb{P}(i \in \mathbf{r})$. This implies

$$d_i = \frac{\mathbb{E}[(\hat{f}_{\mathbf{r}}(X) - \hat{f}(X)) \cdot \mathbf{1}\{i \in \mathbf{r}\}]}{1-p} = \frac{\mathbb{E}[(\hat{f}_{\mathbf{r}}(X) - \hat{f}(X)) \cdot (\mathbf{1}\{i \in \mathbf{r}\} - p)]}{1-p},$$

where the first step holds by our calculation above, while the last step holds since $\mathbb{E}[\hat{f}_{\mathbf{r}}(X)] = \hat{f}(X)$, and consequently $\mathbb{E}[(\hat{f}_{\mathbf{r}}(X) - \hat{f}(X)) \cdot c] = 0$ for any constant c. We then have

$$\begin{aligned}
\|d\|_2^2 = \sum_{i=1}^n d_i \cdot d_i = \sum_{i=1}^n d_i \cdot \frac{\mathbb{E}[(\hat{f}_{\mathbf{r}}(X) - \hat{f}(X)) \cdot (\mathbf{1}\{i \in \mathbf{r}\} - p)]}{1-p} &= \mathbb{E}\left[\sum_{i=1}^n d_i \cdot \frac{(\hat{f}_{\mathbf{r}}(X) - \hat{f}(X)) \cdot (\mathbf{1}\{i \in \mathbf{r}\} - p)}{1-p}\right] \\
&\le \mathbb{E}\left[(\hat{f}_{\mathbf{r}}(X) - \hat{f}(X))^2\right]^{1/2} \cdot \mathbb{E}\left[\left(\sum_{i=1}^n \frac{(d_i \cdot (\mathbf{1}\{i \in \mathbf{r}\} - p)}{1-p}\right)^2\right]^{1/2} \le \mathbb{E}\left[\left(\sum_{i=1}^n \frac{d_i \cdot (\mathbf{1}\{i \in \mathbf{r}\} - p)}{1-p}\right)^2\right]^{1/2},
\end{aligned}$$

where the next-to-last step holds by the Cauchy–Schwarz inequality, while the last step uses the fact that $\hat{f}_{\mathbf{r}}(X)$ is a random variable taking values in $[-1, 1]$ and with mean $\hat{f}(X)$, and consequently $\mathbb{E}\left[(\hat{f}_{\mathbf{r}}(X) - \hat{f}(X))^2\right] = \mathrm{Var}(\hat{f}_{\mathbf{r}}(X)) \le 1$. We can rewrite the remaining expected value as

$$\begin{aligned}
\mathbb{E}\left[\left(\sum_{i=1}^n \frac{d_i \cdot (\mathbf{1}\{i \in \mathbf{r}\} - p)}{1-p}\right)^2\right] &= \sum_{i=1}^n \sum_{j=1}^n d_i d_j \mathbb{E}\left[\frac{\mathbf{1}\{i \in \mathbf{r}\} - p}{1-p} \cdot \frac{\mathbf{1}\{j \in \mathbf{r}\} - p}{1-p}\right] \\
&= \frac{1}{(1-p)^2} \sum_{i=1}^n \sum_{j=1}^n d_i d_j \mathrm{Cov}(\mathbf{1}\{i \in \mathbf{r}\}, \mathbf{1}\{j \in \mathbf{r}\}) = \frac{1}{(1-p)^2} d^\top \mathbf{C} d,
\end{aligned}$$

where $\mathbf{C}$ is the covariance matrix of the random vector $(\mathbf{1}\{1 \in \mathbf{r}\}, \ldots, \mathbf{1}\{n \in \mathbf{r}\})$, with entries $\mathbf{C}_{ii} = p(1-p)$ for all $i \in [n]$, and $\mathbf{C}_{ij} = -\frac{p(1-p)}{n-1}$ for all $i \neq j \in [n]$. Combining everything, we have shown that

$$\|d\|_2^2 \leq \frac{1}{1-p}\sqrt{d^\top \mathbf{C} d} \leq \frac{1}{1-p} \cdot \|d\|_2 \cdot \|\mathbf{C}\|^{1/2},$$

which implies that $\|d\|_2^2 \leq \frac{\|\mathbf{C}\|}{(1-p)^2}$. Finally, we calculate $\|\mathbf{C}\| = p(1-p) \cdot \frac{n}{n-1}$, which completes the proof of the first claim.

Next we consider subbagging with B many bags. We will now write $\hat{f}^* = \mathcal{A}_{\text{bag}}((X_j, Y_j)_{j\in[n]})$ and $\hat{f}^*_{-i} = \mathcal{A}_{\text{bag}}((X_j, Y_j)_{j\in[n]\setminus\{i\}}))$ to denote the fitted models that would be obtained if we averaged over all bags. Then, by the first part of the theorem, we have

$$\frac{1}{n}\sum_{i=1}^n \left(\hat{f}^*(X) - \hat{f}^*_{-i}(X)\right)^2 \leq \frac{1}{n-1} \cdot \frac{p}{1-p}.$$

Moreover, $\hat{f}^*(X)$ is the population mean of the distribution of $\hat{f}_{\mathbf{r}}(X) \in [-1, 1]$ (induced by the distribution of the bag $\mathbf{r}$), and similarly $\hat{f}(X)$ is the sample mean of the same distribution, obtained by averaging over B many i.i.d. draws. In other words, we have $\mathbb{E}[\hat{f}(X)] = \hat{f}^*(X)$, and $\text{Var}(\hat{f}(X)) \leq \frac{1}{B}$, where we recall that we treat the data as fixed and are computing mean and variance only with respect to the draw of the bags. Similarly, $\mathbb{E}[\hat{f}_{-i}(X)] = \hat{f}^*_{-i}(X)$, and $\text{Var}(\hat{f}_{-i}(X)) \leq \frac{1}{B}$. Therefore,

$$\frac{1}{n}\sum_{i=1}^n \mathbb{E}\left[\left(\hat{f}(X) - \hat{f}_{-i}(X)\right)^2\right] \leq \frac{1}{n}\sum_{i=1}^n \left\{\left(\hat{f}^*(X) - \hat{f}^*_{-i}(X)\right)^2 + \text{Var}\left(\hat{f}(X) - \hat{f}_{-i}(X)\right)\right\} \leq \frac{1}{n-1}\frac{p}{1-p} + \frac{4}{B},$$

where for the last step we use the fact that $\text{Var}(A+B) \leq 2\text{Var}(A) + 2\text{Var}(B)$, for any random variables A, B.

5 Discussion We conclude with a discussion of some of the implications and open questions raised by these results.

5.1 Distribution-free prediction with the bagged CV prediction interval To complete the picture, we return to the question of predictive inference via cross-validation. As we have seen above in Section 2, while the CV prediction interval does not yield a distribution-free predictive coverage guarantee in general, using a stable predictor is sufficient to guarantee coverage—and, any bagged algorithm is guaranteed to be stable, by the results of Section 4.

First, let us formally define the exact construction of the bagged CV prediction interval considered here. Let $I_1, \ldots, I_B \subseteq [n]$ denote i.i.d. subsets of size m sampled uniformly without replacement, and let $\hat{f}_{I_b} = \mathcal{A}((X_i, Y_i)_{i\in I_b})$ denote the model trained on the b-th subsample, as before. Define the aggregated model

$$\hat{f}(x) = \frac{1}{B}\sum_{b=1}^B \hat{f}_{I_b}(x). \tag{5.1}$$

We can then consider a prediction interval $\mathcal{C}_{\text{CV}}(X_{n+1}) = \hat{f}(X_{n+1}) \pm \hat{q}$, where now $\hat{q}$ is defined as

$$\hat{q} = \text{Quantile}_{1-\alpha}\big((|Y_i - \hat{f}_{-i}(X_i)|)_{i\in[n]} \text{ where } \hat{f}_{-i}(x) = \frac{1}{B_i}\sum_{b:i\notin I_b} \hat{f}_{I_b}(x), \tag{5.2}$$

where for each $i \in [n]$, we define $B_i = \sum_{b=1}^B \mathbf{1}\{i \notin I_b\}$ (and, if $B_i = 0$ then we simply set $\hat{f}_{-i}(x) = 0$—but when B is large, with high probability this will not be the case for any i). Then define the prediction interval $\mathcal{C}_{\text{CV}}$, and its inflated version $\mathcal{C}^{\varepsilon}_{\text{CV}}$, exactly as in Section 4. This method, and related variants, have been proposed and studied in many different works, including [16, 37, 24, 10, 11, 36, 59, 28], often under the name of "out-of-bag" estimates of the error—that is, $|Y_i - \hat{f}_{-i}(X_i)|$ is an "out-of-bag" residual, because $\hat{f}_{-i}(X_i)$ is computed by averaging over all models $\hat{f}_{I_b}$ fitted on bags I_b for which the data point i is not in the bag. With a slight inflation of this prediction interval, however, we are now equipped to provide a completely distribution-free guarantee.

COROLLARY 5.1. *Let P be any distribution on $\mathcal{X} \times \mathbb{R}$, and let $(X_1, Y_1), \ldots, (X_n, Y_n), (X_{n+1}, Y_{n+1}) \overset{\text{iid}}{\sim} P$. Let $\mathcal{A}$ be a symmetric algorithm, returning models whose outputs lie in $[-1, 1]$. Fix any $\varepsilon > 0$, and define the prediction interval*

$$\mathcal{C}^{\varepsilon}_{\text{CV}}(X_{n+1}) = \left[\hat{f}(X_{n+1}) - (\hat{q} + \varepsilon), \hat{f}(X_{n+1}) + (\hat{q} + \varepsilon)\right],$$

where $\hat{f}$ and $\hat{q}$ are defined as in (5.1) and (5.2). Then

$$\mathbb{P}\left(Y_{n+1} \in \mathcal{C}^{\varepsilon}_{\text{CV}}(X_{n+1})\right) \geq (1-\alpha)\left(1 - \frac{1}{n+1}\right) - \frac{\sqrt{2}}{\varepsilon}\sqrt{\frac{1}{n-1} \cdot \frac{p}{1-p} + \mathcal{O}\left(\frac{1}{B(1-p)}\right)},$$

where $p = m/n$.

This result follows as a corollary from the results developed above. Specifically, by Theorem 4.1, we know that the bagged version of $\mathcal{A}$ is β-ℓ_2-stable with respect to any distribution, for $\beta^2 = \frac{1}{n-1} \cdot \frac{p}{1-p} + \mathcal{O}(\frac{1}{B(1-p)})$ (note that this requires a slight extension of Theorem 4.1, since now $\hat{f}_{-i}$ is computed by aggregating $B_i \approx B(1-p)$, rather than B, many bags; for simplicity we omit the details). This implies that it is $(\varepsilon, (\beta/\varepsilon)^2)$-stable (as discussed in Section 1.1), and we can then apply Theorem 2.2.

5.2 Overview and open questions Through the results presented in this work, we have seen that stability offers an appealing alternative to distributional assumptions for the problem of predictive inference. However, there are fundamental limits on our ability to determine whether a black-box algorithm is stable. Bagging provides one solution to this challenge, by ensuring that any algorithm can be modified to ensure stability. Many open questions remain for each of these directions. First, for predictive inference via cross-validation, are the existing definitions of stability necessary, or can we find relaxations that are sufficient for guaranteeing validity of the prediction intervals? Second, for testing stability, is it possible that defining weaker notions of stability could circumvent the impossibility result—or if not, could we instead determine the mildest possible assumptions on $\mathcal{A}$ that would allow for testing stability? Finally, while we have seen that bagging leads to a distribution-free guarantee of stability, it comes at a steep computational cost, since we must refit the model many times—is this an unavoidable cost, or are there alternative post-processing steps that can pair with any black-box algorithm to provide a stability guarantee?

A Technical lemmas

LEMMA A.1. *Let $M \in \mathbb{R}^{m \times m}$ be a fixed matrix. Then*

$$\sum_{j=1}^{m} \mathbf{1}\left\{M_{j,j} > \text{Quantile}_{1-\alpha}\big((M_{i,j})_{i \in [m] \setminus \{j\}}\big) + \varepsilon\right\} \leq \left(\alpha + \sqrt{2\Delta_{\varepsilon}(M)}\right) m + (1-\alpha),$$

where $\Delta_{\varepsilon}(M) = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} \mathbf{1}\{|M_{i,i} - M_{i,j}| > \varepsilon\}$.

We can understand the intuition behind this result by considering the extreme case where M is constant across all its rows, i.e., $M_{i,1} = \cdots = M_{i,m}$ for all i. Then we can take $\varepsilon = 0$, and $\Delta_{\varepsilon}(M) = 0$. We also have

$$\mathbf{1}\left\{M_{j,j} > \text{Quantile}_{1-\alpha}\big((M_{i,j})_{i \in [m] \setminus \{j\}}\big)\right\} = \mathbf{1}\left\{M_{j,j} > \text{Quantile}_{1-\alpha}\big((M_{i,i})_{i \in [m] \setminus \{j\}}\big)\right\},$$

which can only hold for at most $\alpha m + (1-\alpha)$ many j's—that is, the $\alpha m + (1-\alpha)$ largest diagonal entries.

Proof of Lemma A.1. Without loss of generality, we can assume that $M_{1,1} \geq \cdots \geq M_{m,m}$ (i.e., we can permute the indices $\{1, \ldots, m\}$ as needed so that this is the case). Let $k_* = \lfloor \alpha m + (1-\alpha) \rfloor$. Define the set $S_{\varepsilon}(M) = \{j \in [m] : M_{j,j} > \text{Quantile}_{1-\alpha}\big((M_{i,j})_{i \in [m] \setminus \{j\}}\big) + \varepsilon\}$, so that our goal is to bound $|S_{\varepsilon}(M)|$.

Fix any $j \in [m]$. We have

$$j \in S_{\varepsilon}(M) \iff M_{j,j} > \text{Quantile}_{1-\alpha}\big((M_{i,j})_{i \in [m] \setminus \{j\}}\big) + \varepsilon \iff \sum_{i=1}^{m} \mathbf{1}\{M_{j,j} > M_{i,j} + \varepsilon\} \geq \lceil (1-\alpha)(m-1) \rceil = m - k_*,$$

by definition of the quantile. Moreover, if $M_{j,j} > M_{i,j} + \varepsilon$ and $i \leq j$, then we must have $|M_{i,i} - M_{i,j}| > \varepsilon$ (because $M_{i,i} \geq M_{j,j}$ for any $i \leq j$). Therefore,

$$\sum_{i=1}^{m} \mathbf{1}\{M_{j,j} > M_{i,j} + \varepsilon\} \leq \sum_{i=1}^{m} \mathbf{1}\{|M_{i,i} - M_{i,j}| > \varepsilon \text{ or } i > j\} \leq \sum_{i=1}^{m} \mathbf{1}\{|M_{i,i} - M_{i,j}| > \varepsilon\} + (m-j).$$

Thus, for all $j \in [m]$,

$$j \in S_\varepsilon(M) \implies \sum_{i=1}^{m} \mathbf{1}\{|M_{i,i} - M_{i,j}| > \varepsilon\} \geq j - k_*.$$

Now we are ready to bound $|S_\varepsilon(M)|$. We calculate

$$\begin{aligned}
m^2 \Delta_\varepsilon(M) &= \sum_{i=1}^{m}\sum_{j=1}^{m} \mathbf{1}\{|M_{i,i} - M_{i,j}| > \varepsilon\} \\
&\geq \sum_{i=1}^{m}\sum_{j=1}^{m} \mathbf{1}\{|M_{i,i} - M_{i,j}| > \varepsilon\} \cdot \mathbf{1}\{j > k_*, j \in S_\varepsilon(M)\} \\
&\geq \sum_{j=1}^{m} \mathbf{1}\{j > k_*, j \in S_\varepsilon(M)\} \cdot (j - k_*) \\
&\geq \sum_{j=1}^{m} \mathbf{1}\{j > k_*, j \leq |S_\varepsilon(M)|\} \cdot (j - k_*) = \frac{(|S_\varepsilon(M)| - k_*)_+ \cdot \big((|S_\varepsilon(M)| - k_*)_+ + 1\big)}{2},
\end{aligned}$$

where the last inequality holds since, fixing $|S_\varepsilon(M)|$, the minimal value of the sum $\sum_{j=1}^{m} \mathbf{1}\{j > k_*, j \in S_\varepsilon(M)\} \cdot (j - k_*)$ is attained by choosing $S_\varepsilon(M) = \{1, \ldots, |S_\varepsilon(M)|\}$. Rearranging terms, we therefore see that

$$(|S_\varepsilon(M)| - k_*)_+ \leq m \cdot \sqrt{2\Delta_\varepsilon(M)},$$

which completes the proof by definition of k_*. ∎

Lemma A.2. *Fix any $\gamma > 0$. Under the notation and assumptions of Theorem 3.3, there exists some $(x_*, y_*) \in \mathcal{X} \times \mathcal{Y}$ such that*

$$\mathbb{P}\left((x_*, y_*) \in \mathcal{D}_N \cup \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(\hat{r})}\right) < \gamma.$$

Proof of Lemma A.2. Since $|\mathcal{X} \times \mathcal{Y}| = \infty$, it suffices to show that the number of data points (x, y) with $\mathbb{P}\left((x, y) \in \mathcal{D}_N \cup \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(\hat{r})}\right) \geq \gamma$ is bounded by some finite number.

First, the random variable $\hat{r}$ is finite almost surely, so we can find some finite integer B_1 such that $\mathbb{P}(\hat{r} \leq B_1) \geq 1 - \gamma/4$. Next, each data set $\mathcal{D}^{(r)}$ contains N_r many data points, and N_r is finite almost surely. Therefore, we can find some B_2 such that $\mathbb{P}(N_1 + \cdots + N_{B_1} \leq B_2) \geq 1 - \gamma/4$. Combining these bounds, we see that $\mathbb{P}(\mathcal{E}) \geq 1 - \gamma/2$, where $\mathcal{E}$ is the event that $\mathcal{D}_N \cup \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(\hat{r})}$ contains at most $N + B_2$ many data points.

Now let $M \geq 1$ be any integer, and suppose there exist M unique data points $(x_1, y_1), \ldots, (x_M, y_M) \in \mathcal{X} \times \mathcal{Y}$ with $\mathbb{P}\left((x_m, y_m) \in \mathcal{D}_N \cup \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(\hat{r})}\right) \geq \gamma$ for each m. We then calculate

$$\begin{aligned}
M \cdot \gamma \leq \sum_{m=1}^{M} \mathbb{P}\left((x_m, y_m) \in \mathcal{D}_N \cup \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(\hat{r})}\right) &= \mathbb{E}\left[\sum_{m=1}^{M} \mathbf{1}\left\{(x_m, y_m) \in \mathcal{D}_N \cup \mathcal{D}^{(1)} \cup \cdots \cup \mathcal{D}^{(\hat{r})}\right\}\right] \\
&\leq \mathbb{E}\left[(N + B_2) + M \cdot \mathbf{1}_{\mathcal{E}^c}\right] = N + B_2 + M \cdot \mathbb{P}(\mathcal{E}^c) \leq N + B_2 + \frac{\gamma}{2} \cdot M,
\end{aligned}$$

where the second inequality holds by definition of the event $\mathcal{E}$, since the data points $(x_1, y_1), \ldots, (x_M, y_M)$ are unique. We must therefore have $M \leq \frac{N + B_2}{\gamma/2}$, which completes the proof. ∎

Acknowledgments The author gratefully acknowledges support from the National Science Foundation via grants DMS-1654076 and DMS-2023109, the Office of Naval Research via grants N00014-20-1-2337 and N00014-24-1-2544, the Sloan Foundation, and the American Institute of Mathematics. The author thanks Yuetian Luo and Richard Samworth for helpful feedback.

References

[1] S. Agarwal and P. Niyogi, *Stability and generalization of bipartite ranking algorithms*, in International Conference on Computational Learning Theory, Springer, 2005, pp. 32–47.

[2] N. Amann, H. Leeb, and L. Steinberger, *Assumption-lean conditional predictive inference via the jackknife and the jackknife+*, arXiv preprint arXiv:2312.14596, (2023).

[3] S. Andonova, A. Elisseeff, T. Evgeniou, and M. Pontil, *A simple algorithm for learning stable machines*, in ECAI, 2002, pp. 513–517.

[4] A. N. Angelopoulos, R. F. Barber, and S. Bates, *Theoretical foundations of conformal prediction*, arXiv preprint arXiv:2411.11824, (2024).

[5] A. N. Angelopoulos and S. Bates, *Conformal prediction: A gentle introduction*, Foundations and trends® in machine learning, 16 (2023), pp. 494–591.

[6] M. Austern and W. Zhou, *Asymptotics of cross-validation*, arXiv preprint arXiv:2001.11111, (2020).

[7] R. F. Barber, E. J. Candes, A. Ramdas, and R. J. Tibshirani, *Predictive inference with the jackknife+*, The Annals of Statistics, 49 (2021), pp. 486–507.

[8] A. Bayle, L. Janson, and L. Mackey, *The relative instability of model comparison with cross-validation*, arXiv preprint arXiv:2508.04409, (2025).

[9] P. Bayle, A. Bayle, L. Janson, and L. Mackey, *Cross-validation confidence intervals for test error*, Advances in Neural Information Processing Systems, 33 (2020), pp. 16339–16350.

[10] H. Bostrom, L. Asker, R. Gurung, I. Karlsson, T. Lindgren, and P. Papapetrou, *Conformal prediction using random survival forests*, in 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2017, pp. 812–817.

[11] H. Boström, H. Linusson, T. Löfström, and U. Johansson, *Accelerating difficulty estimation for conformal regression forests*, Annals of Mathematics and Artificial Intelligence, 81 (2017), pp. 125–144.

[12] O. Bousquet and A. Elisseeff, *Stability and generalization*, Journal of machine learning research, 2 (2002), pp. 499–526.

[13] L. Breiman, *Bagging predictors*, Machine learning, 24 (1996), pp. 123–140.

[14] Q. Chen, V. Syrgkanis, and M. Austern, *Debiased machine learning without sample-splitting for stable estimators*, Advances in Neural Information Processing Systems, 35 (2022), pp. 3096–3109.

[15] W. Chen, K.-J. Chun, and R. F. Barber, *Discretized conformal prediction for efficient distribution-free inference*, Stat, 7 (2018), p. e173.

[16] D. Devetyarov and I. Nouretdinov, *Prediction with confidence based on a random forest classifier*, in IFIP international conference on artificial intelligence applications and innovations, Springer, 2010, pp. 37–44.

[17] L. Devroye and T. Wagner, *Distribution-free inequalities for the deleted and holdout error estimates*, IEEE Transactions on Information Theory, 25 (1979), pp. 202–207.

[18] C. Dwork and A. Roth, *The algorithmic foundations of differential privacy*, Foundations and trends® in theoretical computer science, 9 (2014), pp. 211–407.

[19] B. Efron and G. Gong, *A leisurely look at the bootstrap, the jackknife, and cross-validation*, The American Statistician, 37 (1983), pp. 36–48.

[20] A. Elisseeff, T. Evgeniou, M. Pontil, and L. P. Kaelbing, *Stability of randomized learning algorithms.*, Journal of Machine Learning Research, 6 (2005).

[21] Y. Freund, Y. Mansour, and R. E. Schapire, *Generalization bounds for averaged classifiers*, Annals of Statistics, (2004), pp. 1698–1722.

[22] W. Gao and Z.-H. Zhou, *Uniform convergence, stability and learnability for ranking problems.*, in IJCAI, 2013, pp. 1337–1343.

[23] M. Hardt, B. Recht, and Y. Singer, *Train faster, generalize better: Stability of stochastic gradient descent*, in International conference on machine learning, PMLR, 2016, pp. 1225–1234.

[24] U. Johansson, H. Boström, T. Löfström, and H. Linusson, *Regression conformal prediction with random forests*, Machine learning, 97 (2014), pp. 155–176.

[25] S. Kale, R. Kumar, and S. Vassilvitskii, *Cross-validation and mean-square stability.*, in ICS, 2011, pp. 487–495.

[26] M. Kearns and D. Ron, *Algorithmic stability and sanity-check bounds for leave-one-out cross-validation*, in Proceedings of the tenth annual conference on Computational learning theory, 1997, pp. 152–162.

[27] B. Kim and R. F. Barber, *Black-box tests for algorithmic stability*, Information and Inference: A Journal of the IMA, 12 (2023), pp. 2690–2719.

[28] B. Kim, C. Xu, and R. Barber, *Predictive inference is free with the jackknife+-after-bootstrap*, Advances in Neural Information Processing Systems, 33 (2020), pp. 4138–4149.

[29] S. Kutin and P. Niyogi, *The interaction of stability and weakness in adaboost*, University of Chicago Department of Computer Science, (2001).

[30] Y. Lan, T.-Y. Liu, T. Qin, Z. Ma, and H. Li, *Query-level stability and generalization in learning to rank*, in Proceedings of the 25th international conference on Machine learning, 2008, pp. 512–519.

[31] J. Lei, *Fast exact conformalization of the lasso using piecewise linear homotopy*, Biometrika, 106 (2019), pp. 749–764.

[32] J. Lei, *A modern theory of cross-validation through the lens of stability*, arXiv preprint arXiv:2505.23592, (2025).

[33] J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman, *Distribution-free predictive inference for regression*, Journal of the American Statistical Association, 113 (2018), pp. 1094–1111.

[34] R. Liang and R. F. Barber, *Algorithmic stability implies training-conditional coverage for distribution-free prediction methods*, arXiv preprint arXiv:2311.04295, (2023).

[35] R. Liang, J. A. Soloff, R. F. Barber, and R. Willett, *Assumption-free stability for ranking problems*, arXiv preprint arXiv:2506.02257, (2025).

[36] H. Linusson, U. Johansson, and H. Boström, *Efficient conformal predictor ensembles*, Neurocomputing, 397 (2020), pp. 266–278.

[37] T. Löfström, U. Johansson, and H. Boström, *Effective utilization of data in inductive conformal prediction using ensembles of neural networks*, in The 2013 International Joint Conference on Neural Networks (IJCNN), IEEE, 2013, pp. 1–8.

[38] Y. Luo and R. F. Barber, *Is algorithmic stability testable? a unified framework under computational constraints*, arXiv preprint arXiv:2405.15107, (2024).

[39] S. Mukherjee, P. Niyogi, T. Poggio, and R. Rifkin, *Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization*, Advances in Computational Mathematics, 25 (2006), pp. 161–193.

[40] E. Ndiaye, *Stable conformal prediction sets*, in International Conference on Machine Learning, PMLR, 2022, pp. 16462–16479.

[41] I. Nouretdinov, T. Melluish, and V. Vovk, *Ridge regression confidence machine*, in Proceedings of the Eighteenth International Conference on Machine Learning, 2001, pp. 385–392.

[42] H. Papadopoulos, K. Proedrou, V. Vovk, and A. Gammerman, *Inductive confidence machines for regression*, in European conference on machine learning, Springer, 2002, pp. 345–356.

[43] T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi, *General conditions for predictivity in learning theory*, Nature, 428 (2004), pp. 419–422.

[44] T. Poggio, R. Rifkin, S. Mukherjee, and A. Rakhlin, *Bagging regularizes*, (2002).

[45] W. H. Rogers and T. J. Wagner, *A finite sample distribution-free performance bound for local discrimination rules*, The Annals of Statistics, (1978), pp. 506–514.

[46] G. Shafer and V. Vovk, *A tutorial on conformal prediction.*, Journal of Machine Learning Research, 9 (2008).

[47] S. Shalev-Shwartz, O. Shamir, N. Srebro, and K. Sridharan, *Learnability, stability and uniform convergence*, The Journal of Machine Learning Research, 11 (2010), pp. 2635–2670.

[48] J. Soloff, R. Barber, and R. Willett, *Building a stable classifier with the inflated argmax*, Advances in Neural Information Processing Systems, 37 (2024), pp. 70349–70380.

[49] J. A. Soloff, R. F. Barber, and R. Willett, *Bagging provides assumption-free stability*, Journal of Machine Learning Research, 25 (2024), pp. 1–35.

[50] J. A. Soloff, R. F. Barber, and R. Willett, *Stability via resampling: statistical problems beyond the real line*, arXiv preprint arXiv:2405.09511, (2024).

[51] L. Steinberger and H. Leeb, *Conditional predictive inference for high-dimensional stable algorithms*, arXiv preprint arXiv:1809.01412, (2018).

[52] M. Stone, *Cross-validatory choice and assessment of statistical predictions*, Journal of the royal statistical society: Series B (Methodological), 36 (1974), pp. 111–133.

[53] V. Vovk, *Cross-conformal predictors*, Annals of Mathematics and Artificial Intelligence, 74 (2015), pp. 9–28.

[54] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic learning in a random world*, Springer, 2005.

[55] V. Vovk, I. Nouretdinov, V. Manokhin, and A. Gammerman, *Cross-conformal predictive distributions*, in conformal and probabilistic prediction and applications, PMLR, 2018, pp. 37–51.

[56] Y.-X. Wang, J. Lei, and S. E. Fienberg, *Learning with differential privacy: Stability, learnability and the sufficiency and necessity of erm principle*, Journal of Machine Learning Research, 17 (2016), pp. 1–40.

[57] A. Wibisono, L. Rosasco, and T. Poggio, *Sufficient conditions for uniform stability of regularization algorithms*, Computer Science and Artificial Intelligence Laboratory Technical Report, MIT-CSAIL-TR-2009-060, (2009), p. 156.

[58] H. Xu, C. Caramanis, and S. Mannor, *Sparse algorithms are not stable: A no-free-lunch theorem*, IEEE transactions on pattern analysis and machine intelligence, 34 (2011), pp. 187–193.

[59] H. Zhang, J. Zimmerman, D. Nettleton, and D. J. Nordman, *Random forest prediction intervals*, The American Statistician, (2020).