The role of algorithmic stability in distribution-free inference

Rina Foygel Barber, University of Chicago

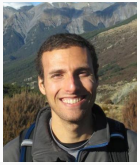
http://rinafb.github.io

Overview

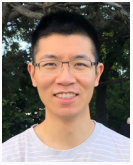
- Intro: algorithmic stability
- Part 1: Predictive inference via cross-validation
- Part 2: Hardness of testing stability
- Part 3: Stability guarantees via bagging


Emmanuel Candès


Aaditya Ramdas


Ryan Tibshirani


Byol Kim

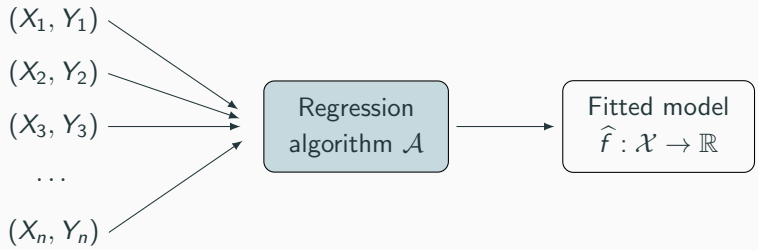

Yuetian Luo


Jake Soloff


Rebecca Willett

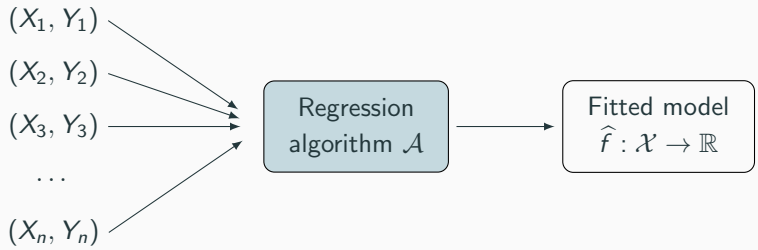
Research supported by NSF, ONR, Sloan, American Institute of Mathematics (AIM)

Intro: algorithmic stability



Stability: small perturbations to training data $\rightsquigarrow$ only small changes in $\widehat{f}$

Intro: algorithmic stability



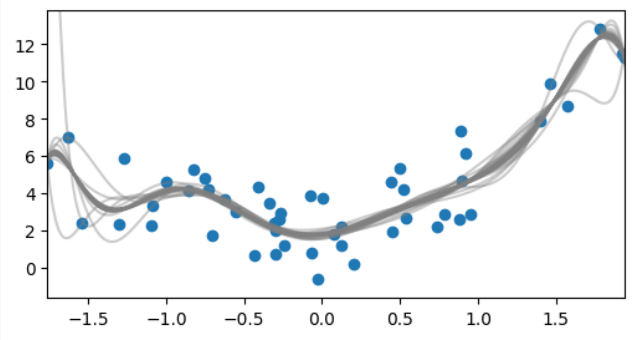
Stability: small perturbations to training data $\rightsquigarrow$ only small changes in $\widehat{f}$

e.g., deleting one training point

Intro: algorithmic stability

How to define stability?

- Uniform/worst-case — no data point may be highly influential
- Average-case — most data points are not highly influential



Intro: algorithmic stability

Definition: β-uniform-stability

For any data points $(X_1, Y_1), \ldots, (X_n, Y_n), (X, Y)$,

$$\left|\widehat{f}(X) - \widehat{f}_{-i}(X)\right| \leq \beta \quad \text{for all } i \in [n]$$

$\widehat{f} = \mathcal{A}((X_j, Y_j)_{j \in [n]})$

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

Intro: algorithmic stability

Definition: β-uniform-stability

For any data points $(X_1, Y_1), \ldots, (X_n, Y_n), (X, Y)$,

$$\left|\widehat{f}(X) - \widehat{f}_{-i}(X)\right| \leq \beta \quad \text{for all } i \in [n]$$

$\widehat{f} = \mathcal{A}((X_j, Y_j)_{j \in [n]})$ $\qquad$ $\widehat{f}_{-i} = \mathcal{A}((X_j, Y_j)_{j \in [n] \setminus \{i\}})$

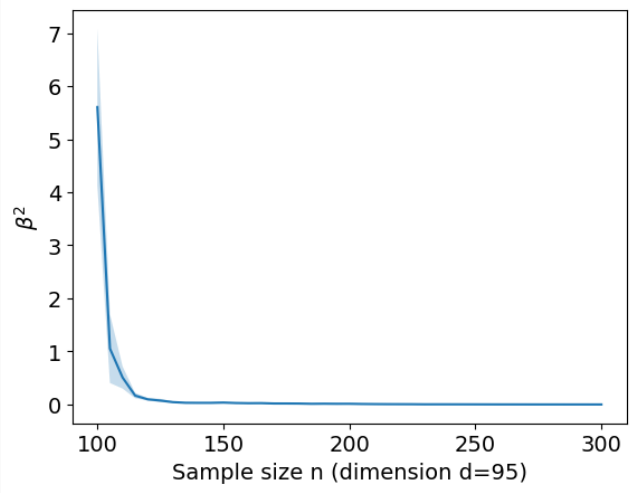
Definition: β-ℓ_2-stability

For data points $(X_1, Y_1), \ldots, (X_n, Y_n), (X, Y)$,

$$\frac{1}{n} \sum_{i=1}^{n} \left(\widehat{f}(X) - \widehat{f}_{-i}(X)\right)^2 \leq \beta^2$$

Intro: algorithmic stability

Example: ℓ_2-stability for least squares



Intro: algorithmic stability

What does algorithmic stability offer?

- Asymptotics of cross-validation[1]
- Reproducibility[2], privacy[3]
- Learnability — can we learn a model $\widehat{f}$ with low risk (test error)?[4]
- Generalization — can we estimate the risk (test error) of $\widehat{f}$?[5]

[1] Kale et al. 2011; Austern & Zhou 2020; Bayle et al. 2020; Bayle, Janson, Mackey 2025; Lei 2025

[2] Yu 2013; Murdoch et al. 2019

[3] Dwork & Roth 2014; Wang, Lei, Fienberg 2016

[4] Shalev-Shwartz et al. 2010

[5] Kutin & Niyogi 2001; Bousquet & Elisseeff 2002; Freund, Mansour, Schapire 2004; Mukherjee et al. 2006

Intro: algorithmic stability

What does algorithmic stability offer?

- Asymptotics of cross-validation[1]
- Reproducibility[2], privacy[3]
- Learnability — can we learn a model $\widehat{f}$ with low risk (test error)?[4]
- Generalization — can we estimate the risk (test error) of $\widehat{f}$?[5]

Key question (Part 1)

Can we use algorithmic stability to perform predictive inference without relying on model assumptions?

[1] Kale et al. 2011; Austern & Zhou 2020; Bayle et al. 2020; Bayle, Janson, Mackey 2025; Lei 2025

[2] Yu 2013; Murdoch et al. 2019

[3] Dwork & Roth 2014; Wang, Lei, Fienberg 2016

[4] Shalev-Shwartz et al. 2010

[5] Kutin & Niyogi 2001; Bousquet & Elisseeff 2002; Freund, Mansour, Schapire 2004; Mukherjee et al. 2006

Intro: algorithmic stability

Some existing algorithms are provably stable:

k-nearest neighbors, ridge regression, stochastic gradient descent[6]

But many algorithms are sometimes unstable,

or are too complex to theoretically verify stability.

[6] Rogers & Wagner 1978; Devroye & Wagner 1979; Bousquet & Elisseeff 2002; Wibinoso et al. 2009; Hardt et al. 2016

Intro: algorithmic stability

Some existing algorithms are provably stable:

k-nearest neighbors, ridge regression, stochastic gradient descent[6]

But many algorithms are sometimes unstable,

or are too complex to theoretically verify stability.

Key question (Part 2)

Is there a test that can empirically verify whether $\mathcal{A}$ is stable,

for complex black-box algorithms?

Key question (Part 3)

Is there a way to convert any algorithm into a stable algorithm,

while still retaining its good performance?

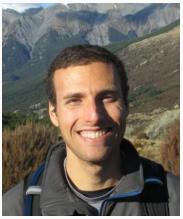
[6] Rogers & Wagner 1978; Devroye & Wagner 1979; Bousquet & Elisseeff 2002; Wibinoso et al. 2009; Hardt et al. 2016

Part 1: Predictive inference via cross-validation

Predictive inference with the jackknife+, B., Candès, Ramdas, Tibshirani

Emmanuel Candès
Stanford

Aaditya Ramdas
Carnegie Mellon



Ryan Tibshirani
UC Berkeley

Distribution-free predictive inference

Given a fitted model

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

how do we construct prediction intervals for future data?

$$\text{Observe } X_{n+1} \;\rightsquigarrow\; \text{prediction } \widehat{f}(X_{n+1}) \text{ for } Y_{n+1}$$

Distribution-free predictive inference

Given a fitted model

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

how do we construct prediction intervals for future data?

$$\text{Observe } X_{n+1} \;\rightsquigarrow\; \text{prediction } \widehat{f}(X_{n+1}) \text{ for } Y_{n+1}$$

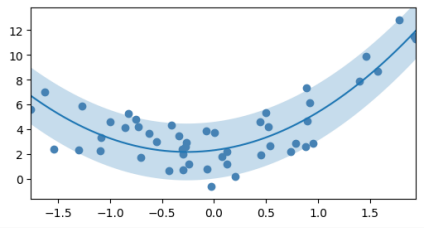
Aim to avoid distributional assumptions—

- Noise $Y_{n+1} - \mathbb{E}\left[Y_{n+1} \mid X_{n+1}\right]$
- Error in estimation $\widehat{f}(X_{n+1}) - \mathbb{E}\left[Y_{n+1} \mid X_{n+1}\right]$

Distribution-free predictive inference

Goal: predictive coverage without model assumptions

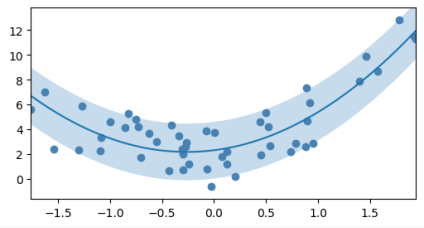
$$\mathbb{P}\left\{ Y_{n+1} \in \left[\widehat{f}(X_{n+1}) - q, \widehat{f}(X_{n+1}) + q \right] \right\} \geq 1 - \alpha$$



Distribution-free predictive inference

Goal: predictive coverage without model assumptions

$$\mathbb{P}\left\{ Y_{n+1} \in \left[\widehat{f}(X_{n+1}) - q, \widehat{f}(X_{n+1}) + q \right] \right\} \geq 1 - \alpha$$



Key question

Can we use algorithmic stability to perform predictive inference without relying on model assumptions?

- Use training residuals?

$$q = \text{Quantile}_{1-\alpha}\left(\{|Y_i - \widehat{f}(X_i)|\}_{i\in[n]}\right)$$

Problem: $\widehat{f}$ may overfit to training data $\rightsquigarrow$ undercoverage

Distribution-free predictive inference

- Use training residuals?

$$q = \text{Quantile}_{1-\alpha}\left(\{|Y_i - \widehat{f}(X_i)|\}_{i\in[n]}\right)$$

  Problem: $\widehat{f}$ may overfit to training data $\rightsquigarrow$ undercoverage

- Use a holdout set?

$$q = \text{Quantile}_{1-\alpha}\left(\{|Y_i - \widehat{f}(X_i)|\}_{i=n+1,\ldots,n+m}\right)$$

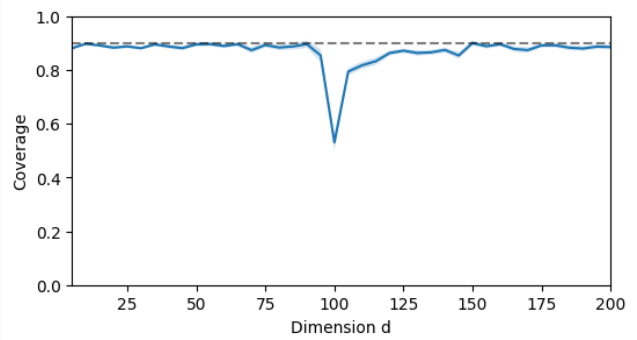
  Problem: cost of additional samples (or, cost of data splitting)

Distribution-free predictive inference

- Use cross-validation?

$$q_{\text{CV}} = \text{Quantile}_{1-\alpha}\left(\{|Y_i - \widehat{f}_{-i}(X_i)|\}_{i\in[n]}\right)$$

Problem: coverage guarantees hold asymptotically,
but may fail in finite samples under instability



A guarantee via stability

For an *unknown* data distribution P—

- We do not want to assume $\mathcal{A}$ is accurate for modeling P
- Sufficient to assume $\mathcal{A}$ is stable w.r.t. data drawn from P?

A guarantee via stability

For an *unknown* data distribution P—

- We do not want to assume $\mathcal{A}$ is accurate for modeling P
- Sufficient to assume $\mathcal{A}$ is stable w.r.t. data drawn from P?

Theorem

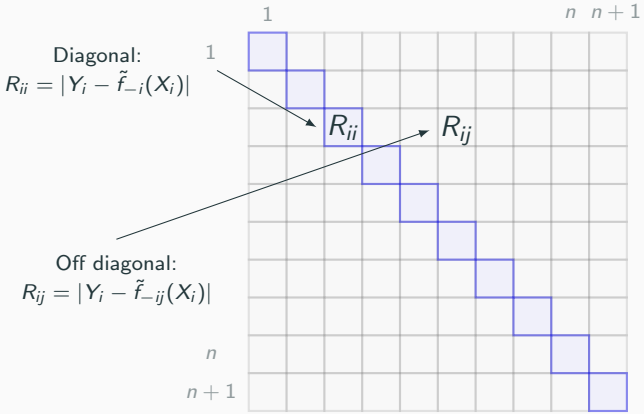
Suppose the data points are i.i.d., and $\mathcal{A}$ is β-ℓ_2-stable. Then

$$\mathbb{P}\left\{ Y_{n+1} \in \widehat{f}(X_{n+1}) \pm (q_{\mathrm{CV}} + \epsilon) \right\} \geq 1 - \alpha - \left(2\beta/\epsilon + \frac{1}{n+1} \right)$$

Proof idea

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

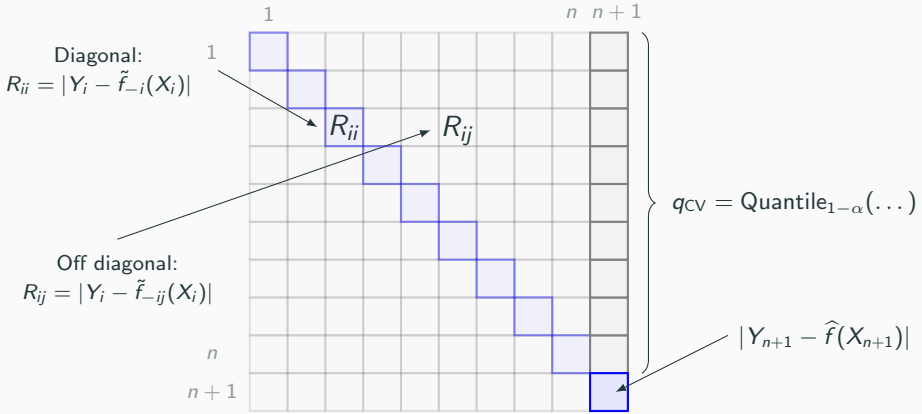
Define a matrix of residuals R:



Proof idea

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

Define a matrix of residuals R:



Proof idea

- By definition, coverage holds if

$$R_{n+1,n+1} \leq q_{\mathsf{CV}} + \epsilon \quad \text{where} \quad q_{\mathsf{CV}} = Q_{1-\alpha}(\{R_{i,n+1}\}_{i \in [n]})$$

Proof idea

- By definition, coverage holds if

$$R_{n+1,n+1} \leq q_{\mathsf{CV}} + \epsilon \quad \text{where} \quad q_{\mathsf{CV}} = Q_{1-\alpha}(\{R_{i,n+1}\}_{i\in[n]})$$

- Since the data is i.i.d.,

$$\mathbb{P}\left\{R_{n+1,n+1} \leq Q_{1-\alpha}(\{R_{ii}\}_{i\in[n]})\right\} \approx 1-\alpha.$$

Proof idea

- By definition, coverage holds if

$$R_{n+1,n+1} \leq q_{\mathrm{CV}} + \epsilon \quad \text{where} \quad q_{\mathrm{CV}} = Q_{1-\alpha}(\{R_{i,n+1}\}_{i\in[n]})$$

- Since the data is i.i.d.,

$$\mathbb{P}\left\{R_{n+1,n+1} \leq Q_{1-\alpha}(\{R_{ii}\}_{i\in[n]})\right\} \approx 1-\alpha.$$

- By stability of $\mathcal{A}$,

$$\mathbb{E}\left[(R_{ii} - R_{ij})^2\right] \leq \beta^2 \text{ for all } i \neq j$$

and so $Q_{1-\alpha}(\{R_{i,n+1}\}_{i\in[n]}) \approx Q_{1-\alpha}(\{R_{ii}\}_{i\in[n]})$

Implications

CV prediction interval has valid coverage if $\mathcal{A}$ is stable —
does not rely on assuming $\mathcal{A}$ is accurate (avoid model assumptions)

But, not every $\mathcal{A}$ is stable...

Implications

CV prediction interval has valid coverage if $\mathcal{A}$ is stable —
does not rely on assuming $\mathcal{A}$ is accurate (avoid model assumptions)

But, not every $\mathcal{A}$ is stable...

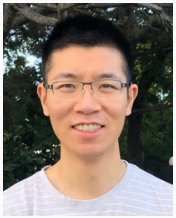
- Choose an $\mathcal{A}$ known to be stable?
- Use an alternative method? e.g., conformal prediction[7], jackknife+, ...
- Or, can we verify the stability of $\mathcal{A}$?

[7] Vovk et al 2005

Part 2: Hardness of testing stability

Black-box tests for algorithmic stability, Kim & B.
Is algorithmic stability testable? a unified framework under computational constraints, Luo & B.

Byol Kim
Sookmyung Women's University



Yuetian Luo
Rutgers

Testing stability

Key question

Is there a test that can empirically verify whether $\mathcal{A}$ is stable,
for complex black-box algorithms?

We want to construct a test $\widehat{T}(\mathcal{A}, \mathcal{D}, \xi)$ that:

($\mathcal{D}$: available data; ξ: random seed)

- Returns answer 1 if confident that $(\mathcal{A}, P, n)$ is stable
- Returns answer 0 otherwise

And, $\widehat{T}$ only has *black-box* access to the algorithm $\mathcal{A}$

Testing stability

In this part we will assume $\mathcal{A}$ treats training data symmetrically, and will work with a variant of ℓ_2-stability:

Definition: (ϵ, δ)-stability w.r.t. P

For data points $(X_1, Y_1), \ldots, (X_n, Y_n), (X, Y) \overset{\text{iid}}{\sim} P$,

$$\mathbb{P}_P \left\{ \left| \widehat{f}(X) - \widehat{f}_{-i}(X) \right| \geq \epsilon \right\} \leq \delta$$

Defining a black-box test

The black-box setting: learn how $\mathcal{A}$ works by running it on data

- Samples bootstrapped from available real data
- Semisynthetic data obtained by perturbing the real data
- Simulated data generated from a fitted model
- ...

Defining a black-box test

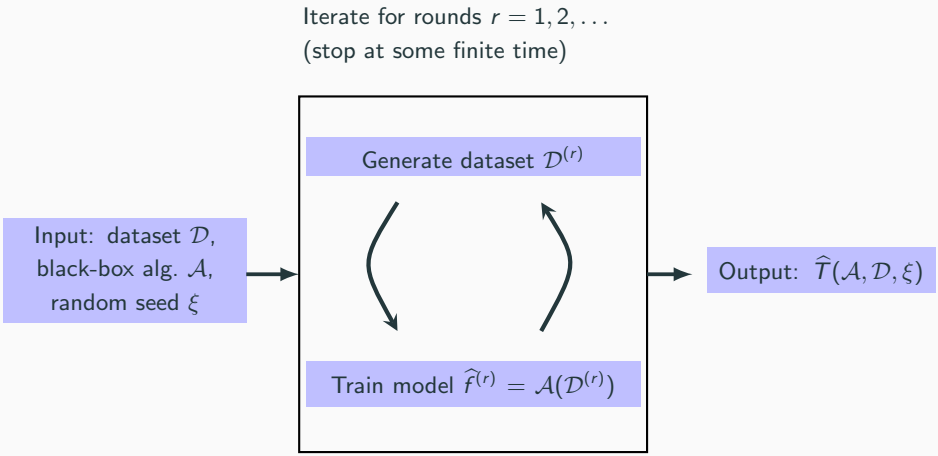
The black-box setting: learn how $\mathcal{A}$ works by running it on data

- Samples bootstrapped from available real data
- Semisynthetic data obtained by perturbing the real data
- Simulated data generated from a fitted model
- ...

But we cannot "look inside the black box" of $\mathcal{A}$, e.g.,

- Cannot compute $\sup_{x',y'} \left[\mathcal{A}(\mathcal{D} \cup \{(x', y')\})\right](x)$
- ...

Defining a black-box test



A hardness result

- Testing stability of $\mathcal{A}$ at sample size n
- Available data $\mathcal{D}$ contains N data points $\overset{\text{iid}}{\sim} P$
- Assume $|\mathcal{X} \times \mathcal{Y}| = \infty$ (to prevent exhaustive search)

A hardness result

- Testing stability of $\mathcal{A}$ at sample size n
- Available data $\mathcal{D}$ contains N data points $\overset{\text{iid}}{\sim} P$
- Assume $|\mathcal{X} \times \mathcal{Y}| = \infty$ (to prevent exhaustive search)

Theorem

Suppose $\widehat{T}$ offers distribution-free validity:

$$\mathbb{P}_P\left\{\widehat{T}(\mathcal{A}, \mathcal{D}, \xi) = 1\right\} \leq \alpha \quad \text{for all } \mathcal{A}, P \text{ that is } \textit{not } (\epsilon, \delta)\text{-stable}$$

Then the power of $\widehat{T}$ is bounded as:

$$\mathbb{P}_P\left\{\widehat{T}(\mathcal{A}, \mathcal{D}, \xi) = 1\right\} \leq \frac{\alpha}{(1-\delta)^{N/(n-1)}} \quad \text{for all } \mathcal{A}, P$$

A hardness result

Compare to a trivial solution (for $N = n + 1$):

- Let $\widehat{f} = \mathcal{A}(\{(X_i, Y_i)\}_{i\in[n]})$, $\widehat{f}_{-n} = \mathcal{A}(\{(X_i, Y_i)\}_{i\in[n-1]})$
- If $|\widehat{f}(X_{n+1}) - \widehat{f}_{-n}(X_{n+1})| < \epsilon$, return $\widehat{T} = 1$ with prob. $\frac{\alpha}{1-\delta}$
- Otherwise return $\widehat{T} = 0$

$\rightsquigarrow$ power $= \frac{\alpha}{1-\delta}$ for a ϵ-uniformly-stable $\mathcal{A}$

A hardness result

Compare to a trivial solution (for $N = n + 1$):

- Let $\widehat{f} = \mathcal{A}(\{(X_i, Y_i)\}_{i\in[n]})$, $\widehat{f}_{-n} = \mathcal{A}(\{(X_i, Y_i)\}_{i\in[n-1]})$
- If $|\widehat{f}(X_{n+1}) - \widehat{f}_{-n}(X_{n+1})| < \epsilon$, return $\widehat{T} = 1$ with prob. $\frac{\alpha}{1-\delta}$
- Otherwise return $\widehat{T} = 0$

$\rightsquigarrow$ power $= \frac{\alpha}{1-\delta}$ for a ϵ-uniformly-stable $\mathcal{A}$

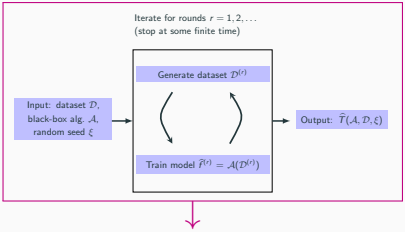
Compare to bound $\frac{\alpha}{(1-\delta)^{\frac{n+1}{n-1}}}$ via theorem

A hardness result

Interpretation:

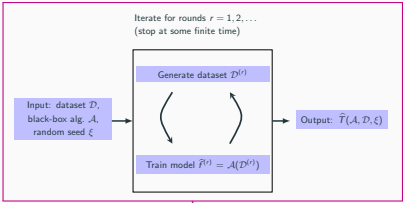
- *Every* valid test has low power: if $N = \mathcal{O}(n)$ and $\delta = o(1)$ then power $\approx \alpha$
- Can't improve on the power of a trivial solution
- No information can be gained from additional calls to $\mathcal{A}$, or from resampling/bootstrapping/simulating/modeling/etc

Proof idea



1. Find (x_*, y_*) with $\mathbb{P}_P\left\{(x_*, y_*) \in \mathcal{D} \cup \mathcal{D}^{(1)} \cup \mathcal{D}^{(2)} \cup \ldots\right\} \approx 0$

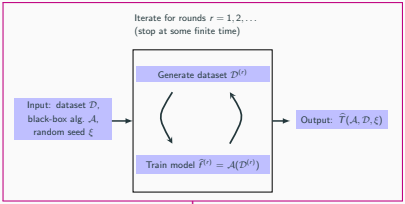
Proof idea



1. Find (x_*, y_*) with $\mathbb{P}_P\left\{(x_*, y_*) \in \mathcal{D} \cup \mathcal{D}^{(1)} \cup \mathcal{D}^{(2)} \cup \ldots\right\} \approx 0$
2. Define a corrupted distribution

$$P' = (1-c) \cdot P + c \cdot \delta_{(x_*, y_*)} \quad \text{with } c \approx 1 - (1-\delta)^{1/(n-1)}$$

Proof idea



1. Find (x_*, y_*) with $\mathbb{P}_P\left\{(x_*, y_*) \in \mathcal{D} \cup \mathcal{D}^{(1)} \cup \mathcal{D}^{(2)} \cup \ldots\right\} \approx 0$
2. Define a corrupted distribution

$$P' = (1-c) \cdot P + c \cdot \delta_{(x_*, y_*)} \quad \text{with } c \approx 1 - (1-\delta)^{1/(n-1)}$$

3. Define a corrupted algorithm

$$\mathcal{A}'(\mathcal{D}) = \begin{cases} \text{Return } x \mapsto 1, \text{ if } |\mathcal{D}| = n \text{ and } (x_*, y_*) \in \mathcal{D} \\ \text{Return } x \mapsto 0, \text{ if } |\mathcal{D}| = n-1 \text{ and } (x_*, y_*) \in \mathcal{D} \\ \text{Otherwise, run } \mathcal{A} \end{cases}$$

Proof idea

For data sampled $\overset{\text{iid}}{\sim} P$, $(x_*, y_*) \notin \mathcal{D} \cup \mathcal{D}^{(1)} \cup \mathcal{D}^{(2)} \cup \dots$ w.h.p.

$$\implies \mathbb{P}_P \left\{ \widehat{T}(\mathcal{A}, \mathcal{D}, \xi) = 1 \right\} \approx \mathbb{P}_P \left\{ \widehat{T}(\mathcal{A}', \mathcal{D}, \xi) = 1 \right\}$$

Proof idea

For data sampled $\overset{\text{iid}}{\sim} P$, $(x_*, y_*) \notin \mathcal{D} \cup \mathcal{D}^{(1)} \cup \mathcal{D}^{(2)} \cup \dots$ w.h.p.

$$\implies \mathbb{P}_P \left\{ \widehat{T}(\mathcal{A}, \mathcal{D}, \xi) = 1 \right\} \approx \mathbb{P}_P \left\{ \widehat{T}(\mathcal{A}', \mathcal{D}, \xi) = 1 \right\}$$

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

$$\mathbb{P}_{P'} \left\{ \widehat{T}(\mathcal{A}', \mathcal{D}, \xi) = 1 \right\} \geq \mathbb{P}_P \left\{ \widehat{T}(\mathcal{A}', \mathcal{D}, \xi) = 1 \right\} \cdot \underbrace{(1 - c)^N}_{\approx (1-\delta)^{N/(n-1)}}$$

Proof idea

For data sampled $\overset{\text{iid}}{\sim} P$, $(x_*, y_*) \notin \mathcal{D} \cup \mathcal{D}^{(1)} \cup \mathcal{D}^{(2)} \cup \dots$ w.h.p.

$$\implies \mathbb{P}_P\left\{\widehat{T}(\mathcal{A}, \mathcal{D}, \xi) = 1\right\} \approx \mathbb{P}_P\left\{\widehat{T}(\mathcal{A}', \mathcal{D}, \xi) = 1\right\}$$

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

$$\mathbb{P}_{P'}\left\{\widehat{T}(\mathcal{A}', \mathcal{D}, \xi) = 1\right\} \geq \mathbb{P}_P\left\{\widehat{T}(\mathcal{A}', \mathcal{D}, \xi) = 1\right\} \cdot \underbrace{(1-c)^N}_{\approx (1-\delta)^{N/(n-1)}}$$

For data sampled $\overset{\text{iid}}{\sim} P'$,

$$\mathcal{A}' \text{ is unstable with respect to } P' \implies \mathbb{P}_{P'}\left\{\widehat{T}(\mathcal{A}', \mathcal{D}, \xi) = 1\right\} \leq \alpha$$

Part 3: Stability guarantees via bagging

Bagging provides assumption-free stability, Soloff, B., Willett
Stability via resampling: statistical problems beyond the real line, Soloff, B., Willett

Jake Soloff
U. Michigan

Rebecca Willett
U. Chicago

Key question

Since algorithmic stability can't be certified empirically
(and we don't want to always use simple algorithms like k-NN) .....

Key question

Is there a way to convert any algorithm into a stable algorithm,
while still retaining its good performance?

Key question

Since algorithmic stability can't be certified empirically
(and we don't want to always use simple algorithms like k-NN) .....

Key question

Is there a way to convert any algorithm into a stable algorithm,
while still retaining its good performance?

Desired behavior:

- If $\mathcal{A}$ is stable $\rightsquigarrow$ return a similar fitted model to $\widehat{f} = \mathcal{A}(...)$
- If $\mathcal{A}$ is unstable $\rightsquigarrow$ replace $\widehat{f}$ with a stable fitted model

Background on bagging

Empirically, bagging (& other ensembling procedures) have been observed to improve stability dramatically.

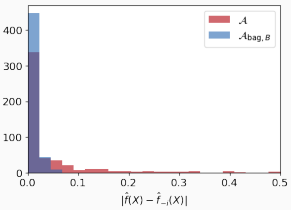
Bagging

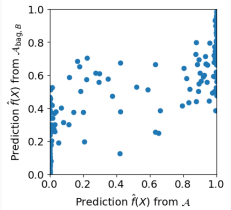
- Sample subsets of data ("bags") r_b for $b = 1, \ldots, B$
- Fit models $\widehat{f}_b = \mathcal{A}((X_i, Y_i)_{i \in r_b})$
- $\mathcal{A}_{\text{bag},B}$ returns aggregated model $\widehat{f}$:

$$\widehat{f}(x) := \frac{1}{B} \sum_b \widehat{f}_b(x)$$

Background on bagging

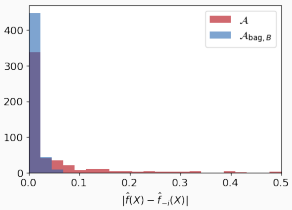
Logistic regression with $n = 500$, $d = 200$

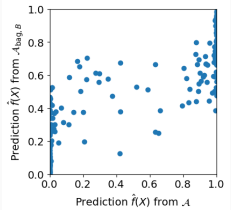




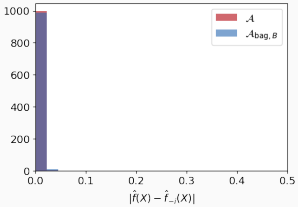
Background on bagging

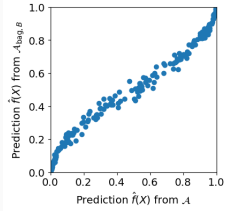
Logistic regression with $n = 500$, $d = 200$





Logistic regression with $n = 1000$, $d = 200$





Background on bagging

Bagging is widely used in statistics—
Random forests, variable selection, robust Bayesian inference, ...[8]

Many theoretical results—
Bagging induces smoothness, reduces variance, creates robustness[9]

[8] Breiman 2001; Meinshausen & Bühlmann 2010; Huggins &Miller 2023

[9] Bühlmann & Yu 2002; Grandvalet 2004; Friedman & Hall 2000; Elisseeff et al. 2005

Theoretical guarantees

Constructing the bags — common options:[10]

- Bootstrapping ("bagging") = subsets r_b of size m,
  sampled uniformly with replacement

$$p = \mathbb{P}\{i \in r_b\} = 1 - (1 - 1/n)^m$$

- Subbagging = subsets r_b of size $m < n$,
  sampled uniformly without replacement

$$p = \mathbb{P}\{i \in r_b\} = m/n$$

[10] Breiman 1996; Andonova et al. 2002

Theoretical guarantees

Theorem

Let $\mathcal{A}$ be any algorithm, returning functions with outputs in $[-1, 1]$.

Then its bagged version $\mathcal{A}_{\text{bag},B}$ satisfies

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

for any data points $(X_1, Y_1), \ldots, (X_n, Y_n), X$.

Theoretical guarantees

Theorem

Let $\mathcal{A}$ be any algorithm, returning functions with outputs in $[-1, 1]$.

Then its bagged version $\mathcal{A}_{\mathsf{bag},B}$ satisfies

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

for any data points $(X_1, Y_1), \ldots, (X_n, Y_n), X$.

- Implies β-ℓ_2-stability

Theoretical guarantees

Theorem

Let $\mathcal{A}$ be any algorithm, returning functions with outputs in $[-1, 1]$.

Then its bagged version $\mathcal{A}_{\mathsf{bag},B}$ satisfies

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

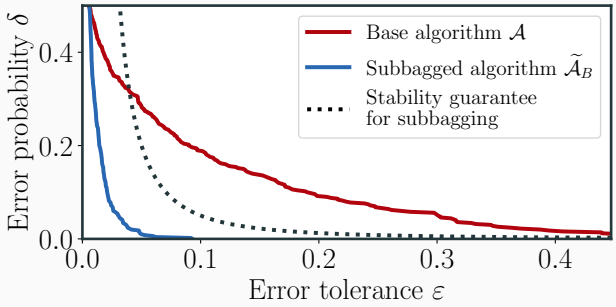
for any data points $(X_1, Y_1), \ldots, (X_n, Y_n), X$.

- Implies β-ℓ_2-stability
- Another interpretation: for $\delta = \beta^2/\epsilon^2$,

$$\left|\widehat{f}(X) - \widehat{f}_{-i}(X)\right| \geq \epsilon \text{ for at most } \delta n \text{ many } i\text{'s}$$

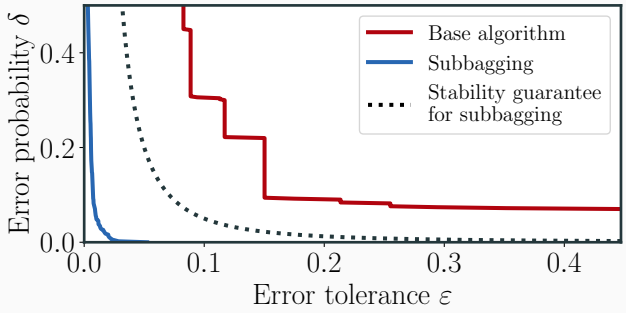
Empirical results

Logistic regression ($d = 200$, $n = 500$, $m = n/2$, $B = 10000$)



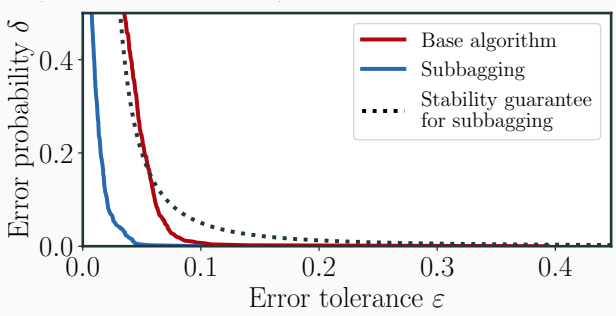
Empirical results

Regression trees ($d = 40$, $n = 500$, $m = n/2$, $B = 10000$)



Empirical results

Neural networks ($d = 200$, $n = 500$, $m = n/2$, $B = 10000$)



Proof idea

Let $\mathbf{A}$ be the adjacency matrix of data points & bags:

$$\mathbf{A}_{ir} = \mathbb{1}\{i \in r\}.$$

Let $\mathbf{B}$ be a centered and rescaled version:

$$\mathbf{B}_{ir} = \frac{\mathbf{A}_{ir} - p}{1 - p}.$$

Proof idea

Let $\mathbf{A}$ be the adjacency matrix of data points & bags:

$$\mathbf{A}_{ir} = \mathbb{1}\{i \in r\}.$$

Let $\mathbf{B}$ be a centered and rescaled version:

$$\mathbf{B}_{ir} = \frac{\mathbf{A}_{ir} - p}{1 - p}.$$

- $\mathbf{B}$ maps bagged predictors to leave-one-out differences:

$$\widehat{f}(X) - \widehat{f}_{-i}(X) = \mathbb{E}_r \left[\mathbf{B}_{ir} \widehat{f}_r(X) \right].$$

- Bound norm of $\mathbf{B}$ $\implies$ stability guarantee

Interpretation

- Bagging stabilizes any (bounded) $\mathcal{A}$ on any $\mathcal{D}$
- And, if $p \approx 1$ (e.g., each bag deletes $\mathcal{O}(n^{1/2})$ data points), we expect $\mathcal{A}_{\text{bag},B} \approx \mathcal{A}$ for 'nice' algorithms

Interpretation

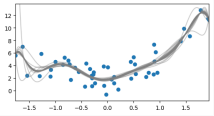
- Bagging stabilizes any (bounded) $\mathcal{A}$ on any $\mathcal{D}$
- And, if $p \approx 1$ (e.g., each bag deletes $\mathcal{O}(n^{1/2})$ data points),
  we expect $\mathcal{A}_{\text{bag},B} \approx \mathcal{A}$ for 'nice' algorithms

- Extensions to stability guarantees when $\mathcal{A}$ returns high-dimensional output, discrete output, etc[11]

Open question: is the computational cost of bagging unavoidable?

[11] Soloff et al. 2024, Liang et al. 2025

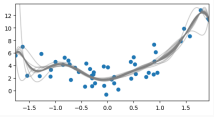
Summary



Part 1: Predictive inference via cross-validation

$\rightsquigarrow$ stability assumption replaces model assumptions

Summary



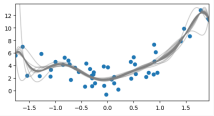
Part 1: Predictive inference via cross-validation

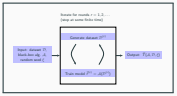
$\rightsquigarrow$ stability assumption replaces model assumptions

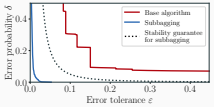
Part 2: Hardness of testing stability

$\rightsquigarrow$ cannot certify stability without other assumptions

Summary







Part 1: Predictive inference via cross-validation

$\rightsquigarrow$ stability assumption replaces model assumptions

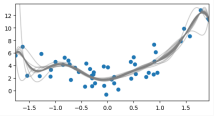
Part 2: Hardness of testing stability

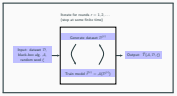
$\rightsquigarrow$ cannot certify stability without other assumptions

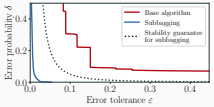
Part 3: Stability guarantees via bagging

$\rightsquigarrow$ any $\mathcal{A}$ becomes stable after bagging

Summary







Part 1: Predictive inference via cross-validation

$\rightsquigarrow$ stability assumption replaces model assumptions

Part 2: Hardness of testing stability

$\rightsquigarrow$ cannot certify stability without other assumptions

Part 3: Stability guarantees via bagging

$\rightsquigarrow$ any $\mathcal{A}$ becomes stable after bagging

---

Thank you!