Stability of Black Box Algorithms

Rina Foygel Barber (joint w/ Byol Kim, Jake Soloff, Rebecca Willett)

http://rinafb.github.io/

Outline

- Background on algorithmic stability
- Part 1: hardness of testing stability[1]
- Part 2: stability for bagged algorithms[2]

Collaborators:

Byol Kim

Part 1

Jake Soloff

Rebecca Willett

Part 2

[1] Kim & B., Black-box tests for algorithmic stability, arXiv:2111.15546

[2] Soloff, B., & Willett, Bagging provides assumption-free stability, arxiv:2301.12600

Background: algorithmic stability

$$\text{Data } (X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \mathbb{R} \quad \xrightarrow{\text{algorithm } \mathcal{A}} \quad \text{Fitted model } \widehat{f}$$

- Can $\widehat{f}$ estimate the true model for $Y \mid X$?
- Is $\widehat{f}$ guaranteed to predict Y with low test error?
- Is training error $\approx$ test error?
- Can we estimate how well $\widehat{f}$ predicts Y from X?

more assumptions

distribution-free?

Background: algorithmic stability

Concentration / consistency: $\widehat{f} \approx \widehat{f}'$ if we resample entire data set

Stability: $\widehat{f} \approx \widehat{f}'$ if we resample small fraction of data set

Background: algorithmic stability

Definition

$\mathcal{A}$ is (ϵ, δ)-stable with respect to distribution P & sample size n if

$$\mathbb{P}_P\left\{ \left|\widehat{f}(X_{n+1}) - \widehat{f}^{\setminus i}(X_{n+1})\right| > \epsilon \right\} \leq \delta \text{ for } (X_j, Y_j) \stackrel{\text{iid}}{\sim} P$$

$\mathcal{A}$ trained on $\{(X_j, Y_j); j \in [n]\}$ $\mathcal{A}$ trained on $\{(X_j, Y_j) : j \in [n]\backslash i\}$

Notes:

- We assume $\mathcal{A}$ treats training data symmetrically

$$\mathcal{A}\big((X_1, Y_1), \ldots, (X_n, Y_n)\big) = \mathcal{A}\big((X_{\sigma(1)}, Y_{\sigma(1)}), \ldots, (X_{\sigma(n)}, Y_{\sigma(n)})\big)$$

- Framework & results allow for a randomized $\mathcal{A}$

Motivation for algorithmic stability

Stability has implications for:

- Generalization [Bousquet & Elisseeff 2002; Elisseeff et al 2005]
- Learnability [Shalev-Shwartz et al 2010]
- Predictive inference
  [Steinberger & Leeb 2018; B., Candès, Ramdas, Tibshirani 2021]

Motivation for algorithmic stability — generalization

After training a model $\widehat{f} = \mathcal{A}\big((X_i, Y_i)_{i\in[n]}\big)$...

- Want to estimate $L(\widehat{f})$, where $L(f) = \mathbb{E}\left[\ell(f(X), Y)\right]$
- Leave-one-out estimate $L_{\text{LOO}}(\widehat{f}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\widehat{f}^{\setminus i}(X_i), Y_i)$

Motivation for algorithmic stability — generalization

After training a model $\widehat{f} = \mathcal{A}\big((X_i, Y_i)_{i\in[n]}\big)$...

- Want to estimate $L(\widehat{f})$, where $L(f) = \mathbb{E}\left[\ell(f(X), Y)\right]$
- Leave-one-out estimate $L_{\mathrm{LOO}}(\widehat{f}) = \frac{1}{n}\sum_{i=1}^n \ell(\widehat{f}^{\setminus i}(X_i), Y_i)$

Stability leads to generalization: [Bousquet & Elisseeff 2002]

- If ℓ is bounded and $\mathcal{A}$ satisfies

$$\mathbb{E}\left[\left|\ell(\widehat{f}(X_{n+1}), Y_{n+1}) - \ell(\widehat{f}^{\setminus i}(X_{n+1}), Y_{n+1})\right|\right] \leq \epsilon$$

  then

$$L(\widehat{f}) \leq L_{\mathrm{LOO}}(\widehat{f}) + \mathcal{O}_P\left(n^{-1/2} + \epsilon^{1/2}\right).$$

Motivation for algorithmic stability — predictive inference

Definition: distribution-free predictive set

A map from data $(X_i, Y_i)_{i\in[n]}$ to a prediction band $\widehat{C}_n$ s.t.

$$\mathbb{P}_{(X_i,Y_i)\overset{\mathrm{iid}}{\sim}P}\left\{Y_{n+1}\in\widehat{C}_n(X_{n+1})\right\}\geq 1-\alpha$$

for <u>every</u> distribution P.

Motivation for algorithmic stability — predictive inference

Definition: distribution-free predictive set

A map from data $(X_i, Y_i)_{i\in[n]}$ to a prediction band $\widehat{C}_n$ s.t.

$$\mathbb{P}_{(X_i,Y_i)\stackrel{\mathrm{iid}}{\sim}P}\left\{Y_{n+1}\in\widehat{C}_n(X_{n+1})\right\}\geq 1-\alpha$$

for <u>every</u> distribution P.

Methods:

- Conformal prediction — high computational cost [Vovk et al 2005]
- Split conformal (i.e., holdout set) — less precise b/c split data
- Jackknife a.k.a. leave-one-out cross-validation — is it distrib.-free?

Motivation for algorithmic stability — predictive inference

Jackknife: fix any regression algorithm $\mathcal{A}$, then compute

$$\widehat{f} = \mathcal{A}\big((X_i, Y_i)_{i\in[n]}\big), \quad \widehat{f}^{\setminus i} = \mathcal{A}\big((X_j, Y_j)_{j\in[n]\setminus\{i\}}\big)$$

Prediction interval for Y_{n+1} given $X_{n+1} = x$:

$$\widehat{C}_n(x) = \widehat{f}(x) \pm Q_{1-\alpha}(R_i)$$

where $R_i = |Y_i - \widehat{f}^{\setminus i}(X_i)| =$ ith leave-one-out residual

Motivation for algorithmic stability — predictive inference

Jackknife: fix any regression algorithm $\mathcal{A}$, then compute

$$\widehat{f} = \mathcal{A}\big((X_i, Y_i)_{i\in[n]}\big), \quad \widehat{f}^{\setminus i} = \mathcal{A}\big((X_j, Y_j)_{j\in[n]\setminus\{i\}}\big)$$

Prediction interval for Y_{n+1} given $X_{n+1} = x$:

$$\widehat{C}_n(x) = \widehat{f}(x) \pm Q_{1-\alpha}(R_i)$$

where $R_i = |Y_i - \widehat{f}^{\setminus i}(X_i)| =$ ith leave-one-out residual

Is this method distribution-free?

- No assumption-free guarantees — $\widehat{f}$ & $\widehat{f}^{\setminus i}$ may behave differently
- If $\mathcal{A}$ is (ϵ, δ)-stable, guarantees w/o any assumptions on P
  [Steinberger & Leeb 2018; B., Candès, Ramdas, Tibshirani 2021]

Motivation for algorithmic stability

At a high level...
We want methods that are valid with no untestable assumptions

- We can't test whether P satisfies distributional assumptions (e.g., parametric model / smoothness / etc)
- Some robust methods (e.g., jackknife) instead assume $\mathcal{A}$ is stable
- But, is this another untestable assumption?

Aren't most algorithms stable?

Some algorithms are known to satisfy stability:

- Nearest neighbors:

$$\widehat{f}(x) = \frac{1}{k} \sum_{i \in k\text{-NN}(x)} Y_i$$

  Stable if we choose $k \ll n$

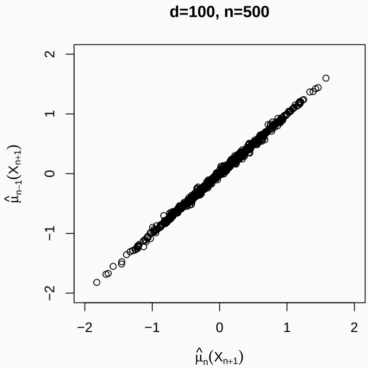
- Ridge regression:

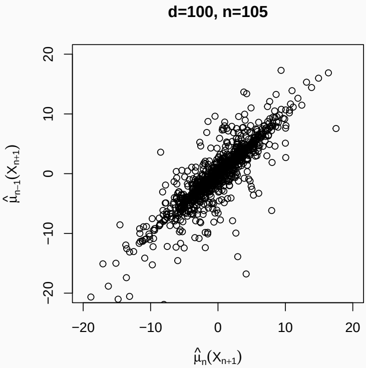
$$\widehat{f} = \phi(x; \widehat{\beta}_n) \text{ where } \widehat{\beta}_n = \arg\min_{\beta} \left\{ \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i; \phi(X_i; \beta)) + \lambda \|\beta\|_2^2 \right\}$$

  Stable if f and ℓ are Lipschitz

Aren't most algorithms stable?

Exhibit A: least squares – known to be unstable if $d \approx n$





Aren't most algorithms stable?

Exhibit B: modern ML methods – too complex to prove stability

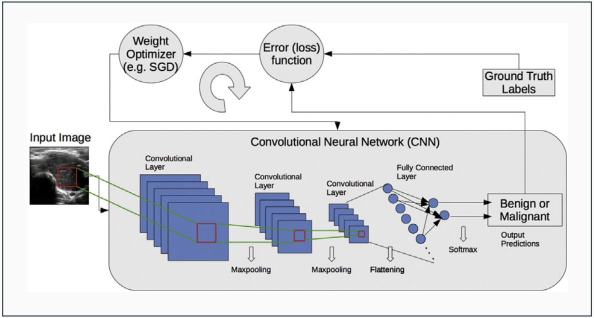


Figure from A Survey of Deep-Learning Applications in Ultrasound: Artificial Intelligence-Powered Ultrasound for Improving Clinical Workflow, Akkus et al 2019

Aren't most algorithms stable?

Exhibit C: some methods have instability built in

- Lasso (`glmnet` R package):
  `glmnet(x,y,..., lambda.min.ratio = ifelse(nobs < nvars,0.01,1e-04) ,...)`
- Highly adaptive Lasso (`hal9001` R package):
  `SL.hal(Y,X,..., nfolds = ifelse(length(Y) <= 100,20,10) ,...)`
- Multiple imputation (`mice` & `midastouch` R package):
  In an old version of the code: `outout <- ifelse(nobs>250,FALSE,TRUE)`
- ...

Part 1: Testing stability in the black box setting

The black box setting: we learn how $\mathcal{A}$ works by running it on data, e.g.:

- Run $\mathcal{A}$ on samples bootstrapped from available real data
- Run $\mathcal{A}$ on semisynthetic data obtained by perturbing the real data
- Run $\mathcal{A}$ on simulated data obtained by fitting a model to real data
- Etc.

Part 1: Testing stability in the black box setting

The black box setting: we learn how $\mathcal{A}$ works by running it on data, e.g.:

- Run $\mathcal{A}$ on samples bootstrapped from available real data
- Run $\mathcal{A}$ on semisynthetic data obtained by perturbing the real data
- Run $\mathcal{A}$ on simulated data obtained by fitting a model to real data
- Etc.

But, we cannot "look inside the black box" of $\mathcal{A}$ or of a fitted $\widehat{f}$:

- Cannot compute $\sup_{(x',y')} \left|[\mathcal{A}(\mathcal{D})](x) - [\mathcal{A}(\mathcal{D} \cup (x',y'))](x)\right|$
- Cannot check if $\widehat{f} = \mathcal{A}(\mathcal{D})$ is Lipschitz
- Etc.

Part 1: Testing stability in the black box setting

We want to construct a test $\widehat{T} = \widehat{T}_{n,\epsilon,\delta}(\mathcal{A}, \underbrace{\mathcal{D}_\ell, \mathcal{D}_u}_{\text{available labeled \& unlabeled data}})$ that:

$$\begin{cases} \text{Returns 1 if we are confident that } (\mathcal{A}, P, n) \text{ is } (\epsilon, \delta)\text{-stable} \\ \text{Returns 0 otherwise} \end{cases}$$

Part 1: Testing stability in the black box setting

available labeled & unlabeled data

We want to construct a test $\widehat{T} = \widehat{T}_{n,\epsilon,\delta}(\mathcal{A}, \mathcal{D}_\ell, \mathcal{D}_u)$ that:

$$\begin{cases} \text{Returns 1 if we are confident that } (\mathcal{A}, P, n) \text{ is } (\epsilon,\delta)\text{-stable} \\ \text{Returns 0 otherwise} \end{cases}$$

- We require $\widehat{T}$ to be a valid distribution-free test of (ϵ, δ)-stability:

$$\mathbb{P}\left\{ \widehat{T}_{n,\epsilon,\delta}(\mathcal{A}, \mathcal{D}_\ell, \mathcal{D}_u) = 1 \right\} \leq \alpha \text{ for any } (\mathcal{A}, P, n) \text{ that is } \underline{\text{not}}\ (\epsilon,\delta)\text{-stable}$$

with respect to data $\mathcal{D}_\ell, \mathcal{D}_u$ drawn i.i.d. from P

- We want $\widehat{T}$ to have high power for detecting stability:

$$\mathbb{P}\left\{ \widehat{T}_{n,\epsilon,\delta}(\mathcal{A}, \mathcal{D}_\ell, \mathcal{D}_u) = 1 \right\} \overset{???}{\gg} \alpha \text{ for } (\epsilon,\delta)\text{-stable } (\mathcal{A}, P, n)$$

Black-box tests

Definition: black-box test

available labeled & unlabeled data

$\widehat{T} = \widehat{T}(\mathcal{A}, \mathcal{D}_\ell, \mathcal{D}_u)$ is a black-box test if it can be defined as follows:

Black-box tests

Definition: black-box test

available labeled & unlabeled data

$\widehat{T} = \widehat{T}(\mathcal{A}, \mathcal{D}_\ell, \mathcal{D}_u)$ is a black-box test if it can be defined as follows:

- At step $r = 1$, generate a new dataset (e.g., via subsampling/bootstrap/simulation)

$$(\mathcal{D}_\ell^{(1)}, \mathcal{D}_u^{(1)}) = f^{(1)}\big[\mathcal{D}_\ell, \mathcal{D}_u\big],$$

  and train and evaluate the model,

$$\widehat{f}^{(1)} = \mathcal{A}\big(\mathcal{D}_\ell^{(1)}\big), \quad \widehat{\mathbf{Y}}^{(1)} = \widehat{f}^{(1)}\big(\mathcal{D}_u^{(1)}\big).$$

Black-box tests

Definition: black-box test

available labeled & unlabeled data

$\widehat{T} = \widehat{T}(\mathcal{A}, \mathcal{D}_\ell, \mathcal{D}_u)$ is a black-box test if it can be defined as follows:

- At step $r = 1$, generate a new dataset (e.g., via subsampling/bootstrap/simulation)

$$(\mathcal{D}_\ell^{(1)}, \mathcal{D}_u^{(1)}) = f^{(1)}\big[\mathcal{D}_\ell, \mathcal{D}_u\big],$$

  and train and evaluate the model,

$$\widehat{f}^{(1)} = \mathcal{A}\big(\mathcal{D}_\ell^{(1)}\big), \quad \widehat{\mathbf{Y}}^{(1)} = \widehat{f}^{(1)}\big(\mathcal{D}_u^{(1)}\big).$$

- At step $r = 2$, generate a new dataset

$$(\mathcal{D}_\ell^{(2)}, \mathcal{D}_u^{(2)}) = f^{(2)}\big[\mathcal{D}_\ell, \mathcal{D}_u, \mathcal{D}_\ell^{(1)}, \mathcal{D}_u^{(1)}, \widehat{\mathbf{Y}}^{(1)}\big],$$

  and train and evaluate the model,

$$\widehat{f}^{(2)} = \mathcal{A}\big(\mathcal{D}_\ell^{(2)}\big), \quad \widehat{\mathbf{Y}}^{(2)} = \widehat{f}^{(2)}\big(\mathcal{D}_u^{(2)}\big).$$

Black-box tests

Definition: black-box test

available labeled & unlabeled data

$\widehat{T} = \widehat{T}(\mathcal{A}, \mathcal{D}_\ell, \mathcal{D}_u)$ is a black-box test if it can be defined as follows:

- At step $r = 1$, generate a new dataset (e.g., via subsampling/bootstrap/simulation)

$$(\mathcal{D}_\ell^{(1)}, \mathcal{D}_u^{(1)}) = f^{(1)}\big[\mathcal{D}_\ell, \mathcal{D}_u\big],$$

  and train and evaluate the model,

$$\widehat{f}^{(1)} = \mathcal{A}\big(\mathcal{D}_\ell^{(1)}\big), \quad \widehat{\mathbf{Y}}^{(1)} = \widehat{f}^{(1)}\big(\mathcal{D}_u^{(1)}\big).$$

- At step $r = 2$, generate a new dataset

$$(\mathcal{D}_\ell^{(2)}, \mathcal{D}_u^{(2)}) = f^{(2)}\big[\mathcal{D}_\ell, \mathcal{D}_u, \mathcal{D}_\ell^{(1)}, \mathcal{D}_u^{(1)}, \widehat{\mathbf{Y}}^{(1)}\big],$$

  and train and evaluate the model,

$$\widehat{f}^{(2)} = \mathcal{A}\big(\mathcal{D}_\ell^{(2)}\big), \quad \widehat{\mathbf{Y}}^{(2)} = \widehat{f}^{(2)}\big(\mathcal{D}_u^{(2)}\big).$$

- Repeat for $r = 3, 4, \ldots$

Definition: black-box test

available labeled & unlabeled data

$\widehat{T} = \widehat{T}(\mathcal{A}, \mathcal{D}_\ell, \mathcal{D}_u)$ is a black-box test if it can be defined as follows:

- At step $r = 1$, generate a new dataset (e.g., via subsampling/bootstrap/simulation)

$$(\mathcal{D}_\ell^{(1)}, \mathcal{D}_u^{(1)}) = f^{(1)}\big[\mathcal{D}_\ell, \mathcal{D}_u\big],$$

  and train and evaluate the model,

$$\widehat{f}^{(1)} = \mathcal{A}\big(\mathcal{D}_\ell^{(1)}\big), \quad \widehat{\mathbf{Y}}^{(1)} = \widehat{f}^{(1)}\big(\mathcal{D}_u^{(1)}\big).$$

- At step $r = 2$, generate a new dataset

$$(\mathcal{D}_\ell^{(2)}, \mathcal{D}_u^{(2)}) = f^{(2)}\big[\mathcal{D}_\ell, \mathcal{D}_u, \mathcal{D}_\ell^{(1)}, \mathcal{D}_u^{(1)}, \widehat{\mathbf{Y}}^{(1)}\big],$$

  and train and evaluate the model,

$$\widehat{f}^{(2)} = \mathcal{A}\big(\mathcal{D}_\ell^{(2)}\big), \quad \widehat{\mathbf{Y}}^{(2)} = \widehat{f}^{(2)}\big(\mathcal{D}_u^{(2)}\big).$$

- Repeat for $r = 3, 4, \ldots$
- Finally, define $\widehat{T} = g\big[\mathcal{D}_\ell, \mathcal{D}_u, (\mathcal{D}_\ell^{(r)})_{r\geq 1}, (\mathcal{D}_u^{(r)})_{r\geq 1}, (\widehat{\mathbf{Y}}^{(r)})_{r\geq 1}\big]$.

Binomial test

Let $\kappa = \min\left\{\frac{|\mathcal{D}_\ell|}{n}, \frac{|\mathcal{D}_\ell|+|\mathcal{D}_u|}{n+1}\right\}$

$\rightsquigarrow$ can construct $\lfloor \kappa \rfloor$ many data sets $(X_1^k, Y_1^k), \ldots, (X_n^k, Y_n^k), X_{n+1}^k$

A simple binomial test

Binomial test

Let $\kappa = \min\left\{\frac{|\mathcal{D}_\ell|}{n}, \frac{|\mathcal{D}_\ell|+|\mathcal{D}_u|}{n+1}\right\}$

$\rightsquigarrow$ can construct $\lfloor\kappa\rfloor$ many data sets $(X_1^k, Y_1^k), \ldots, (X_n^k, Y_n^k), X_{n+1}^k$

A simple binomial test

- For each data set $k = 1, \ldots, \lfloor\kappa\rfloor$, fit models

$$\widehat{f}_k = \mathcal{A}\big((X_i^k, Y_i^k)_{i\in[n]}\big), \quad \widehat{f}_k^{\setminus n} = \mathcal{A}\big((X_i^k, Y_i^k)_{i\in[n-1]}\big)$$

& compare predictions:

$$\Delta_k = \left|\ \widehat{f}_k(X_{n+1}^k) - \widehat{f}_k^{\setminus n}(X_{n+1}^k)\ \right|$$

- Compare against Binom($\lfloor\kappa\rfloor$,δ) at level α:

$$\widehat{T} = \mathbf{1}\Big\{\ \textstyle\sum_k \mathbf{1}_{\Delta_k > \epsilon}\ \leq \text{the } \alpha\text{-quantile of Binom}(\lfloor\kappa\rfloor,\delta)\Big\}$$

with randomization to handle discreteness

Performance of the binomial test

Theorem: validity of the simple binomial test

If $(\mathcal{A}, P, n)$ is not (ϵ, δ)-stable, then

$$\mathbb{P}\left\{ \widehat{T} = 1 \right\} \leq \alpha.$$

Theorem: power of the simple binomial test

If $(\mathcal{A}, P, n)$ is (ϵ, δ)-stable, & either $\delta^*_\epsilon = 0$ or $\delta \leq 1 - \alpha^{1/\lfloor\kappa\rfloor}$,

$$\mathbb{P}\left\{ \widehat{T} = 1 \right\} = \left\{ \alpha \cdot \left(\frac{1 - \delta^*_\epsilon}{1 - \delta} \right)^{\lfloor\kappa\rfloor} \right\} \wedge 1$$

$\delta^*_\epsilon = \min\{\delta : (\mathcal{A}, P, n) \text{ is } (\epsilon, \delta)\text{-stable}\}$

Performance of the binomial test

- The binomial test has validity, but power is low
- Unsurprising b/c it doesn't make efficient use of the data—
  can we improve power by extracting more info from the data?

A hardness result

Recall $\kappa = \min\left\{\frac{|\mathcal{D}_\ell|}{n}, \frac{|\mathcal{D}_\ell|+|\mathcal{D}_u|}{n+1}\right\}$

Theorem: upper bound on power

Let $\widehat{T}$ be <u>any</u> black-box test of stability that is valid at level $\leq \alpha$.
If $(\mathcal{A}, P, n)$ is (ϵ, δ)-stable,

$$\mathbb{P}\left\{\widehat{T}=1\right\} \leq \left\{\alpha \cdot \left(\frac{1-\delta^*_\epsilon}{1-\delta}\right)^\kappa\right\} \wedge 1.$$

A hardness result

Interpretation:

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

Proof sketch

Suppose $(\mathcal{A}, P, n)$ is (ϵ, δ)-stable.

Proof idea: construct $(\mathcal{A}', P', n)$ that is not stable, such that:

- $P \approx P'$ so that $\mathrm{d}_{\mathrm{TV}}(\text{data from } P, \text{data from } P')$ is small
- And, if data $\sim P$, then $\mathcal{A}$ and $\mathcal{A}'$ return the same output
- So, $\mathbb{P}_{(\mathcal{A},P,n)}\left\{\widehat{T} = 1\right\} \approx \mathbb{P}_{(\mathcal{A}',P',n)}\left\{\widehat{T} = 1\right\} \leq \alpha$

Proof sketch

$$\mathbb{P}\left\{\widehat{T}=1\right\} \le \alpha \left(\frac{1-\delta^*_\epsilon}{1-\delta}\right)^{\frac{|\mathcal{D}_\ell|}{n}}$$

Distribution P': draw $(X, Y) \sim P$, then return

$$\begin{cases} (X, Y) & \text{with probability } 1-c \\ (X, y_*) & \text{with probability } c, \end{cases}$$

for a small constant $c > 0$

Proof sketch

$$\mathbb{P}\left\{\widehat{T}=1\right\} \leq \alpha \left(\frac{1-\delta_\epsilon^*}{1-\delta}\right)^{\frac{|\mathcal{D}_\ell|}{n}}$$

Distribution P': draw $(X, Y) \sim P$, then return

$$\begin{cases}(X, Y) & \text{with probability } 1-c \\ (X, y_*) & \text{with probability } c,\end{cases}$$

for a small constant $c > 0$

Algorithm $\mathcal{A}'$:
Given data $(x_1, y_1), \ldots, (x_m, y_m)$ & test point x_{m+1},

- If $m = n$ and $y_i = y_*$ for any i, return a corrupted prediction
- Otherwise, return $[\mathcal{A}\big((x_1, y_1), \ldots, (x_m, y_m)\big)](x_{m+1})$

Proof sketch

$$\mathbb{P}\left\{\widehat{T}=1\right\} \leq \alpha\left(\frac{1-\delta^*_\epsilon}{1-\delta}\right)^{\frac{|\mathcal{D}_\ell|}{n}}$$

Distribution P': draw $(X, Y) \sim P$, then return

$$\begin{cases}(X, Y) & \text{with probability } 1-c \\ (X, y_*) & \text{with probability } c,\end{cases}$$

for a small constant $c > 0$

choose s.t. $(\mathcal{A}', P', n)$ is not (ϵ, δ) stable

Algorithm $\mathcal{A}'$:
Given data $(x_1, y_1), \ldots, (x_m, y_m)$ & test point x_{m+1},

- If $m = n$ and $y_i = y_*$ for any i, return a corrupted prediction
- Otherwise, return $[\mathcal{A}\big((x_1, y_1), \ldots, (x_m, y_m)\big)](x_{m+1})$

Proof sketch

$$\mathbb{P}\left\{\widehat{T}=1\right\} \leq \alpha \left(\frac{1-\delta_\epsilon^*}{1-\delta}\right)^{\frac{|\mathcal{D}_\ell|}{n}} \quad \text{and} \quad \mathbb{P}\left\{\widehat{T}=1\right\} \leq \alpha \left(\frac{1-\delta_\epsilon^*}{1-\delta}\right)^{\frac{|\mathcal{D}_\ell|+|\mathcal{D}_u|}{n+1}}$$

Distribution P': draw $(X, Y) \sim P$, then return

$$\begin{cases} (X, Y) & \text{with probability } 1-c \\ \cancel{(X, y_*)}\ (x_*, Y) & \text{with probability } c, \end{cases}$$

for a small constant $c > 0$

choose s.t. $(\mathcal{A}', P', n)$ is not (ϵ, δ) stable

Algorithm $\mathcal{A}'$:
Given data $(x_1, y_1), \ldots, (x_m, y_m)$ & test point x_{m+1},

- If $m = n$ and $\overset{x_i = x_*}{\cancel{y_i = y_*}}$ for any i, return a corrupted prediction
- Otherwise, return $[\mathcal{A}((x_1, y_1), \ldots, (x_m, y_m))](x_{m+1})$

Part 1: summary & open questions

Our results:

- The simple binomial test uses the data very inefficiently,
  but can still be the most powerful distribution-free test of stability
- More sophisticated strategies (simulating/bootstrapping/etc)
  do not help to determine stability

Part 1: summary & open questions

Our results:

- The simple binomial test uses the data very inefficiently,
  but can still be the most powerful distribution-free test of stability
- More sophisticated strategies (simulating/bootstrapping/etc)
  do not help to determine stability

Open questions:

- Are there mild assumptions on $\mathcal{A}, P$ that make stability testable?
- Are we using a definition of stability that's too strong?
- Is there a way to convert any algorithm into a stable algorithm,
  with a pre- or post-processing step?

Part 2: Stability of bagged algorithms

Question inspired by Part 1

- Is there a way to convert any algorithm into a stable algorithm?

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

Part 2: Stability of bagged algorithms

Question inspired by Part 1

- Is there a way to convert any algorithm into a stable algorithm?

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

Bagging

- Sample subsets $S_b \subseteq [n]$ for $b = 1, \ldots, B$
- Fit models $\widehat{f}_b = \mathcal{A}((X_i, Y_i) : i \in S_b)$
- $\mathcal{A}_{\text{bag}}$ returns aggregated model $\widehat{f}$:

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

Background on bagging

The most common options:

- Classical bagging = subsets S_b of size m, sampled w/ replacement

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

- Subbagging = subsets S_b of size $m < n$, sampled w/o replacement

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

Background on bagging

Bagging appears in:

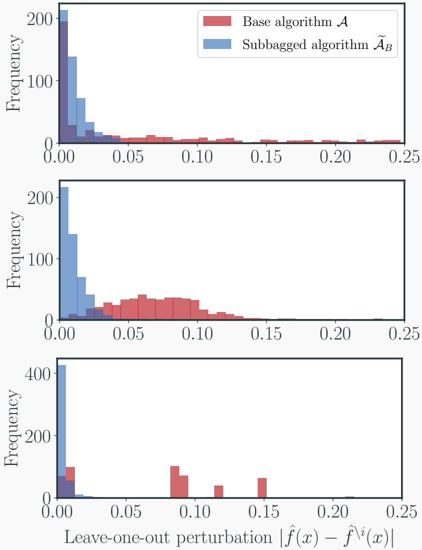
- Random forests [Breiman 2001]
- Variable selection in regression [Meinshausen & Bühlmann 2010]
- Classification in the presence of class imbalance [Li 2007]
- Robust Bayesian inference (BayesBag) [Huggins & Miller 2023]
- & many more

Background on bagging

Many results on theoretical properties—
Bagging induces smoothness, reduces variance, creates robustness
[Bühlmann&Yu 2002, Grandvalet 2004, Friedman&Hall 2000, Elisseeff et al 2005, . . .]

Some results show stability of bagging in limited regimes ($m \ll n$)
[Poggio et al 2002, Chen et al 2022]

Bagging helps stability



$\leftarrow$ logistic regression

$\leftarrow$ neural networks

$\leftarrow$ regression trees

Re-defining stability

Definition from Part 1

$\mathcal{A}$ is (ϵ, δ)-stable if

$$\mathbb{P}_P \left\{ \left| \widehat{f}(x) - \widehat{f}^{\setminus i}(x) \right| > \epsilon \right\} \leq \delta$$

when trained on $\mathcal{D} = (X_1, Y_1), \ldots, (X_n, Y_n) \stackrel{\text{iid}}{\sim} P$, and tested on $x \sim P_X$

Re-defining stability

Definition from Part 1

$\mathcal{A}$ is (ϵ, δ)-stable if

$$\mathbb{P}_P \left\{ \left| \widehat{f}(x) - \widehat{f}^{\setminus i}(x) \right| > \epsilon \right\} \leq \delta$$

when trained on $\mathcal{D} = (X_1, Y_1), \ldots, (X_n, Y_n) \stackrel{\text{iid}}{\sim} P$, and tested on $x \sim P_X$

Updated definition (strictly stronger)

$\mathcal{A}$ is (ϵ, δ)-stable if

$$\frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{ \left| \widehat{f}(x) - \widehat{f}^{\setminus i}(x) \right| > \epsilon \} \leq \delta$$

when trained on any $\mathcal{D} = (X_1, Y_1), \ldots, (X_n, Y_n)$, and tested on any x

Main result: stability guarantee

Recall $p = \mathbb{P}\{i \in S_b\} = \begin{cases} m/n & \text{for subbagging} \\ 1-(1-1/n)^m & \text{for classical bagging} \end{cases}$

Theorem

Let $\mathcal{A}$ be any base algorithm that returns predictions in $[0,1]$.
For any n, m, as $B \to \infty$, $\mathcal{A}_{\text{bag}}$ satisfies (ϵ, δ)-stability for every

$$\delta\epsilon^2 \geq \frac{1}{4(n-1)} \cdot \frac{p}{1-p}$$

Main result: stability guarantee

Recall $p = \mathbb{P}\{i \in S_b\} = \begin{cases} m/n & \text{for subbagging} \\ 1-(1-1/n)^m & \text{for classical bagging} \end{cases}$

Theorem

Let $\mathcal{A}$ be any base algorithm that returns predictions in $[0,1]$.
For any n, m, as $B \to \infty$, $\mathcal{A}_{\text{bag}}$ satisfies (ϵ, δ)-stability for every

$$\delta\epsilon^2 \geq \frac{1}{4(n-1)} \cdot \frac{p}{1-p}$$

- Framework & results allow for a randomized $\mathcal{A}$
- Results extend to finite B (use Hoeffding's inequality)
- Can also extend to models with unbounded output, via either clipping predictions, or letting ϵ adapt to the range of $\widehat{f}$

Main result: stability guarantee

Regimes for subbagging:

- **Proportional sampling:** $m = \mathcal{O}(n)$
  Stability is guaranteed for $\delta\epsilon^2 \gtrsim n^{-1}$
- **Massive subsampling:** $m = \mathcal{O}(n^\beta)$ **for** $0 < \beta < 1$
  Stability is guaranteed for $\delta\epsilon^2 \gtrsim n^{-(2-\beta)}$
  [See also existing stability guarantees by Chen, Syrgkanis, Austern 2022]
- **Minimal subsampling:** $m = n - \mathcal{O}(n^\beta)$ **for** $0 < \beta < 1$
  Stability is guaranteed for $\delta\epsilon^2 \gtrsim n^{-\beta}$

Proof sketch

Suppose $S \subset [n]$ contains δn "bad" data points:

$$|\widehat{f}(x) - \widehat{f}^{\setminus i}(x)| \geq \epsilon$$

where $\widehat{f} = \mathcal{A}_{\text{bag}}(\mathcal{D})$ averages over all bags S_b,
while $\widehat{f}^{\setminus i} = \mathcal{A}_{\text{bag}}(\mathcal{D}_{-i})$ averages over all bags $S_b \not\ni i$.

Proof sketch

Suppose $S \subset [n]$ contains δn "bad" data points:

$$|\widehat{f}(x) - \widehat{f}^{\setminus i}(x)| \geq \epsilon$$

where $\widehat{f} = \mathcal{A}_{\text{bag}}(\mathcal{D})$ averages over all bags S_b,
while $\widehat{f}^{\setminus i} = \mathcal{A}_{\text{bag}}(\mathcal{D}_{-i})$ averages over all bags $S_b \not\ni i$.

Proof idea: a double counting argument for $L = \sum_{i \in S} |\widehat{f}(x) - \widehat{f}^{\setminus i}(x)|$

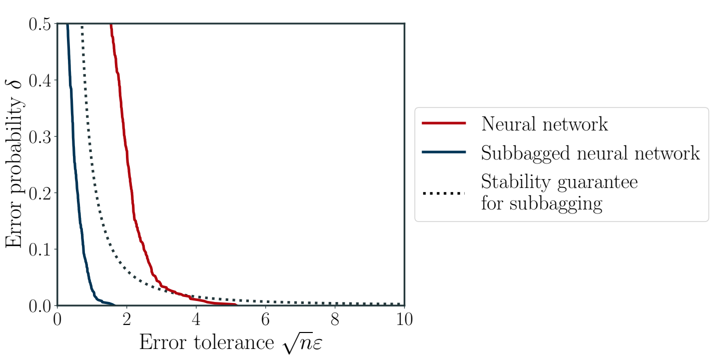
- Summing over i, we see $L \geq \epsilon \cdot |S| = \delta \epsilon n$
- We can also rewrite L by summing over bags b, because

$$\widehat{f}^{\setminus i}(x) = \mathbb{E}\left[\widehat{f}_b(x) \;\middle|\; i \notin S_b\right] = \frac{1}{1-p}\mathbb{E}\left[\widehat{f}_b(x)\mathbb{1}_{i \in S_b}\right]$$

where expectation is taken over a randomly drawn bag S_b.

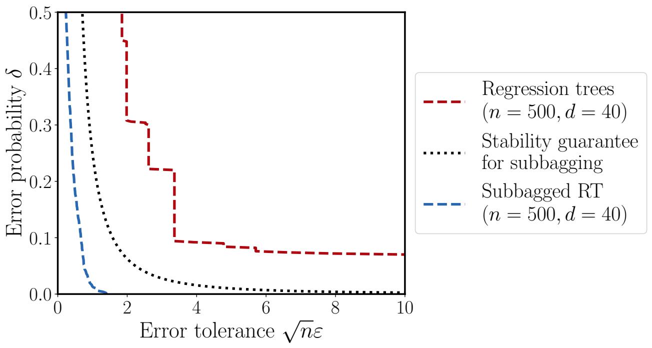
Empirical results

Neural networks



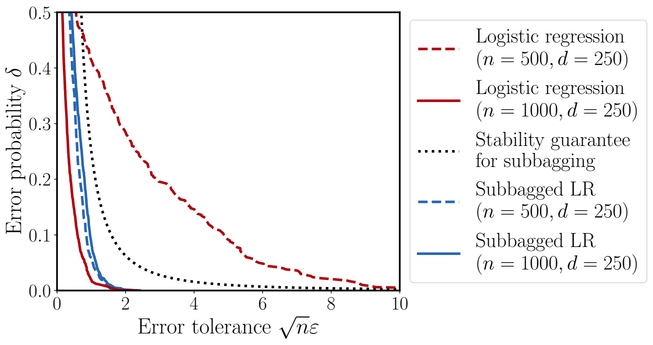
Empirical results

Regression trees



Empirical results

Logistic regression + ridge penalty with $\lambda = 0.001$



Is the guarantee tight?

Theorem: a matching bound for subbagging

There exists a base algorithm $\mathcal{A}$ that returns predictions in $[0,1]$, such that as $B \to \infty$, $\mathcal{A}_{\text{bag}}$ is not (ϵ, δ)-stable for any

$$\delta\epsilon^2 < \frac{1}{c(n-1)} \cdot \frac{p}{1-p}$$

for a constant c, as long as $1/n \ll \delta \leq 1/2$ and $\min\{p, 1-p\} \gg 1/n$.

Is the guarantee tight?

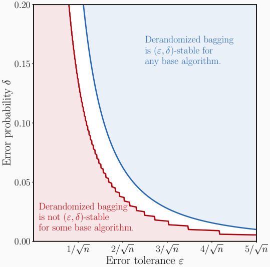
Theorem: a matching bound for subbagging

There exists a base algorithm $\mathcal{A}$ that returns predictions in $[0,1]$, such that as $B \to \infty$, $\mathcal{A}_{\text{bag}}$ is <u>not</u> (ϵ, δ)-stable for any

$$\delta\epsilon^2 < \frac{1}{c(n-1)} \cdot \frac{p}{1-p}$$

for a constant c, as long as $1/n \ll \delta \leq 1/2$ and $\min\{p, 1-p\} \gg 1/n$.

Illustration for $n = 500$, $m = 250$:



Is the guarantee tight?

Proof sketch: a "voting" algorithm

- Let $X_i = 1$ for δn many $i \in [n]$, otherwise $X_i = 0$
- Want to construct $\mathcal{A}$ so that $|\widehat{f}(x) - \widehat{f}^{\setminus i}(x)| \geq \epsilon$ for all i with $X_i = 1$

Is the guarantee tight?

Proof sketch: a "voting" algorithm

- Let $X_i = 1$ for δn many $i \in [n]$, otherwise $X_i = 0$
- Want to construct $\mathcal{A}$ so that $|\widehat{f}(x) - \widehat{f}^{\setminus i}(x)| \geq \epsilon$ for all i with $X_i = 1$
- Let $\widehat{f}_b(x) = \mathbb{1}_{\sum_{i \in b} x_i \geq t}$ for some $t \approx p\delta$
    - If $X_i = 1$, then $i \in S_b \Rightarrow \widehat{f}_b(x)$ is a bit more likely to be 1
    - If $X_i = 0$, then $i \in S_b \Rightarrow \widehat{f}_b(x)$ is a bit more likely to be 0
    - Exact probabilities calculated via the HyperGeometric distrib

Stability definitions

Compare to a more strict definition of stability:

Worst-case stability

$\mathcal{A}$ is ϵ-worst-case-stable if

$$\max_i \left|\widehat{f}(x) - \widehat{f}^{\setminus i}(x)\right| \leq \epsilon \quad \leftarrow \textbf{rather than } \sum_i \mathbb{1}_{|\widehat{f}(x)-\widehat{f}^{\setminus i}(x)|>\epsilon} \leq \delta n$$

when trained on any data set $\mathcal{D}$, and tested on any x

Stability definitions

Compare to a more strict definition of stability:

Worst-case stability

$\mathcal{A}$ is ϵ-worst-case-stable if

$$\max_i \left|\widehat{f}(x) - \widehat{f}^{\setminus i}(x)\right| \leq \epsilon \quad \leftarrow \text{rather than } \sum_i \mathbb{1}_{|\widehat{f}(x)-\widehat{f}^{\setminus i}(x)|>\epsilon} \leq \delta n$$

when trained on any data set $\mathcal{D}$, and tested on any x

Results (with $B \to \infty$):

- For any $\mathcal{A}$, $\mathcal{A}_{\text{bag}}$ is p-worst-case-stable
- For any $\epsilon < p$, there exists an $\mathcal{A}$ s.t. $\mathcal{A}_{\text{bag}}$ is not ϵ-worst-case-stable

Part 2: summary & open questions

Our results:

- Classical bagging & subbagging can be applied to <u>any</u> algorithm $\mathcal{A}$ to achieve an assumption-free stability guarantee
- Downstream, this verifies generalization, predictive inference, etc properties for bagged algorithms

Part 2: summary & open questions

Our results:

- Classical bagging & subbagging can be applied to any algorithm $\mathcal{A}$ to achieve an assumption-free stability guarantee
- Downstream, this verifies generalization, predictive inference, etc properties for bagged algorithms

Open questions:

- How does bagging perform relative to other definitions of stability?
- Guarantees for aggregation procedures aside from averaging?
- Guarantees for structured prediction problems ($\mathcal{Y} \not\subseteq \mathbb{R}$)?

Thank you!