Computer-intensive Estimation and Prediction for Time Series and Deep Neural Networks

Defense Talk

Kejin Wu
(Advised by Prof. Dimitris Politis)

Department of Mathematics
University of California, San Diego

April 17, 2025

Overview

1. Prediction inference of Non-linear Autoregressive models
2. Deep Model-free generative prediction method for regression
3. Scalable subsampling for DNN training

Prediction inference of Non-linear Autoregressive models[1]

[1] This part is based on:

- Wu, K. and Politis, D.N., Bootstrap Prediction Inference of Nonlinear Autoregressive Models, *Journal of Time Series Analysis* 2024, 45, 800-822.
- Wu, K., Non-parametric Forward Bootstrap on Predicting Non-linear Time Series: Consistency, Pertinence and Debiasing, *Stats* 2023, 6(3), 839-867.

Background

Background

- **Time series** is a (discrete-time) stochastic process, i.e., $\{X_t, t \in \mathbb{Z}\}$. Its realization is called time series data, e.g., heights of ocean tides, and counts of sunspots.

Background

- **Time series** is a (discrete-time) stochastic process, i.e., $\{X_t, t \in \mathbb{Z}\}$. Its realization is called time series data, e.g., heights of ocean tides, and counts of sunspots.

- **Prediction inference** is about determining Optimal Predictor (OP), usually in L_2 or L_1 sense, and Prediction Interval (PI), percentile or centered version, of future value X_{T+k}, $k \geq 1$, based on observed $\{X_0, \ldots, X_T\}$. We are concerned about the Coverage Rate (CVR) and Length (LEN) of PI.

Two situations

Two situations

Simple case:

If $\{X_0, \ldots, X_T\}$ are *i.i.d.*. Take sample mean and sample median to be L_2 and L_1 OPs, respectively. Rely on sample quantile values to build PIs.

Two situations

Simple case:

If $\{X_0, \ldots, X_T\}$ are *i.i.d.*. Take sample mean and sample median to be L_2 and L_1 OPs, respectively. Rely on sample quantile values to build PIs.

Time series model:

We assume that the time series data is generated by some underlying mechanism:

$$X_t = G(\boldsymbol{X}_{t-p}, \epsilon_t),$$

where:

- $G(\cdot, \cdot)$ could be any suitable linear/non-linear function that makes the time series have desired property.
- $\epsilon_t \sim F_\epsilon$ is called innovation and assumed to be *i.i.d.* with appropriate moments and independent with $X_{t-i}, i \geq 1$.
- $\boldsymbol{X}_{t-p}$ represents $\{X_{t-1}, \ldots, X_{t-p}\}$.

Monte Carlo (MC) simulation for multi-step ahead prediction

Monte Carlo (MC) simulation for multi-step ahead prediction

Apply Monte Carlo (MC) simulation to do predictions:

1. Simulate $\{\epsilon_{T+1}^{(i)}, \ldots, \epsilon_{T+k}^{(i)}\}_{i=1}^{M}$ from F_{ϵ}.
2. Compute pseudo $\{X_{T+k}^{(i)}\}_{i=1}^{M}$, i.e., $X_{T+j}^{(i)} = G(\boldsymbol{X}_{T+j-p}, \epsilon_{T+j}^{(i)})$, for $j = 1, \ldots, k$.
3. Take sample mean and median of $\{X_{T+k}^{(i)}\}_{i=1}^{M}$ to approximate optimal predictors, respectively. Take corresponding quantile values to approximate PIs with arbitrary coverage rates. We call such type of PI Simulation PI (SPI).

Monte Carlo (MC) simulation for multi-step ahead prediction

Apply Monte Carlo (MC) simulation to do predictions:

1. Simulate $\{\epsilon_{T+1}^{(i)}, \ldots, \epsilon_{T+k}^{(i)}\}_{i=1}^{M}$ from F_{ϵ}.
2. Compute pseudo $\{X_{T+k}^{(i)}\}_{i=1}^{M}$, i.e., $X_{T+j}^{(i)} = G(\boldsymbol{X}_{T+j-p}, \epsilon_{T+j}^{(i)})$, for $j = 1, \ldots, k$.
3. Take sample mean and median of $\{X_{T+k}^{(i)}\}_{i=1}^{M}$ to approximate optimal predictors, respectively. Take corresponding quantile values to approximate PIs with arbitrary coverage rates. We call such type of PI Simulation PI (SPI).

Limitation: In practice, model information is generally not known to participators. Thus, this prediction is *Oracle*.

Bootstrap for multi-step prediction

Bootstrap for multi-step prediction

Apply Bootstrap to do predictions:

1. Bootstrap $\{\hat{\epsilon}^{(i)}_{T+1}, \ldots, \hat{\epsilon}^{(i)}_{T+k}\}_{i=1}^{M}$ from $\widehat{F}_{\epsilon}$.
2. Compute pseudo $\{\widehat{X}^{(i)}_{T+k}\}_{i=1}^{M}$ iteratively, i.e., $\widehat{X}^{(i)}_{T+j} = \widehat{G}(\boldsymbol{X}_{T+j-p}, \hat{\epsilon}^{(i)}_{T+j})$, for $j = 1, \ldots, k$.
3. Take sample mean and median of $\{\widehat{X}^{(i)}_{T+k}\}_{i=1}^{M}$ to approximate optimal predictors, respectively. Take corresponding quantile values to approximate PIs with arbitrary coverage rates. We call such type of PI Quantile PI (QPI).

Bootstrap for multi-step prediction

Apply Bootstrap to do predictions:

1. Bootstrap $\{\hat{\epsilon}_{T+1}^{(i)}, \ldots, \hat{\epsilon}_{T+k}^{(i)}\}_{i=1}^{M}$ from $\widehat{F}_{\epsilon}$.
2. Compute pseudo $\{\widehat{X}_{T+k}^{(i)}\}_{i=1}^{M}$ iteratively, i.e., $\widehat{X}_{T+j}^{(i)} = \widehat{G}(\boldsymbol{X}_{T+j-p}, \hat{\epsilon}_{T+j}^{(i)})$, for $j = 1, \ldots, k$.
3. Take sample mean and median of $\{\widehat{X}_{T+k}^{(i)}\}_{i=1}^{M}$ to approximate optimal predictors, respectively. Take corresponding quantile values to approximate PIs with arbitrary coverage rates. We call such type of PI Quantile PI (QPI).

Limitation: In practice with finite samples, this Bootstrap-based PI suffers undercoverage.

Main algorithm of our prediction method

Main algorithm of our prediction method

1. Do estimations to get $\widehat{G}(\cdot,\cdot)$ and $\widehat{F}_{\epsilon}$. Then, perform the bootstrap prediction to get $\widehat{X}_{T+k}$.

Main algorithm of our prediction method

1. Do estimations to get $\widehat{G}(\cdot, \cdot)$ and $\widehat{F}_{\epsilon}$. Then, perform the bootstrap prediction to get $\widehat{X}_{T+k}$.
2. Generate a pseudo series $\{X_0^*, \ldots, X_{T+k}^*\}$ by viewing $\widehat{G}(\cdot, \cdot)$ and $\widehat{F}_{\epsilon}$ as the true model and innovation distribution in the bootstrap world.

Main algorithm of our prediction method

1 Do estimations to get $\widehat{G}(\cdot, \cdot)$ and $\widehat{F}_\epsilon$. Then, perform the bootstrap prediction to get $\widehat{X}_{T+k}$.

2 Generate a pseudo series $\{X_0^*, \ldots, X_{T+k}^*\}$ by viewing $\widehat{G}(\cdot, \cdot)$ and $\widehat{F}_\epsilon$ as the true model and innovation distribution in the bootstrap world.

3 Re-estimate model to get $\widehat{G}^*(\cdot, \cdot)$ with $\{X_0^*, \ldots, X_T^*\}$; Re-define $\{X_{T-p+1}^* = X_{T-p+1}, \ldots, X_T^* = X_T\}$. Then do the bootstrap prediction with $\widehat{G}^*(\cdot, \cdot)$ and $\widehat{F}_\epsilon$ to get $\widehat{X}_{T+k}^*$. Record the predictive root $X_{T+k}^* - \widehat{X}_{T+k}^*$ in the bootstrap world.

Main algorithm of our prediction method

1. Do estimations to get $\widehat{G}(\cdot,\cdot)$ and $\widehat{F}_\epsilon$. Then, perform the bootstrap prediction to get $\widehat{X}_{T+k}$.
2. Generate a pseudo series $\{X_0^*,\ldots,X_{T+k}^*\}$ by viewing $\widehat{G}(\cdot,\cdot)$ and $\widehat{F}_\epsilon$ as the true model and innovation distribution in the bootstrap world.
3. Re-estimate model to get $\widehat{G}^*(\cdot,\cdot)$ with $\{X_0^*,\ldots,X_T^*\}$; Re-define $\{X_{T-p+1}^* = X_{T-p+1}, \ldots, X_T^* = X_T\}$. Then do the bootstrap prediction with $\widehat{G}^*(\cdot,\cdot)$ and $\widehat{F}_\epsilon$ to get $\widehat{X}_{T+k}^*$. Record the predictive root $X_{T+k}^* - \widehat{X}_{T+k}^*$ in the bootstrap world.
4. Repeat the above process M times, collect M predictive roots and take its empirical distribution to approximate the distribution of $X_{T+k} - \widehat{X}_{T-k}$.

Main algorithm of our prediction method

1. Do estimations to get $\widehat{G}(\cdot,\cdot)$ and $\widehat{F}_\epsilon$. Then, perform the bootstrap prediction to get $\widehat{X}_{T+k}$.
2. Generate a pseudo series $\{X_0^*, \ldots, X_{T+k}^*\}$ by viewing $\widehat{G}(\cdot,\cdot)$ and $\widehat{F}_\epsilon$ as the true model and innovation distribution in the bootstrap world.
3. Re-estimate model to get $\widehat{G}^*(\cdot,\cdot)$ with $\{X_0^*, \ldots, X_T^*\}$; Re-define $\{X_{T-p+1}^* = X_{T-p+1}, \ldots, X_T^* = X_T\}$. Then do the bootstrap prediction with $\widehat{G}^*(\cdot,\cdot)$ and $\widehat{F}_\epsilon$ to get $\widehat{X}_{T+k}^*$. Record the predictive root $X_{T+k}^* - \widehat{X}_{T+k}^*$ in the bootstrap world.
4. Repeat the above process M times, collect M predictive roots and take its empirical distribution to approximate the distribution of $X_{T+k} - \widehat{X}_{T-k}$.
5. The $(1-\alpha)100\%$ PI for X_{T+k} centered at $\widehat{X}_{T+k}$ can be approximated by $[\widehat{X}_{T+k} + q(\alpha/2), \widehat{X}_{T+k} + q(1-\alpha/2)]$, where $q(\alpha)$ is the α-quantile of the empirical distribution of $X_{T+k}^* - \widehat{X}_{T+k}^*$.

Pertinent Prediction Interval (PPI)

Definition (informal): Pertinent Prediction Interval (PPI)

In brief, a PI is pertinent if it accounts for all variability involved in the prediction procedure; see Politis (2015) and Wang and Politis (2021) for formal definitions.

Pertinent Prediction Interval (PPI)

Definition (informal): Pertinent Prediction Interval (PPI)

In brief, a PI is pertinent if it accounts for all variability involved in the prediction procedure; see Politis (2015) and Wang and Politis (2021) for formal definitions.

1 $\sup_a |\widehat{F}_\epsilon(a) - F_\epsilon(a)| \xrightarrow{p} 0.$

Pertinent Prediction Interval (PPI)

Definition (informal): Pertinent Prediction Interval (PPI)

In brief, a PI is pertinent if it accounts for all variability involved in the prediction procedure; see Politis (2015) and Wang and Politis (2021) for formal definitions.

1. $\sup_a |\widehat{F}_\epsilon(a) - F_\epsilon(a)| \xrightarrow{p} 0.$
2. $\sup_a |\mathbb{P}(\tau_T A_m^* \leq a) - \mathbb{P}(\tau_T A_m \leq a)| \xrightarrow{p} 0,$

Pertinent Prediction Interval (PPI)

Definition (informal): Pertinent Prediction Interval (PPI)

In brief, a PI is pertinent if it accounts for all variability involved in the prediction procedure; see Politis (2015) and Wang and Politis (2021) for formal definitions.

1. $\sup_a |\widehat{F}_\epsilon(a) - F_\epsilon(a)| \xrightarrow{p} 0.$
2. $\sup_a |\mathbb{P}(\tau_T A_m^* \leq a) - \mathbb{P}(\tau_T A_m \leq a)| \xrightarrow{p} 0,$

Pertinent Prediction Interval (PPI)

Definition (informal): Pertinent Prediction Interval (PPI)

In brief, a PI is pertinent if it accounts for all variability involved in the prediction procedure; see Politis (2015) and Wang and Politis (2021) for formal definitions.

1. $\sup_a |\widehat{F}_\epsilon(a) - F_\epsilon(a)| \xrightarrow{p} 0.$
2. $\sup_a |\mathbb{P}(\tau_T A_m^* \leq a) - \mathbb{P}(\tau_T A_m \leq a)| \xrightarrow{p} 0,$

For example, we assume that we can decompose $G(\boldsymbol{X}_{t-p}, \epsilon_t)$ as $M(\boldsymbol{X}_{t-p}) + \epsilon_t$;
$A_m^* = \widehat{M}^*(\boldsymbol{x}) - \widehat{M}(\boldsymbol{x})$; $A_m = \widehat{M}(\boldsymbol{x}) - M(\boldsymbol{x})$.

Parametric Non-linear Autoregressive (NLAR) models

First, we consider the case that we can decompose $G(\boldsymbol{X}_{t-p}, \epsilon_t)$ as a parametric non-linear model[2]:

$$X_t = G(X_{t-1}, \epsilon_t) = m(X_{t-1}, \theta_1) + \sigma(X_{t-1}, \theta_2)\epsilon_t,$$

[2] To simplify notation, we consider models with order 1.

Parametric Non-linear Autoregressive (NLAR) models

First, we consider the case that we can decompose $G(\boldsymbol{X}_{t-p}, \epsilon_t)$ as a parametric non-linear model[2]:

$$X_t = G(X_{t-1}, \epsilon_t) = m(X_{t-1}, \theta_1) + \sigma(X_{t-1}, \theta_2)\epsilon_t,$$

[2] To simplify notation, we consider models with order 1.

Parametric Non-linear Autoregressive (NLAR) models

First, we consider the case that we can decompose $G(\boldsymbol{X}_{t-p}, \epsilon_t)$ as a parametric non-linear model[2]:

$$X_t = G(X_{t-1}, \epsilon_t) = m(X_{t-1}, \theta_1) + \sigma(X_{t-1}, \theta_2)\epsilon_t,$$

where:

- $m(\cdot)$ is the mean function which is Lipschitz continuous w.r.t. the first and second arguments for their domain, respectively.
- $\sigma(\cdot)$ is the positive and bounded variance function which is Lipschitz continuous w.r.t. the first and second arguments for their domains, respectively.
- $\theta_1 \in \Theta_1$ and $\theta_2 \in \Theta_2$, where Θ_1 and Θ_2 are all bounded sets in $\mathbb{R}^d$.
- For ϵ_t, it is mean zero and variance 1; $f_\epsilon(\cdot)$ is continuous and everywhere positive.

[2] To simplify notation, we consider models with order 1.

Two-step estimation process

Two-step estimation process

- $$\widehat{\theta}_1 = \arg\min_{\vartheta \in \Theta_1} L_T(\vartheta) = \arg\min_{\vartheta \in \Theta_1} \frac{1}{T} \sum_{t=1}^{T} (X_t - m(X_{t-1}, \vartheta))^2$$

Two-step estimation process

- $$\widehat{\theta}_1 = \arg\min_{\vartheta\in\Theta_1} L_T(\vartheta) = \arg\min_{\vartheta\in\Theta_1} \frac{1}{T}\sum_{t=1}^{T}(X_t - m(X_{t-1},\vartheta))^2$$
- $$\widehat{\theta}_2 = \arg\min_{\vartheta\in\Theta_2} K_T(\vartheta,\widehat{\theta}_1) = \arg\min_{\vartheta\in\Theta_2} \left|\frac{1}{T}\sum_{t=1}^{T}\left(\frac{X_t - m(X_{t-1},\widehat{\theta}_1)}{\sigma(X_{t-1},\vartheta)}\right)^2 - 1\right|.$$

Consistency of OP and asymptotic validity of QPI

Consistency of OP and asymptotic validity of QPI

Theorem 1: Consistency of prediction

For $k \geq 1$ we have:

$$\sup_{|x| \leq c_T} \left| F_{X^*_{T+k}|X_T,\ldots,X_0}(x) - F_{X_{T+k}|X_T}(x) \right| \xrightarrow{p} 0,$$

where

Consistency of OP and asymptotic validity of QPI

Theorem 1: Consistency of prediction

For $k \geq 1$ we have:

$$\sup_{|x| \leq c_T} \left| F_{X^*_{T+k}|X_T,\ldots,X_0}(x) - F_{X_{T+k}|X_T}(x) \right| \xrightarrow{p} 0,$$

where

- $X^*_{T+k} = \mathcal{G}(X_T; \hat{\epsilon}^*_{T+1}, \ldots, \hat{\epsilon}^*_{T+k}; \widehat{\theta})$. This is computed by $X^*_{T+i} = m(X^*_{T+i-1}, \widehat{\theta}_1)$ $+\sigma(X^*_{T+i-1}, \widehat{\theta}_2)\hat{\epsilon}^*_{T+i}$ iteratively for $i = 1, \ldots, k$. Similar for X_{T+k}.
- $\{\hat{\epsilon}^*_i\}_{i=T+1}^{T+k}$ are $i.i.d. \sim \widehat{F}_\epsilon$.
- c_T is an appropriate sequence converges to infinity as T converges to infinity.
- $F_{X^*_{T+k}|X_T,\ldots,X_0}(x)$ is the distribution of k-step ahead future value in the bootstrap world, i.e., conditional on all observed data.
- $F_{X_{T+k}|X_T}(x)$ is the distribution of k-step ahead future value in the real world.

Estimation inference of $\widehat{\theta_1}$ and $\widehat{\theta_2}$, $\widehat{\theta_1^*}$ and $\widehat{\theta_2^*}$

Theorem 2: Estimation inference

Based on the realization $\{X_0, \ldots, X_T\} \in \Omega_T$, where $\mathbb{P}((X_0, \ldots, X_T) \notin \Omega_T) = o(1)$ as $T \to \infty$, under other suitable assumptions, we have:

$$\sqrt{T}(\widehat{\theta_1} - \theta_1) \xrightarrow{d} N(0, B_1^{-1}\Omega_1 B_1^{-1})\ ;\ \ \sqrt{T}(\widehat{\theta_2} - \theta_2) \xrightarrow{d} N(0, B_2^{-1}\Omega_2 B_2^{-1});$$

$$\sqrt{T}(\widehat{\theta_1^*} - \widehat{\theta_1}) \xrightarrow{d} N(0, B_1^{-1}\Omega_1 B_1^{-1})\ ;\ \ \sqrt{T}(\widehat{\theta_2^*} - \widehat{\theta_2}) \xrightarrow{d} N(0, B_2^{-1}\Omega_2 B_2^{-1});$$

where

- $\Omega_1 = 4 \cdot \mathbb{E}(\sigma(X_0, \theta_2) R_1 \sigma(X_0, \theta_2))$; $B_1 = 2 \cdot \mathbb{E}\left(\nabla\phi(X_0, \theta_1)(\nabla\phi(X_0, \theta_1))^\top\right)$; $R_1 = \nabla\phi(X_0, \theta_1)$ $\nabla\phi(X_0, \theta_1)^\top$; here ∇ is the gradient operator w.r.t. θ_1.
- $\Omega_2 = 4 \cdot \mathbb{E}(B_3 R_2 B_3^\top)$; $B_3 = \mathbb{E}(\nabla g(X_1, X_0, \theta_2, \theta_1))$; $R_2 = (g(X_1, X_0, \theta_2, \theta_1) - 1)^2$; $B_2 = 2 \cdot (\mathbb{E}(\nabla g(X_1, X_0, \theta_2, \theta_1)) \cdot (\mathbb{E}(\nabla g(X_1, X_0, \theta_2, \theta_1))^\top$; $g(X_1, X_0, \theta_2, \theta_1) = \left(\frac{X_1 - \phi(X_0, \theta_1)}{\sigma(X_0, \theta_2)}\right)^2$; here ∇ is the gradient operator w.r.t. θ_2.

Non-parametric NLAR models

When the parametric format is unknown, we assume that we only know the data-generating mechanism of time series consists of two parts:

$$X_t = G(X_{t-1}, \epsilon_t) = m(X_{t-1}) + \sigma(X_{t-1})\epsilon_t.$$

Local constant estimator

$$\widetilde{m}_h(x) = \frac{\sum_{t=1}^{T} K(\frac{x-X_{t-1}}{h})X_t}{\sum_{t=1}^{T} K(\frac{x-X_{t-1}}{h})} \text{ and } \widetilde{\sigma}_h(x) = \frac{\sum_{t=1}^{T} K(\frac{x-X_{t-1}}{h})(X_t - \widetilde{m}_h(X_{t-1}))^2}{\sum_{t=1}^{T} K(\frac{x-X_{t-1}}{h})};$$

QPI and PPI in the non-parametric prediction approach

Theorem 3: QPI and PPI of non-parametric prediction

Let $\widehat{m}_g(x)$ and $\widehat{\sigma}_g(x)$ be estimated mean and variance functions to generate bootstrap series in the bootstrap world. With the under-smoothing debiasing strategy, i.e., we take $g = h$ and take a bandwidth rate satisfying $hT^{1/5} \to 0$. The QPI and PPI are still possible with the main prediction algorithm.

Simulation for parametric prediction approach

Simulation model (Threshold model):

$$X_t = (0.5 \cdot X_{t-1} + 0.2 \cdot X_{t-2} + 0.1 \cdot X_{t-3})I(X_{t-1} \leq 0) + (0.8 \cdot X_{t-1})I(X_{t-1} > 0) + \epsilon_t; \epsilon_t \sim N(0, 1).$$

Simulation setting:

We take the number of bootstrap times $M = 1000$. We repeat simulations $N = 5000$ times. We take $\alpha = 0.05$.

Simulation measurement:

- $$\text{CVR of the } k\text{-th step ahead prediction} = \frac{1}{N}\sum_{n=1}^{N} I_{X_{n,k} \in [L_{n,k}, U_{n,k}]}, \text{for } k = 1, \ldots, 5.$$
- $$\text{LEN of the } k\text{-th step ahead PI} = \frac{1}{N}\sum_{n=1}^{N}(U_{n,k} - L_{n,k}), \text{for } k = 1, \ldots, 5,$$

where $[L_{n,k}, U_{n,k}]$ and $X_{n,k}$ represent k-th step ahead prediction intervals and the true future value in the n-th replication, respectively.

Simulation results

Threshold Model:	$X_t = (0.5 \cdot X_{t-1} + 0.2 \cdot X_{t-2} + 0.1 \cdot X_{t-3}) I(X_{t-1} \le 0) + (0.8 \cdot X_{t-1}) I(X_{t-1} > 0) + \epsilon_t, \epsilon_t \sim N(0,1)$									
	CVR for each step					LEN for each step				
$T = 400$	1	2	3	4	5	1	2	3	4	5
QPI-f	0.9420	0.9506	0.9468	0.9444	0.9372	3.88	4.68	5.11	5.40	5.58
QPI-p	0.9462	0.9512	0.9502	0.9474	0.9428	3.92	4.72	5.16	5.45	5.64
L_2-PPI-f	0.9446	0.9510	0.9486	0.9470	0.9408	3.90	4.71	5.15	5.44	5.63
L_2-PPI-p	0.9466	0.9542	0.9516	0.9494	0.9434	3.94	4.75	5.20	5.49	5.69
L_1-PPI-f	0.9448	0.9518	0.9478	0.9468	0.9402	3.90	4.71	5.15	5.44	5.62
L_1-PPI-p	0.9470	0.9544	0.9500	0.9486	0.9436	3.94	4.75	5.20	5.49	5.68
SPI	0.9446	0.9534	0.9508	0.9510	0.9454	3.90	4.71	5.16	5.46	5.65
$T = 100$										
QPI-f	0.9270	0.9304	0.9294	0.9272	0.9250	3.81	4.57	4.98	5.23	5.40
QPI-p	0.9370	0.9412	0.9368	0.9372	0.9372	3.98	4.76	5.19	5.46	5.63
L_2-PPI-f	0.9358	0.9352	0.9338	0.9314	0.9298	3.95	4.71	5.13	5.40	5.59
L_2-PPI-p	0.9454	0.9454	0.9444	0.9430	0.9418	4.10	4.90	5.34	5.63	5.83
L_1-PPI-f	0.9364	0.9360	0.9336	0.9310	0.9304	3.95	4.71	5.13	5.39	5.58
L_1-PPI-p	0.9450	0.9456	0.9432	0.9422	0.9412	4.11	4.90	5.33	5.62	5.81
SPI	0.9446	0.9472	0.9498	0.9474	0.9478	3.90	4.71	5.16	5.46	5.65
$T = 50$										
QPI-f	0.8980	0.9054	0.9018	0.8950	0.8926	3.66	4.47	4.87	5.14	5.38
QPI-p	0.9260	0.9314	0.9272	0.9218	0.9212	4.05	4.97	5.42	5.74	5.99
L_2-PPI-f	0.9340	0.9268	0.9214	0.9164	0.9152	4.22	5.10	5.86	6.89	8.97
L_2-PPI-p	0.9522	0.9478	0.9404	0.9400	0.9376	4.60	5.57	6.36	7.33	9.03
L_1-PPI-f	0.9338	0.9268	0.9194	0.9144	0.9130	4.23	5.09	5.82	6.79	8.71
L_1-PPI-p	0.9522	0.9482	0.9384	0.9378	0.9356	4.61	5.55	6.30	7.20	8.71
SPI	0.9494	0.9448	0.9464	0.9458	0.9462	3.90	4.71	5.16	5.46	5.65

Note: "-f" and "-p" represent fitted and predictive residuals, respectively. "L_2" and "L_1" represent the center of PPI is L_2 and L_1 OP, respectively.

Deep Model-free generative prediction method for regression[3]

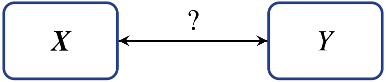
[3]This part is based on:

- Wu, K. and Politis, D.N., Deep Limit Model-free Prediction in Regression. *(Submitted to ACM/IMS Journal of Data Science)*

Regression analysis

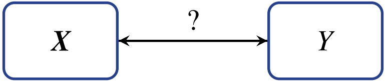
Regression analysis

Regression analysis is a statistical process to explore the relationship between dependent/outcome variable Y and independent/predictors variable $\boldsymbol{X}$:



Regression analysis

Regression analysis is a statistical process to explore the relationship between dependent/outcome variable Y and independent/predictors variable $\boldsymbol{X}$:

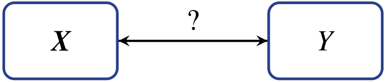


For example,

- Simple linear regression: relationship of heights between father and son;

Regression analysis

Regression analysis is a statistical process to explore the relationship between dependent/outcome variable Y and independent/predictors variable $\boldsymbol{X}$:

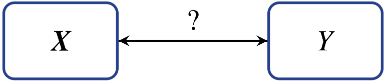


For example,

- Simple linear regression: relationship of heights between father and son;
- Quantile regression: impact of education, experience, etc., on different quantiles of income;

Regression analysis

Regression analysis is a statistical process to explore the relationship between dependent/outcome variable Y and independent/predictors variable $\boldsymbol{X}$:



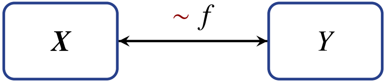
For example,

- Simple linear regression: relationship of heights between father and son;
- Quantile regression: impact of education, experience, etc., on different quantiles of income;
- Casual inference: effects of treatments on patients.

Model as bridge

Model as bridge

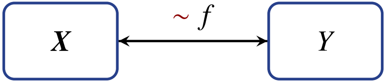
Classically, people assume there is a model f that may explain the relationship between $\boldsymbol{X}$ and Y: [4]



[4] $\sim$ means that the association between $\boldsymbol{X}$ and Y may not be exactly described by f or there is a measurement error.

Model as bridge

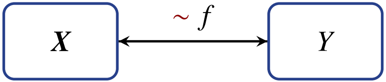
Classically, people assume there is a model f that may explain the relationship between $\boldsymbol{X}$ and Y: [4]



[4] $\sim$ means that the association between $\boldsymbol{X}$ and Y may not be exactly described by f or there is a measurement error.

Model as bridge

Classically, people assume there is a model f that may explain the relationship between $\boldsymbol{X}$ and Y: [4]



For example, a general homoscedastic model:

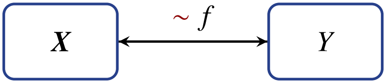
$$Y = f(\boldsymbol{X}) + \varepsilon;$$

where $f(\cdot)$ could be parametric or non-parametric; $\varepsilon \sim F_{\varepsilon}$.

[4] $\sim$ means that the association between $\boldsymbol{X}$ and Y may not be exactly described by f or there is a measurement error.

Model as bridge

Classically, people assume there is a model f that may explain the relationship between $\boldsymbol{X}$ and Y: [4]



For example, a general homoscedastic model:

$$Y = f(\boldsymbol{X}) + \varepsilon;$$

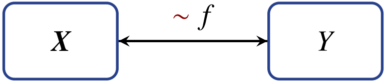
where $f(\cdot)$ could be parametric or non-parametric; $\varepsilon \sim F_{\varepsilon}$.

- Simple linear regression: $Y = \boldsymbol{\beta}^T \boldsymbol{X} + \varepsilon$;

[4] $\sim$ means that the association between $\boldsymbol{X}$ and Y may not be exactly described by f or there is a measurement error.

Model as bridge

Classically, people assume there is a model f that may explain the relationship between $\boldsymbol{X}$ and Y: [4]



For example, a general homoscedastic model:

$$Y = f(\boldsymbol{X}) + \varepsilon;$$

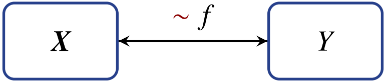
where $f(\cdot)$ could be parametric or non-parametric; $\varepsilon \sim F_{\varepsilon}$.

- Simple linear regression: $Y = \boldsymbol{\beta}^T \boldsymbol{X} + \varepsilon$;
- Quantile regression: $Q_Y(\tau|\boldsymbol{X}) = \boldsymbol{\beta}_\tau^T \boldsymbol{X}$;

[4] $\sim$ means that the association between $\boldsymbol{X}$ and Y may not be exactly described by f or there is a measurement error.

Model as bridge

Classically, people assume there is a model f that may explain the relationship between $\boldsymbol{X}$ and Y: [4]



For example, a general homoscedastic model:

$$Y = f(\boldsymbol{X}) + \varepsilon;$$

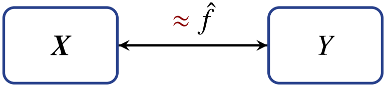
where $f(\cdot)$ could be parametric or non-parametric; $\varepsilon \sim F_{\varepsilon}$.

- Simple linear regression: $Y = \boldsymbol{\beta}^T \boldsymbol{X} + \varepsilon$;
- Quantile regression: $Q_Y(\tau|\boldsymbol{X}) = \boldsymbol{\beta}_\tau^T \boldsymbol{X}$;
- Casual inference: $f(\boldsymbol{x}) = \mathbb{E}\left(Y^1 - Y^0 \mid \boldsymbol{X} = \boldsymbol{x}\right)$ (Conditional Treatment Effects function).

[4] $\sim$ means that the association between $\boldsymbol{X}$ and Y may not be exactly described by f or there is a measurement error.

Estimation of model

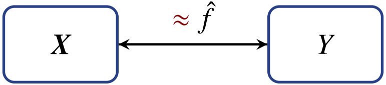
In practice, we estimate $f(\cdot)$ by $\hat{f}(\cdot)$ based on sample $\{\boldsymbol{X}_i, Y_i\}_{i=1}^n$: [5]



[5] Compared to $\sim$, $\approx$ involves additional estimation error.

Estimation of model

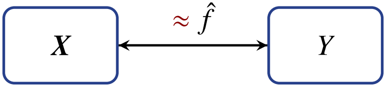
In practice, we estimate $f(\cdot)$ by $\hat{f}(\cdot)$ based on sample $\{\boldsymbol{X}_i, Y_i\}_{i=1}^n$: [5]



[5] Compared to $\sim$, $\approx$ involves additional estimation error.

Estimation of model

In practice, we estimate $f(\cdot)$ by $\hat{f}(\cdot)$ based on sample $\{\boldsymbol{X}_i, Y_i\}_{i=1}^n$: [5]



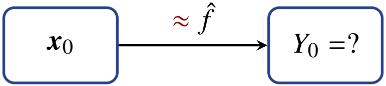
Therefore, we need to quantify the estimation accuracy, e.g., by Confidence Interval (CI).

[5] Compared to $\sim$, $\approx$ involves additional estimation error.

Prediction with model

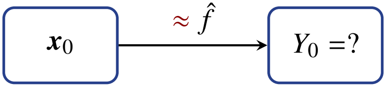
Prediction with model

We care about the prediction of Y_0 given some future value of $\boldsymbol{X}_0 = \boldsymbol{x}_0$ based on $\hat{f}(\cdot)$:



Prediction with model

We care about the prediction of Y_0 given some future value of $\boldsymbol{X}_0 = \boldsymbol{x}_0$ based on $\hat{f}(\cdot)$:



To quantify the prediction accuracy, we build Prediction Interval (PI). However, it usually requires the normality assumption or it suffers the undercoverage in the finite sample cases.

What if model is wrong?

What if model is wrong?

> *Essentially, all models are wrong, but some are useful.*
>
> —*George Box*

What if model is wrong?

> *Essentially, all models are wrong, but some are useful.*
>
> —*George Box*

Possible scenarios in applications:

What if model is wrong?

> *Essentially, all models are wrong, but some are useful.*
>
> —*George Box*

Possible scenarios in applications:

- The true model is non-linear, but we may assume it is linear;

What if model is wrong?

> *Essentially, all models are wrong, but some are useful.*
>
> —*George Box*

Possible scenarios in applications:

- The true model is non-linear, but we may assume it is linear;
- Even a complicated $Y = f(\boldsymbol{X}) + g(\boldsymbol{X}) \cdot \varepsilon$, where f and g are in non-parametric form, could also be wrong since it assumes an additive structure;

What if model is wrong?

Essentially, all models are wrong, but some are useful.

—*George Box*

Possible scenarios in applications:

- The true model is non-linear, but we may assume it is linear;
- Even a complicated $Y = f(\boldsymbol{X}) + g(\boldsymbol{X}) \cdot \varepsilon$, where f and g are in non-parametric form, could also be wrong since it assumes an additive structure;
- Sometimes, a "wrong" model may work better than the true model for prediction purposes.

Basic assumptions

- $\boldsymbol{X}$ and Y have a joint distribution $P_{\boldsymbol{X},Y}$;[6]
- The domain of Y and X are compact sets, respectively, i.e., $\mathcal{Y} := [-M_1, M_1]$ and $\mathcal{X} := [-M_2, M_2]^d$; M_1 and M_2 are two positive constants. [7]

[6]We assume that the joint density of $P_{\boldsymbol{X},Y}$ exists to avoid some potential degenerate cases.

[7]A weaker assumption could be made such that $P(|Y| > \tau) \leq C\rho_1^{-\tau}$ and $P(\|\boldsymbol{X}\| > \tau) \leq C\rho_2^{-\tau}$ for some appropriate ρ_1 and ρ_2 (sub-exponential). Then, the event that $\boldsymbol{X}$ and Y belong to a compact set has a *high probability*.

Basic assumptions

- $\boldsymbol{X}$ and Y have a joint distribution $P_{\boldsymbol{X},Y}$;[6]
- The domain of Y and X are compact sets, respectively, i.e., $\mathcal{Y} := [-M_1, M_1]$ and $\mathcal{X} := [-M_2, M_2]^d$; M_1 and M_2 are two positive constants. [7]

We call our method **Model-free** since no restricted model format is assumed.

[6] We assume that the joint density of $P_{\boldsymbol{X},Y}$ exists to avoid some potential degenerate cases.

[7] A weaker assumption could be made such that $P(|Y| > \tau) \leq C\rho_1^{-\tau}$ and $P(\|\boldsymbol{X}\| > \tau) \leq C\rho_2^{-\tau}$ for some appropriate ρ_1 and ρ_2 (sub-exponential). Then, the event that $\boldsymbol{X}$ and Y belong to a compact set has a *high probability*.

Intuition

Intuition

In the standard regression context we have the diagram, $\boldsymbol{X} \xleftrightarrow{\sim f} Y$; $\sim$ is due to the model misspecification/insufficiency and unobserved measurement error.

Intuition

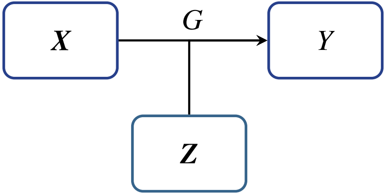
In the standard regression context we have the diagram, $\boldsymbol{X} \xleftrightarrow{\sim f} Y$; $\sim$ is due to the model misspecification/insufficiency and unobserved measurement error.

Can we **outsource** the unobserved error and make our model as flexible as it could?

Intuition

In the standard regression context we have the diagram, $\boldsymbol{X} \overset{\sim f}{\longleftrightarrow} Y$; $\sim$ is due to the model misspecification/insufficiency and unobserved measurement error.

Can we **outsource** the unobserved error and make our model as flexible as it could?



Here, $G : \mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$; $\mathcal{Z}$ is the domain of the reference random variable $\boldsymbol{Z}$.

Noise outsourcing lemma

$G(\cdot,\cdot)$ could make a connection between $\boldsymbol{X}$ and Y.

Noise outsourcing lemma

$G(\cdot, \cdot)$ could make a connection between $\boldsymbol{X}$ and Y.

Lemma 1: Noise outsourcing (Bloem-Reddy et al., 2020)

Let $\boldsymbol{X}$ and Y be random variables with joint distribution $P_{\boldsymbol{X},Y}$. Then, there is a measurable function $G : [0, 1]^p \times \mathcal{X} \to \mathcal{Y}$ such that

$$Y \overset{a.s.}{=} G(\boldsymbol{X}, Z), \quad \text{where } Z \sim \text{Uniform}[0, 1]^p \text{ and } Z \perp\!\!\!\perp \boldsymbol{X}.$$

Noise outsourcing lemma

$G(\cdot,\cdot)$ could make a connection between $\boldsymbol{X}$ and Y.

Lemma 1: Noise outsourcing (Bloem-Reddy et al., 2020)

Let $\boldsymbol{X}$ and Y be random variables with joint distribution $P_{\boldsymbol{X},Y}$. Then, there is a measurable function $G : [0,1]^p \times \mathcal{X} \to \mathcal{Y}$ such that

$$Y \overset{a.s.}{=} G(\boldsymbol{X}, Z), \quad \text{where } Z \sim \text{Uniform}[0,1]^p \text{ and } Z \perp\!\!\!\perp \boldsymbol{X}.$$

In other words, the randomness in the conditional distribution of Y given $\boldsymbol{X} = \boldsymbol{x}$ is outsourced to reference random variable Z through $G(\boldsymbol{x}, Z)$.

A continuous counterpart of $G(\cdot, \cdot)$

A continuous counterpart of $G(\cdot, \cdot)$

To estimate $G(\cdot, \cdot)$ with data, we hope it can possess some smoothness property (at least C^0). It turns out that:

A continuous counterpart of $G(\cdot,\cdot)$

To estimate $G(\cdot,\cdot)$ with data, we hope it can possess some smoothness property (at least C^0). It turns out that:

Proposition 1: A continuous counterpart of $G(\cdot,\cdot)$ exists

Under some basic assumptions, we have a continuous $\widetilde{G}(\cdot,\cdot) : \mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$ such that $\widetilde{G}(\boldsymbol{x}, z) = G(\boldsymbol{x}, z)$ for all $(\boldsymbol{x}, z)$ except a negligible (in Lebesgue measure) set.

A continuous counterpart of $G(\cdot,\cdot)$

To estimate $G(\cdot,\cdot)$ with data, we hope it can possess some smoothness property (at least C^0). It turns out that:

Proposition 1: A continuous counterpart of $G(\cdot,\cdot)$ exists

Under some basic assumptions, we have a continuous $\widetilde{G}(\cdot,\cdot) : \mathcal{X} \times \mathcal{Z} \to \mathcal{Y}$ such that $\widetilde{G}(\boldsymbol{x}, z) = G(\boldsymbol{x}, z)$ for all $(\boldsymbol{x}, z)$ except a negligible (in Lebesgue measure) set.

To simplify the notation, we will keep using $G(\cdot,\cdot)$ for this continuous counterpart. We will focus on estimating this continuous variant by Deep Neural Networks (DNN) and make predictions based on it.

The structure of Deep Neural Networks

The structure of Deep Neural Networks

In short, the structure of fully connected feedforward Deep Neural Networks (DNN) mainly depends on:

- The input and output dimensions;
- Depth: $L \in \mathbb{N}$;
- Width: $\boldsymbol{W} \in \mathbb{N}^L$.

The structure of Deep Neural Networks

In short, the structure of fully connected feedforward Deep Neural Networks (DNN) mainly depends on:

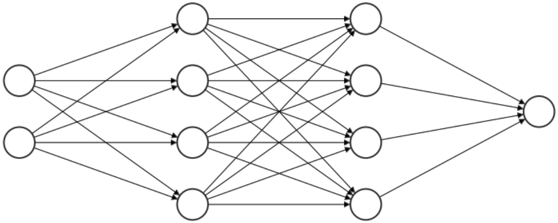
- The input and output dimensions;
- Depth: $L \in \mathbb{N}$;
- Width: $\boldsymbol{W} \in \mathbb{N}^L$.

We can write a DNN function f_{DNN} as

$$f_{\text{DNN}}(\boldsymbol{x}) = \boldsymbol{A}_{L+1}(\phi(\boldsymbol{A}_L(\cdots\phi(\boldsymbol{A}_3\phi(\boldsymbol{A}_2\phi(\boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2) + \boldsymbol{b}_3)\cdots) + \boldsymbol{b}_L) + \boldsymbol{b}_{L+1};$$

where $\{\boldsymbol{A}_i\}_{i=1}^{L+1}$ are weight matrices whose shapes depend on $\boldsymbol{W}$ and output dimension; $\{\boldsymbol{b}_i\}_{i=1}^{L+1}$ are intercept terms; $\phi(\cdot)$ is the activation function.

Figure 1: The illustration of a fully connected DNN with $L = 2$, $W_1 = W_2 = 4$; input dimension and output dimension are 2 and 1, respectively.



Training algorithm

Training algorithm

Algorithm Training procedure to get empirically optimal estimator $\widehat{H}$

1: Initiate a DNN $H_{\boldsymbol{\theta}} \in \mathcal{F}_{\mathrm{DNN}}$[8] and simulate $\{Z_i\}_{i=1}^n$ from P_Z.
2: **for** number of epochs **do**
3: Update $H_{\boldsymbol{\theta}}$ by descending its gradient with the chosen optimization algorithm:

$$\nabla_{\boldsymbol{\theta}} \left\{ \frac{1}{n} \sum_{i=1}^{n} (Y_i - H_{\boldsymbol{\theta}}(\boldsymbol{X}_i, Z_i))^2 \right\}.$$

4: Clip the parameter of $H_{\boldsymbol{\theta}}$ to $[-m, m]$.
5: **end for**
6: **Return** The estimated $\widehat{H}(\cdot, \cdot)$.

[8] $\mathcal{F}_{\mathrm{DNN}}$ is a user-chosen space that contains all DNN candidates.

Error bound for $\widehat{H}$

Error bound for $\widehat{H}$

Theorem 4: A high probability non-asymptotic error bound for $\widehat{H}$

Taking the appropriate reference random variable Z and $\mathcal{F}_{\text{DNN}}$ to be a class of fully connected feedforward DNN functions with width W and depth L.

Error bound for $\widehat{H}$

Theorem 4: A high probability non-asymptotic error bound for $\widehat{H}$

Taking the appropriate reference random variable Z and $\mathcal{F}_{\mathrm{DNN}}$ to be a class of fully connected feedforward DNN functions with width W and depth L.

When sample size n is large enough and under some further mild conditions, we have:

$$\left\|\widehat{H} - G\right\|^2_{L^2(X,Z)} \leq C \cdot n^{-\frac{2}{\tau+d+p}} + o(n^{-\frac{2}{\tau+d+p}}); \text{ for } d+p \geq 2; \tau > 2;$$

with probability at least $1 - \exp(-n^{\frac{d+p}{\tau+d+p}})$; where C is a constant.

Error bound for $\widehat{H}$

Theorem 4: A high probability non-asymptotic error bound for $\widehat{H}$

Taking the appropriate reference random variable Z and $\mathcal{F}_{\text{DNN}}$ to be a class of fully connected feedforward DNN functions with width W and depth L.

When sample size n is large enough and under some further mild conditions, we have:

$$\left\|\widehat{H} - G\right\|^2_{L^2(X,Z)} \leq C \cdot n^{-\frac{2}{\tau+d+p}} + o(n^{-\frac{2}{\tau+d+p}}); \text{ for } d+p \geq 2; \tau > 2;$$

with probability at least $1 - \exp(-n^{\frac{d+p}{\tau+d+p}})$; where C is a constant.

$W := 3^{d+p+3} \max\left\{(d+p)\left\lfloor N_1^{1/(d+p)} \right\rfloor, N_1 + 1\right\}$; $L := 12N_2 + 14 + 2(d+p)$; $N_1 = \left\lceil \frac{n^{\frac{d+p}{2(\tau+d+p)}}}{\log n} \right\rceil$; $N_2 = \lceil \log(n) \rceil$.

Estimation of conditional distribution

Estimation of conditional distribution

Define $\widehat{F}_{\widehat{H}(\boldsymbol{x}_0,Z)}$ as the empirical distribution of $\{\widehat{H}(\boldsymbol{x}_0, Z_i)\}_{i=1}^{S}$; S is the number of Monte Carlo sampling we apply to generate values of reference random variable.

Estimation of conditional distribution

Define $\widehat{F}_{\widehat{H}(\boldsymbol{x}_0,Z)}$ as the empirical distribution of $\{\widehat{H}(\boldsymbol{x}_0, Z_i)\}_{i=1}^{S}$; S is the number of Monte Carlo sampling we apply to generate values of reference random variable.

Under some additional restrictions about $P_{\boldsymbol{X},Y}$, we have

Theorem 5: Uniform estimation of $F_{Y|X}$ based on $\widehat{H}$

we have:

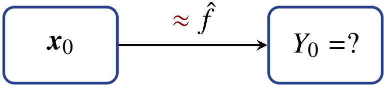
$$\sup_{y} \left| \widehat{F}_{\widehat{H}(\boldsymbol{x}_0,Z)}(y) - F_{Y|\boldsymbol{x}_0}(y) \right| \xrightarrow{p} 0, \text{ as } n \to \infty, S \to \infty,$$

for any $\boldsymbol{x}_0 \in \mathcal{X}$ and $y \in \mathcal{Y}$, with probability at least $1 - \exp(-n^{\frac{d+p}{\tau+d+p}})$.

Motivation to make prediction interval

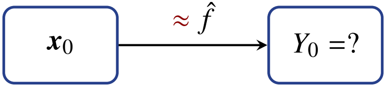
Motivation to make prediction interval

The diagram to do prediction in Background:



Motivation to make prediction interval

The diagram to do prediction in Background:

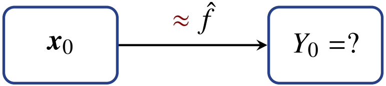


Here, $\approx$ represents error comes from two sources:

1. The association between $\boldsymbol{X}$ and Y is not exactly described by f or there is measurement error;
2. The estimation error within $\hat{f}$.

Motivation to make prediction interval

The diagram to do prediction in Background:



Here, $\approx$ represents error comes from two sources:

1. The association between $\boldsymbol{X}$ and Y is not exactly described by f or there is measurement error;
2. The estimation error within $\hat{f}$.

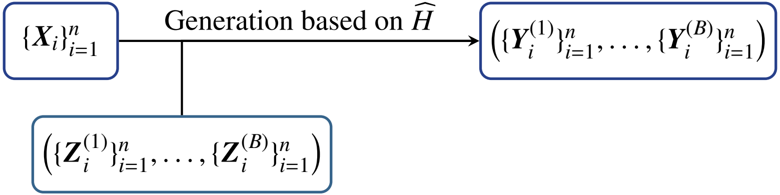
An oracle $G(\cdot, \cdot)$ can solve both error sources a.s.. However, error (2) still exists in practice.

Preparations for PPI

In a similar idea with Part I, we mimic the estimation process by pseudo values:

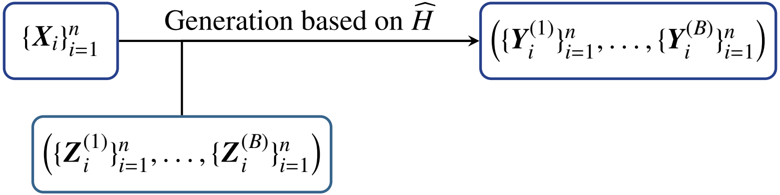
Preparations for PPI

In a similar idea with Part I, we mimic the estimation process by pseudo values:



Preparations for PPI

In a similar idea with Part I, we mimic the estimation process by pseudo values:



Then, make re-estimation to get $\{\widehat{H}^{(b)}\}_{b=1}^B$ based on $\left(\{\boldsymbol{Y}_i^{(1)}\}_{i=1}^n, \ldots, \{\boldsymbol{Y}_i^{(B)}\}_{i=1}^n\right)$, $\{\boldsymbol{X}_i\}_{i=1}^n$ and $\{\boldsymbol{Z}_i\}_{i=1}^n$.

The form of PPI based on $\widehat{H}$

The form of PPI based on $\widehat{H}$

Conditional on $\{(\boldsymbol{X}_i, Y_i, Z_i)\}_{i=1}^n$, we approximate the predictive root R_0 by the variant R_0^*:

$$R_0^* \xrightarrow[d]{Approximate} R_0;$$

where,

The form of PPI based on $\widehat{H}$

Conditional on $\{(\boldsymbol{X}_i, Y_i, Z_i)\}_{i=1}^n$, we approximate the predictive root R_0 by the variant R_0^*:

$$R_0^* \xrightarrow[d]{Approximate} R_0;$$

where,

- R_0 could be $Y_0 - \widehat{Y}_{0,L_2}$; $Y_0 \sim P_{Y|\boldsymbol{x}_0}$ and $\widehat{Y}_{0,L_2} := \mathbb{E}(\widehat{H}(\boldsymbol{x}_0, Z))$ is the *estimated* optimal L_2 conditional point prediction; we approximate it by $\frac{1}{S}\sum_{s=1}^{S}\widehat{H}(\boldsymbol{x}_0, Z_s)$;

The form of PPI based on $\widehat{H}$

Conditional on $\{(\boldsymbol{X}_i, Y_i, Z_i)\}_{i=1}^{n}$, we approximate the predictive root R_0 by the variant R_0^*:

$$R_0^* \xrightarrow[d]{Approximate} R_0;$$

where,

- R_0 could be $Y_0 - \widehat{Y}_{0,L_2}$; $Y_0 \sim P_{Y|\boldsymbol{x}_0}$ and $\widehat{Y}_{0,L_2} := \mathbb{E}(\widehat{H}(\boldsymbol{x}_0, Z))$ is the *estimated* optimal L_2 conditional point prediction; we approximate it by $\frac{1}{S}\sum_{s=1}^{S}\widehat{H}(\boldsymbol{x}_0, Z_s)$;
- R_0^* could be $Y_0^{(b)} - \widehat{Y}_{0,L_2}^{(b)}$; $Y_0^{(b)} \sim \widehat{H}(\boldsymbol{x}_0, Z)$ and $\widehat{Y}_{0,L_2}^{(b)} := \mathbb{E}(\widehat{H}^{(b)}(\boldsymbol{x}_0, Z))$ is the *estimated* optimal L_2 point prediction conditional on training data; we approximate it by $\frac{1}{S}\sum_{s=1}^{S}\widehat{H}^{(b)}(\boldsymbol{x}_0, Z_s)$; $\widehat{H}^{(b)}$ is the b-th re-estimation.

The form of PPI based on $\widehat{H}$

Conditional on $\{(\boldsymbol{X}_i, Y_i, Z_i)\}_{i=1}^n$, we approximate the predictive root R_0 by the variant R_0^*:

$$R_0^* \xrightarrow[d]{Approximate} R_0;$$

where,

- R_0 could be $Y_0 - \widehat{Y}_{0,L_2}$; $Y_0 \sim P_{Y|\boldsymbol{x}_0}$ and $\widehat{Y}_{0,L_2} := \mathbb{E}(\widehat{H}(\boldsymbol{x}_0, Z))$ is the *estimated* optimal L_2 conditional point prediction; we approximate it by $\frac{1}{S}\sum_{s=1}^{S}\widehat{H}(\boldsymbol{x}_0, Z_s)$;
- R_0^* could be $Y_0^{(b)} - \widehat{Y}_{0,L_2}^{(b)}$; $Y_0^{(b)} \sim \widehat{H}(\boldsymbol{x}_0, Z)$ and $\widehat{Y}_{0,L_2}^{(b)} := \mathbb{E}(\widehat{H}^{(b)}(\boldsymbol{x}_0, Z))$ is the *estimated* optimal L_2 point prediction conditional on training data; we approximate it by $\frac{1}{S}\sum_{s=1}^{S}\widehat{H}^{(b)}(\boldsymbol{x}_0, Z_s)$; $\widehat{H}^{(b)}$ is the b-th re-estimation.

The form of PPI based on $\widehat{H}$

Conditional on $\{(\boldsymbol{X}_i, Y_i, Z_i)\}_{i=1}^n$, we approximate the predictive root R_0 by the variant R_0^*:

$$R_0^* \xrightarrow[d]{Approximate} R_0;$$

where,

- R_0 could be $Y_0 - \widehat{Y}_{0,L_2}$; $Y_0 \sim P_{Y|\boldsymbol{x}_0}$ and $\widehat{Y}_{0,L_2} := \mathbb{E}(\widehat{H}(\boldsymbol{x}_0, Z))$ is the *estimated* optimal L_2 conditional point prediction; we approximate it by $\frac{1}{S}\sum_{s=1}^{S}\widehat{H}(\boldsymbol{x}_0, Z_s)$;
- R_0^* could be $Y_0^{(b)} - \widehat{Y}_{0,L_2}^{(b)}$; $Y_0^{(b)} \sim \widehat{H}(\boldsymbol{x}_0, Z)$ and $\widehat{Y}_{0,L_2}^{(b)} := \mathbb{E}(\widehat{H}^{(b)}(\boldsymbol{x}_0, Z))$ is the *estimated* optimal L_2 point prediction conditional on training data; we approximate it by $\frac{1}{S}\sum_{s=1}^{S}\widehat{H}^{(b)}(\boldsymbol{x}_0, Z_s)$; $\widehat{H}^{(b)}$ is the b-th re-estimation.

Thus, a pertinent PI with $1-\alpha$ coverage rate centered at $\widehat{Y}_{0,L_2}$ has the form:

$$\left[\widehat{Y}_{0,L_2} + Q_{\alpha/2}, \widehat{Y}_{0,L_2} + Q_{1-\alpha/2}\right];$$

$Q_{\alpha/2}$ and $Q_{1-\alpha/2}$ are $\alpha/2$ and $1-\alpha/2$ lower quantiles of $P_{R_0^*}$, the distribution of R_0^*. In practice, $P_{R_0^*}$ can be approximated by the empirical distribution of $\{Y_0^{(b)} - \widehat{Y}_{0,L_2}^{(b)}\}_{b=1}^{B}$.

Other DNN generative methods

Recently, Zhou et al. (2023) and Liu et al. (2021) proposed two conditional generators to estimate the conditional distribution in the regression context. Their methods rely on the adversarial training strategy which was first proposed by Goodfellow et al. (2014). We use $\widehat{G}_{\mathrm{KL}}$ and $\widehat{G}_{\mathrm{WA}}$ to represent these two DNN-based deep generators, they can be trained by the below formula:

$$(\widehat{G}_{\mathrm{KL}}, \widehat{D}_{\mathrm{KL}}) = \arg \min_{G_\rho \in \mathcal{F}'_{\mathrm{DNN,G}}} \arg \max_{D_\phi \in \mathcal{F}'_{\mathrm{DNN,D}}} \frac{1}{n} \sum_{i=1}^{n} D_\phi(G_\rho(Z_i, X_i), X_i) - \frac{1}{n} \sum_{i=1}^{n} \exp(D_\phi(Y_i, X_i));$$

$$(\widehat{G}_{\mathrm{WA}}, \widehat{D}_{\mathrm{WA}}) = \arg \min_{G_\rho \in \mathcal{F}_{\mathrm{DNN,G}}} \arg \max_{D_\phi \in \mathcal{F}_{\mathrm{DNN,D}}} \frac{1}{n} \sum_{i=1}^{n} D_\phi(G_\rho(Z_i, X_i), X_i) - \frac{1}{n} \sum_{i=1}^{n} D_\phi(Y_i, X_i).$$

- The objective functions are based on variants of KL-divergence and Wasserstein-1 distance;
- D_ϕ is the discriminator/critic trained together with generator G_ρ adversarially;
- $\mathcal{F}_{\cdot,\cdot}$ and $\mathcal{F}'_{\cdot,\cdot}$ represent appropriate DNN classes.

Simulation setting for PI

Simulation setting for PI

We take the below model to generate n training and T test data:

$$Y_i = X_{i,1}^2 + \exp\left(X_{i,2} + X_{i,3}/3\right) + X_{i,4} - X_{i,5} + \left(0.5 + X_{i,2}^2/2 + X_{i,5}^2/2\right) \cdot \varepsilon_i.$$

Simulation setting for PI

We take the below model to generate n training and T test data:

$$Y_i = X_{i,1}^2 + \exp\left(X_{i,2} + X_{i,3}/3\right) + X_{i,4} - X_{i,5} + \left(0.5 + X_{i,2}^2/2 + X_{i,5}^2/2\right) \cdot \varepsilon_i.$$

We consider 4 types of PI: QPI and PPI based on our method, PI-KL (QPI based on $\widehat{G}_{\mathrm{KL}}$) and PI-WA (QPI based on $\widehat{G}_{\mathrm{WA}}$).

Simulation setting for PI

We take the below model to generate n training and T test data:

$$Y_i = X_{i,1}^2 + \exp\left(X_{i,2} + X_{i,3}/3\right) + X_{i,4} - X_{i,5} + \left(0.5 + X_{i,2}^2/2 + X_{i,5}^2/2\right) \cdot \varepsilon_i.$$

We consider 4 types of PI: QPI and PPI based on our method, PI-KL (QPI based on $\widehat{G}_{\mathrm{KL}}$) and PI-WA (QPI based on $\widehat{G}_{\mathrm{WA}}$).

To simplify notation, we denote all PIs by $\widehat{\mathcal{I}}$ and we consider two coverage rates under different conditioning levels:

$$\mathrm{CV}_1 := P(Y_0 \in \widehat{\mathcal{I}})\ ;\ \mathrm{CV}_2 := P(Y_0 \in \widehat{\mathcal{I}} | \boldsymbol{x}_0).$$

Simulation setting for PI

We take the below model to generate n training and T test data:

$$Y_i = X_{i,1}^2 + \exp\left(X_{i,2} + X_{i,3}/3\right) + X_{i,4} - X_{i,5} + \left(0.5 + X_{i,2}^2/2 + X_{i,5}^2/2\right) \cdot \varepsilon_i.$$

We consider 4 types of PI: QPI and PPI based on our method, PI-KL (QPI based on $\widehat{G}_{\mathrm{KL}}$) and PI-WA (QPI based on $\widehat{G}_{\mathrm{WA}}$).

To simplify notation, we denote all PIs by $\widehat{\mathcal{I}}$ and we consider two coverage rates under different conditioning levels:

$$\mathrm{CV}_1 := P(Y_0 \in \widehat{\mathcal{I}})\ ;\ \mathrm{CV}_2 := P(Y_0 \in \widehat{\mathcal{I}} | \boldsymbol{x}_0).$$

We approximate CV_2 by $\frac{1}{K}\sum_{k=1}^{K} P(Y_0 \in \widehat{\mathcal{I}} | \boldsymbol{x}_0, \{(X_i^k, Y_i^k)\}_{i=1}^n)$; $\{(X_i^k, Y_i^k)\}_{i=1}^n$ is the k-th training sets.

Simulation setting for PI

We take the below model to generate n training and T test data:

$$Y_i = X_{i,1}^2 + \exp\left(X_{i,2} + X_{i,3}/3\right) + X_{i,4} - X_{i,5} + \left(0.5 + X_{i,2}^2/2 + X_{i,5}^2/2\right) \cdot \varepsilon_i.$$

We consider 4 types of PI: QPI and PPI based on our method, PI-KL (QPI based on $\widehat{G}_{\mathrm{KL}}$) and PI-WA (QPI based on $\widehat{G}_{\mathrm{WA}}$).

To simplify notation, we denote all PIs by $\widehat{\mathcal{I}}$ and we consider two coverage rates under different conditioning levels:

$$\mathrm{CV}_1 := P(Y_0 \in \widehat{\mathcal{I}})\ ;\ \mathrm{CV}_2 := P(Y_0 \in \widehat{\mathcal{I}} | \boldsymbol{x}_0).$$

We approximate CV_2 by $\frac{1}{K}\sum_{k=1}^{K} P(Y_0 \in \widehat{\mathcal{I}} | \boldsymbol{x}_0, \{(X_i^k, Y_i^k)\}_{i=1}^n)$; $\{(X_i^k, Y_i^k)\}_{i=1}^n$ is the k-th training sets.
We approximate CV_1 by $\frac{1}{T}\sum_{t=1}^{T} P(Y_0 \in \widehat{\mathcal{I}} | \boldsymbol{x}_t)$; $\boldsymbol{x}_t$ is the t-th test point.

Simulation setting

Simulation setting

We take the same optimization algorithm RMSProp for all methods.

Simulation setting

We take the same optimization algorithm RMSProp for all methods.

We take $T = 2000$; $S = 10000$; $K = 200$; $\alpha = 0.05$ to evaluate different methods.

Simulation setting

We take the same optimization algorithm RMSProp for all methods.

We take $T = 2000$; $S = 10000$; $K = 200$; $\alpha = 0.05$ to evaluate different methods.

We take same DNN structure for all methods.

Simulation setting

We take the same optimization algorithm RMSProp for all methods.

We take $T = 2000$; $S = 10000$; $K = 200$; $\alpha = 0.05$ to evaluate different methods.

We take same DNN structure for all methods.

We also consider the sample standard deviation $\hat{\sigma}_{\mathrm{PI}}$ of 2000 CV_2. Besides, we present the average length (AL) of different PIs.

Simulation results of CV_1

Simulation results of CV_1

Table 1: Simulation results of CV_1 with varying n and p.

	CV_1	AL	CV_1	AL	CV_1	AL
p = 5	n = 200		n = 500		n = 2000	
QPI	0.861(0.170)	5.487(1.054)	0.927(0.110)	6.734(1.463)	0.787(0.177)	3.621(0.855)
PPI	0.893(0.139)	6.208(1.384)	0.941(0.095)	7.258(1.808)	0.789(0.173)	3.728(0.959)
PI-KL	0.842(0.193)	5.496(0.861)	0.869(0.157)	5.434(1.218)	0.913(0.104)	5.670(2.282)
PI-WA	0.852(0.181)	5.439(0.907)	0.882(0.150)	5.970(2.030)	0.899(0.105)	5.365(1.996)
p = 10						
QPI	0.928(0.129)	7.497(0.720)	0.949(0.094)	8.194(0.950)	0.855(0.157)	4.474(0.817)
PPI	**0.944(0.105)**	8.103(1.072)	0.961(0.076)	8.623(1.325)	0.855(0.154)	4.546(0.953)
PI-KL	0.900(0.133)	6.701(0.835)	0.925(0.119)	6.806(0.933)	0.928(0.099)	5.882(1.403)
PI-WA	0.898(0.146)	6.757(0.719)	0.933(0.116)	7.545(1.340)	0.934(0.100)	6.199(1.880)
p = 15						
QPI	0.915(0.137)	7.408(0.669)	0.945(0.097)	7.430(0.949)	0.915(0.123)	5.895(0.647)
PPI	0.930(0.119)	7.760(0.936)	**0.953(0.085)**	7.749(1.172)	0.916(0.121)	5.971(0.807)
PI-KL	0.909(0.136)	7.427(0.817)	0.949(0.095)	8.082(1.068)	0.943(0.089)	6.556(1.491)
PI-WA	0.901(0.137)	6.797(0.687)	0.950(0.095)	7.972(1.312)	0.947(0.088)	6.778(1.541)
p = 20						
QPI	0.879(0.172)	6.726(0.485)	0.959(0.085)	8.830(0.683)	0.940(0.102)	6.849(0.562)
PPI	0.893(0.154)	6.941(0.702)	0.966(0.073)	9.100(0.950)	0.942(0.097)	6.925(0.759)
PI-KL	0.923(0.126)	7.799(0.842)	0.954(0.087)	8.311(0.861)	0.946(0.093)	6.806(1.097)
PI-WA	0.910(0.140)	7.402(0.698)	0.945(0.099)	8.011(0.800)	0.946(0.092)	6.804(1.534)
p = 25						
QPI	0.871(0.172)	7.020(0.287)	0.961(0.088)	9.633(0.645)	0.946(0.099)	7.296(0.475)
PPI	0.884(0.160)	7.189(0.548)	0.967(0.078)	9.881(0.938)	**0.948(0.095)**	7.370(0.695)
PI-KL	0.907(0.142)	7.370(0.618)	0.954(0.090)	8.670(0.813)	0.945(0.093)	6.915(1.009)
PI-WA	0.897(0.151)	7.071(0.510)	0.960(0.081)	8.514(0.942)	0.944(0.097)	7.117(1.491)

Simulation results of CV_2: PPI vs PI-KL

Simulation results of CV_2: PPI vs PI-KL

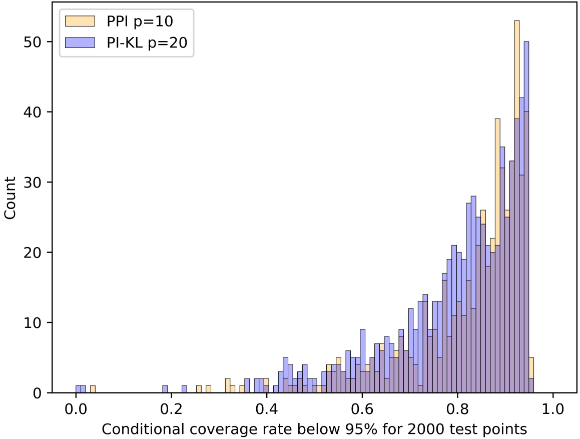


Figure 2: Histograms of all undercoverage CV_2 (CV_2 less than nominal level 95%) of PPI and PI-KL.

Simulation results of CV_2: PPI vs PI-WA

Simulation results of CV_2: PPI vs PI-WA

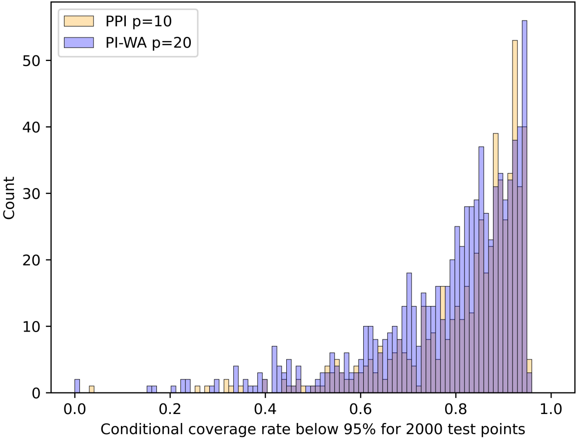


Figure 3: Histograms of all undercoverage CV_2 (CV_2 less than nominal level 95%) of PPI and PI-WA.

Simulation results of CV_2: PPI vs QPI

Simulation results of CV_2: PPI vs QPI

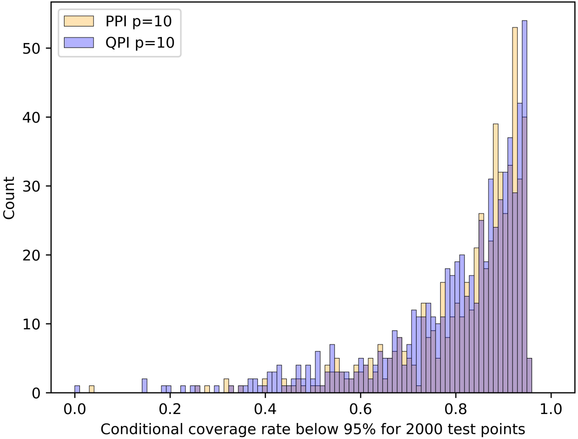


Figure 4: Histograms of all undercoverage CV_2 (CV_2 less than nominal level 95%) of PPI and QPI.

Theoretical explanations

Under further assumptions about the joint distribution $P_{\boldsymbol{X},Y}$, we have:

Theorem 6: Theoretical understanding of PPI with DNN

For an appropriate sequence of sets Ω_n, such that $\mathbb{P}((\{\boldsymbol{X}_i, Y_i, Z_i\}_{i=1}^n) \notin \Omega_n) = o(1)$, PPI can capture the estimation variability under $S \to \infty$ in an appropriate rate for each n, when $n \to \infty$. Furthermore,

$$\sup_y \left| \widehat{F}_{\widehat{H}(\boldsymbol{x}_0,Z)} \star \phi_\sigma(y) - F_{Y|\boldsymbol{x}_0} \star \phi_\sigma(y) \right| \leq \sup_y \left| \widehat{F}_{\widehat{H}(\boldsymbol{x}_0,Z)}(y) - F_{Y|\boldsymbol{x}_0}(y) \right| \text{ with probability 1;}$$

$\widehat{F}_{\widehat{H}(\boldsymbol{x}_0,Z)}$ is the empirical distribution of $\{\widehat{H}(\boldsymbol{x}_0, Z_i)\}_{i=1}^S$; $\star$ is the convolution operator; ϕ_σ is the density function of the normal distribution $N(0, \sigma^2)$.

Scalable Subsampling for Deep Neural Networks training[9]

[9]This part is based on:

- Wu, K. and Politis, D.N., Scalable Subsampling Inference of Deep Neural Networks. *ACM/IMS Journal of Data Science* 2025, 2(1), 1–29.

Motivation

From the part II, we have seen the power of DNN. However, there are two crucial problems that need more thoughts:

Motivation

From the part II, we have seen the power of DNN. However, there are two crucial problems that need more thoughts:

- **The estimation error bound of DNN:** According to the cornerstone of Stone (1982), it is well-known that the optimal convergence rate of the estimation for a p-times continuously differentiable function of a d-dimensional argument is $n^{2p/(2p+d)}$. The existing practically achievable DNN estimator possesses a sub-optimal rate.

Motivation

From the part II, we have seen the power of DNN. However, there are two crucial problems that need more thoughts:

- **The estimation error bound of DNN:** According to the cornerstone of Stone (1982), it is well-known that the optimal convergence rate of the estimation for a p-times continuously differentiable function of a d-dimensional argument is $n^{2p/(2p+d)}$. The existing practically achievable DNN estimator possesses a sub-optimal rate.
- **The computational burden of training DNN:** DNN has been developed rapidly fueled by ever-increasing amounts of data. However, training a large DNN with a huge sample size requires heavy computational resources.

Motivation

From the part II, we have seen the power of DNN. However, there are two crucial problems that need more thoughts:

- **The estimation error bound of DNN:** According to the cornerstone of Stone (1982), it is well-known that the optimal convergence rate of the estimation for a p-times continuously differentiable function of a d-dimensional argument is $n^{2p/(2p+d)}$. The existing practically achievable DNN estimator possesses a sub-optimal rate.
- **The computational burden of training DNN:** DNN has been developed rapidly fueled by ever-increasing amounts of data. However, training a large DNN with a huge sample size requires heavy computational resources.

Motivation

From the part II, we have seen the power of DNN. However, there are two crucial problems that need more thoughts:

- **The estimation error bound of DNN:** According to the cornerstone of Stone (1982), it is well-known that the optimal convergence rate of the estimation for a p-times continuously differentiable function of a d-dimensional argument is $n^{2p/(2p+d)}$. The existing practically achievable DNN estimator possesses a sub-optimal rate.
- **The computational burden of training DNN:** DNN has been developed rapidly fueled by ever-increasing amounts of data. However, training a large DNN with a huge sample size requires heavy computational resources.

We want to kill two birds with one stone.

Intuition

Intuition

- **Variance reduction:** To improve the convergence rate of an estimator, we can try to decrease its variance if its bias is acceptable. This is inspired by *bagging* method of Breiman (1996).

Intuition

- **Variance reduction:** To improve the convergence rate of an estimator, we can try to decrease its variance if its bias is acceptable. This is inspired by *bagging* method of Breiman (1996).
- **Build estimators on subsamples:** To relieve the computational burden, we can repeat the estimation with subsamples if the computation with the whole sample is infeasible. This approach shares a general divide-and-conquer idea originally proposed in computer science (Cormen et al., 1989).

Intuition

- **Variance reduction:** To improve the convergence rate of an estimator, we can try to decrease its variance if its bias is acceptable. This is inspired by *bagging* method of Breiman (1996).
- **Build estimators on subsamples:** To relieve the computational burden, we can repeat the estimation with subsamples if the computation with the whole sample is infeasible. This approach shares a general divide-and-conquer idea originally proposed in computer science (Cormen et al., 1989).

Intuition

- **Variance reduction:** To improve the convergence rate of an estimator, we can try to decrease its variance if its bias is acceptable. This is inspired by *bagging* method of Breiman (1996).
- **Build estimators on subsamples:** To relieve the computational burden, we can repeat the estimation with subsamples if the computation with the whole sample is infeasible. This approach shares a general divide-and-conquer idea originally proposed in computer science (Cormen et al., 1989).

Example: A simple implementation

Suppose we need $O(n^\zeta)$ operations to compute one estimator $\hat{\theta}_n$ for true parameter θ. If we consider $q = O(n/b)$ number of estimations $\hat{\theta}_{b,i}$ on subsamples with size b for $i = 1, \ldots, q$, we can take $\bar{\theta}_{b,n,\mathrm{SS}} := \frac{1}{q}\sum_{i=1}^{q} \hat{\theta}_{b,i}$ to approximate $\hat{\theta}_n$. Then, only $O\left(nb^{\zeta-1}\right)$ operations are needed.

Scalable subsampling

Scalable subsampling

Recently, Politis (2024) proposed one type of non-stochastic *scalable subsampling* technique.

Scalable subsampling

Recently, Politis (2024) proposed one type of non-stochastic *scalable subsampling* technique.

Assume that we observe the sample $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n\}$; $\boldsymbol{U}_i$ represents $(\boldsymbol{X}_i, Y_i)$; then, scalable subsampling relies on $q = \lfloor (n-b)/h \rfloor + 1$ number of subsamples $B_1, \ldots, B_q$ where $B_j = \{\boldsymbol{U}_{(j-1)h+1}, \ldots, \boldsymbol{U}_{(j-1)h+b}\}$; here, $\lfloor \cdot \rfloor$ denotes the floor function, and h controls the amount of overlap (or separation) between B_j and B_{j+1}.

Scalable subsampling

Recently, Politis (2024) proposed one type of non-stochastic *scalable subsampling* technique.

Assume that we observe the sample $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n\}$; $\boldsymbol{U}_i$ represents $(\boldsymbol{X}_i, Y_i)$; then, scalable subsampling relies on $q = \lfloor (n-b)/h \rfloor + 1$ number of subsamples $B_1, \ldots, B_q$ where $B_j = \{\boldsymbol{U}_{(j-1)h+1}, \ldots, \boldsymbol{U}_{(j-1)h+b}\}$; here, $\lfloor \cdot \rfloor$ denotes the floor function, and h controls the amount of overlap (or separation) between B_j and B_{j+1}.

Tuning b and h can make scalable subsampling samples have different overlapping rates:

Scalable subsampling

Recently, Politis (2024) proposed one type of non-stochastic *scalable subsampling* technique.

Assume that we observe the sample $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n\}$; $\boldsymbol{U}_i$ represents $(\boldsymbol{X}_i, Y_i)$; then, scalable subsampling relies on $q = \lfloor (n-b)/h \rfloor + 1$ number of subsamples $B_1, \ldots, B_q$ where $B_j = \{\boldsymbol{U}_{(j-1)h+1}, \ldots, \boldsymbol{U}_{(j-1)h+b}\}$; here, $\lfloor \cdot \rfloor$ denotes the floor function, and h controls the amount of overlap (or separation) between B_j and B_{j+1}.

Tuning b and h can make scalable subsampling samples have different overlapping rates:

- if $h = 1$, the overlap is the maximum possible;

Scalable subsampling

Recently, Politis (2024) proposed one type of non-stochastic *scalable subsampling* technique.

Assume that we observe the sample $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n\}$; $\boldsymbol{U}_i$ represents $(\boldsymbol{X}_i, Y_i)$; then, scalable subsampling relies on $q = \lfloor (n-b)/h \rfloor + 1$ number of subsamples $B_1, \ldots, B_q$ where $B_j = \{\boldsymbol{U}_{(j-1)h+1}, \ldots, \boldsymbol{U}_{(j-1)h+b}\}$; here, $\lfloor \cdot \rfloor$ denotes the floor function, and h controls the amount of overlap (or separation) between B_j and B_{j+1}.

Tuning b and h can make scalable subsampling samples have different overlapping rates:

- if $h = 1$, the overlap is the maximum possible;
- if $h = 0.2b$, there is 80% overlap between B_j and B_{j+1};

Scalable subsampling

Recently, Politis (2024) proposed one type of non-stochastic *scalable subsampling* technique.

Assume that we observe the sample $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n\}$; $\boldsymbol{U}_i$ represents $(\boldsymbol{X}_i, Y_i)$; then, scalable subsampling relies on $q = \lfloor (n-b)/h \rfloor + 1$ number of subsamples $B_1, \ldots, B_q$ where $B_j = \{\boldsymbol{U}_{(j-1)h+1}, \ldots, \boldsymbol{U}_{(j-1)h+b}\}$; here, $\lfloor \cdot \rfloor$ denotes the floor function, and h controls the amount of overlap (or separation) between B_j and B_{j+1}.

Tuning b and h can make scalable subsampling samples have different overlapping rates:

- if $h = 1$, the overlap is the maximum possible;
- if $h = 0.2b$, there is 80% overlap between B_j and B_{j+1};
- if $h = b$, there is no overlap between B_j and B_{j+1} but these two blocks are adjacent;

Scalable subsampling

Recently, Politis (2024) proposed one type of non-stochastic *scalable subsampling* technique.

Assume that we observe the sample $\{\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n\}$; $\boldsymbol{U}_i$ represents $(\boldsymbol{X}_i, Y_i)$; then, scalable subsampling relies on $q = \lfloor (n-b)/h \rfloor + 1$ number of subsamples $B_1, \ldots, B_q$ where $B_j = \{\boldsymbol{U}_{(j-1)h+1}, \ldots, \boldsymbol{U}_{(j-1)h+b}\}$; here, $\lfloor \cdot \rfloor$ denotes the floor function, and h controls the amount of overlap (or separation) between B_j and B_{j+1}.

Tuning b and h can make scalable subsampling samples have different overlapping rates:

- if $h = 1$, the overlap is the maximum possible;
- if $h = 0.2b$, there is 80% overlap between B_j and B_{j+1};
- if $h = b$, there is no overlap between B_j and B_{j+1} but these two blocks are adjacent;
- if $h = 1.2b$, there is a block of about $0.2b$ data points that separate the blocks B_j and B_{j+1}.

Scalable subsampling estimation of DNN

Define the scalable subsampling DNN estimator as $\overline{f}_{\mathrm{DNN}}(\boldsymbol{X}) = \frac{1}{q}\sum_{j=1}^{q}\widehat{f}_{\mathrm{DNN},b,j}(\boldsymbol{X})$; here, $q = \lfloor (n-b)/h \rfloor + 1$, and $\widehat{f}_{\mathrm{DNN},b,j}(\cdot)$ is trained DNN with the j-th subsample B_j.

Scalable subsampling estimation of DNN

Define the scalable subsampling DNN estimator as $\overline{f}_{\text{DNN}}(\boldsymbol{X}) = \frac{1}{q}\sum_{j=1}^{q}\widehat{f}_{\text{DNN},b,j}(\boldsymbol{X})$; here, $q = \lfloor (n-b)/h \rfloor + 1$, and $\widehat{f}_{\text{DNN},b,j}(\cdot)$ is trained DNN with the j-th subsample B_j.
Under below assumptions:

Scalable subsampling estimation of DNN

Define the scalable subsampling DNN estimator as $\overline{f}_{\text{DNN}}(\boldsymbol{X}) = \frac{1}{q}\sum_{j=1}^{q} \widehat{f}_{\text{DNN},b,j}(\boldsymbol{X})$; here, $q = \lfloor (n-b)/h \rfloor + 1$, and $\widehat{f}_{\text{DNN},b,j}(\cdot)$ is trained DNN with the j-th subsample B_j.

Under below assumptions:

- $\mathbb{E}(\widehat{f}_{\text{DNN}}(\boldsymbol{x}) - f(\boldsymbol{x})) = O(n^{-\Lambda/2})$ uniformly for all $\boldsymbol{x} \in \mathcal{X}$ for some constant $\Lambda > 0$;

Scalable subsampling estimation of DNN

Define the scalable subsampling DNN estimator as $\overline{f}_{\mathrm{DNN}}(\boldsymbol{X}) = \frac{1}{q}\sum_{j=1}^{q}\widehat{f}_{\mathrm{DNN},b,j}(\boldsymbol{X})$; here, $q = \lfloor (n-b)/h \rfloor + 1$, and $\widehat{f}_{\mathrm{DNN},b,j}(\cdot)$ is trained DNN with the j-th subsample B_j.
Under below assumptions:

- $\mathbb{E}(\widehat{f}_{\mathrm{DNN}}(\boldsymbol{x}) - f(\boldsymbol{x})) = O(n^{-\Lambda/2})$ uniformly for all $\boldsymbol{x} \in \mathcal{X}$ for some constant $\Lambda > 0$;
- The bias exponent in the assumption above satisfies the inequality: $\Lambda > \frac{\xi}{\xi+d}$.

Scalable subsampling estimation of DNN

Define the scalable subsampling DNN estimator as $\overline{f}_{\mathrm{DNN}}(\boldsymbol{X}) = \frac{1}{q}\sum_{j=1}^{q}\widehat{f}_{\mathrm{DNN},b,j}(\boldsymbol{X})$; here, $q = \lfloor (n-b)/h \rfloor + 1$, and $\widehat{f}_{\mathrm{DNN},b,j}(\cdot)$ is trained DNN with the j-th subsample B_j.
Under below assumptions:

- $\mathbb{E}(\widehat{f}_{\mathrm{DNN}}(\boldsymbol{x}) - f(\boldsymbol{x})) = O(n^{-\Lambda/2})$ uniformly for all $\boldsymbol{x} \in \mathcal{X}$ for some constant $\Lambda > 0$;
- The bias exponent in the assumption above satisfies the inequality: $\Lambda > \frac{\xi}{\xi+d}$.

Scalable subsampling estimation of DNN

Define the scalable subsampling DNN estimator as $\overline{f}_{\mathrm{DNN}}(\boldsymbol{X}) = \frac{1}{q}\sum_{j=1}^{q} \widehat{f}_{\mathrm{DNN},b,j}(\boldsymbol{X})$; here, $q = \lfloor (n-b)/h \rfloor + 1$, and $\widehat{f}_{\mathrm{DNN},b,j}(\cdot)$ is trained DNN with the j-th subsample B_j.
Under below assumptions:

- $\mathbb{E}(\widehat{f}_{\mathrm{DNN}}(\boldsymbol{x}) - f(\boldsymbol{x})) = O(n^{-\Lambda/2})$ uniformly for all $\boldsymbol{x} \in \mathcal{X}$ for some constant $\Lambda > 0$;
- The bias exponent in the assumption above satisfies the inequality: $\Lambda > \frac{\xi}{\xi+d}$.

we have

Theorem 7:

Under appropriate assumptions, let $b = h = n^{\beta}$; $\beta = \frac{1}{1+\Lambda-\frac{\xi}{\xi+d}}$. Then, with probability at least $(1-\exp(-n^{\frac{d}{\xi+d}}\log^6 n))^q$:

$$\left\|\overline{f}_{\mathrm{DNN}} - f\right\|_{L_2(X)}^2 \leq n^{\frac{-\Lambda}{\Lambda+\frac{d}{\xi+d}}}\mathcal{L}(n);$$

where $\mathcal{L}(n)$ is a slowly varying function involving a constant and all $\log(n)$ terms.

Simulation setting

To perform simulations, we consider below models:

- Model-1: $Y_i = X_{i,1}^2 + \sin(X_{i,2} + X_{i,3}) + \epsilon$, where $\boldsymbol{X} \sim N(0, \boldsymbol{I}_3)$; $\epsilon \sim N(0, 1)$;
- Model-2: $Y_i = X_{i,1}^2 + \sin(X_{i,2} + X_{i,3}) + \exp(-|X_{i,4} + X_{i,5}|) + \epsilon$, where $\boldsymbol{X} \sim N(0, \boldsymbol{I}_3)$; $\epsilon \sim N(0, 1)$.

To be consistent with folk wisdom, we build $\widehat{f}_{\mathrm{DNN},b,i}$ with a relatively large depth to decrease the bias but also guarantee that the DNN estimator is in the under-parameterized region. We also consider other 5 DNN estimators trained with the whole sample:

(1) A DNN possesses the same depth and width as $\widehat{f}_{\mathrm{DNN},b,i}$. We denote it "S-DNN";

(2) A DNN possesses the same depth as $\widehat{f}_{\mathrm{DNN},b,i}$, but a larger width so that its parameter size is close to the sample size. We denote it "DNN-deep-1";

(3) A DNN possesses the same depth as $\widehat{f}_{\mathrm{DNN},b,i}$, but a larger width so that its parameter size is close to half of the sample size. We denote it "DNN-deep-2";

(4) A DNN possesses only one hidden layer, but a larger width so that its parameter size is close to the sample size. We denote it "DNN-wide-1";

(5) A DNN possesses only one hidden layer, but a larger width so that its parameter size is close to half of the sample size. We denote it "DNN-wide-2".

Hyperparameter setting and metric

To evaluate the performance of different DNN estimators, we apply the empirical MSE and MSPE criteria:

$$\text{MSE}:=\frac{1}{n}\sum_{i=1}^{n}(\widetilde{f}_{\text{DNN}}(\boldsymbol{x}_i)-f(\boldsymbol{x}_i))^2\ ;\ \text{MSPE}:=\frac{1}{N}\sum_{i=1}^{N}(\widetilde{f}_{\text{DNN}}(\boldsymbol{x}_{0,i})-f(\boldsymbol{x}_{0,i}))^2;$$

here $\widetilde{f}_{\text{DNN}}(\cdot)$ represents different DNN estimators and $f(\cdot)$ is the true regression function; $\{\boldsymbol{x}_i, y_i\}_{i=1}^{n}$ are training data; $\{\boldsymbol{x}_{0,i}, y_{0,i}\}_{i=1}^{N}$ are test data; we take $N = 2 \cdot 10^5$.

Simulation results are averaged from 200 replications.

Simulation results

Simulation results

Table 2: MSE/MSPE and Training Time (in seconds) of different DNN models

Estimator:	SS-DNN	S-DNN	DNN-deep-1	DNN-deep-2	DNN-wide-1	DNN-wide-2
Model-1, $n = 10^4$						
Width	[15,15,15]	[15,15,15]	[65,65,65]	[45,45,45]	[2000]	[1000]
MSE	0.0296	0.0536	0.0533	0.0522	0.0426	0.0431
MSPE	0.0310	0.0564	0.0572	0.0570	0.0453	0.0449
Training Time	353	379	561	468	483	363
Model-2, $n = 10^4$						
Width	[15,15,15]	[15,15,15]	[65,65,65]	[45,45,45]	[2000]	[1000]
MSE	0.0757	0.0830	0.1076	0.0980	0.0729	0.0728
MSPE	0.0790	0.0875	0.1114	0.1045	0.0754	0.0749
Training Time	359	376	560	471	551	394
Model-2, $n = 2 \cdot 10^4$						
Width	[20,20,20]	[20,20,20]	[95,95,95]	[65,65,65]	[2800]	[1400]
MSE	0.0490	0.0653	0.0686	0.0675	0.0635	0.0635
MSPE	0.0502	0.0670	0.0692	0.0689	0.0623	0.0626
Training Time	748	775	1684	1198	1549	998

Acknowledgements I

Acknowledgements I

I would like to thank Professor Dimitris Politis. As my advisor, my friend, and my life guider, he always supported my research and gave me valuable suggestions. Moreover, he led me to a path I never imagined about.

Acknowledgements II

I would like to thank my committee members, Ery Arias-Castro, Yian Ma and Danna Zhang.

Acknowledgements II

I would like to thank my committee members, Ery Arias-Castro, Yian Ma and Danna Zhang.

I would also like to thank scholars I have been honored and humbled to work alongside: Wagner Barreto-Souza, Rangan Gupta, Nicholas Jacobson, Sayar Karmakar, Christian Pierdzioch, Oisín Ryan, and Raanju Sundararajan.

Acknowledgements III

I would like to thank all my friends I met during my Ph.D. life.

Acknowledgements III

I would like to thank all my friends I met during my Ph.D. life.

I would also like to thank the administrative staff, particularly Scott Rollans and Mark Whelan, for their constant support.

Acknowledgements IV

I would like to thank my parents. This talk is dedicated to them.

Thank you!

References

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.

Bloem-Reddy, B., Whye, Y., et al. (2020). Probabilistic symmetries and invariant neural networks. *Journal of Machine Learning Research*, 21(90):1–61.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24:123–140.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (1989). *Introduction to algorithms*. MIT press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Liu, S., Zhou, X., Jiao, Y., and Huang, J. (2021). Wasserstein generative learning of conditional distribution. *arXiv preprint arXiv:2112.10039*.

Pang, T., Yang, X., Dong, Y., Su, H., and Zhu, J. (2020). Bag of tricks for adversarial training. *arXiv preprint arXiv:2010.00467*.

Politis, D. N. (2015). *Model-free prediction in regression*. Springer.

Politis, D. N. (2024). Scalable subsampling: computation, aggregation and inference. *Biometrika*, 111(1):347–354.

References

Stone, C. J. (1982). Optimal global rates of convergence for nonparametric regression. *The annals of statistics*, 10(4):1040–1053.

Wang, Y. and Politis, D. N. (2021). Model-free bootstrap and conformal prediction in regression: Conditionality, conjecture testing, and pertinent prediction intervals. *arXiv preprint arXiv:2109.12156*.

Yarotsky, D. and Zhevnerchuk, A. (2020). The phase diagram of approximation rates for deep neural networks. *Advances in neural information processing systems*, 33:13005–13015.

Zhou, X., Jiao, Y., Liu, J., and Huang, J. (2023). A deep generative approach to conditional sampling. *Journal of the American Statistical Association*, 118(543):1837–1848.

Backup I: Additional slides

Estimation of $G(\cdot,\cdot)$

Define the optimal estimator:

$$H_0 = \arg\min_H \mathbb{E}\,(Y - H(\boldsymbol{X}, Z))^2 = \arg\min_H \mathcal{R};$$

A simple decomposition shows that:

$$\mathbb{E}\,(Y - H(\boldsymbol{X}, Z))^2 = \mathbb{E}\,(Y - G(\boldsymbol{X}, Z) + G(\boldsymbol{X}, Z) - H(\boldsymbol{X}, Z))^2\,.$$

Thus, the optimal H_0 is unique almost surely and we can take it as the continuous counterpart of G.

Difference between traditional MSE risk

Recall that the risk for standard regression tasks is

$$\mathbb{E}[(Y - h(\boldsymbol{X}))^2] := \mathcal{R}_s.$$

Table 3: Comparison between standard regression risk and our risk

	Geometry	σ-algebra
$\mathcal{R}_s$	The optimal estimator is the projection of Y onto a closed subspace $\mathcal{S}_X$ of L_2 consisting of all random variables which can be written in a function of $\boldsymbol{X}$.	$\mathbb{E}(Y\|\boldsymbol{X})$ is $\mathcal{D}_X$-measurable.[10]
$\mathcal{R}$	The optimal estimator is a projection of Y onto an extended version of $\mathcal{S}_X$ by random variable Z.	$Y \overset{a.s.}{=} G(\boldsymbol{X}, Z)$ is $\mathcal{D}_{(X,Z)}$-measurable.

[10] $\mathcal{D}_X$ is the σ-algebra generated by $\boldsymbol{X}$; $\mathbb{E}(Y|\boldsymbol{X})$ could also equal to Y a.s. if Y is $\mathcal{D}_X$-measurable, e.g., $\mathbb{E}(Y|Y) = Y$.

Intuition behind our Deep limit model-free prediction algorithm

We provide a toy example to explain the motivation of our training procedure.

Remark: An illustration example

Suppose we need to estimate the coefficient β of a linear regression model $Y = \beta^T \cdot \boldsymbol{X} + \epsilon$ with a fixed design based on samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$; here, ϵ has zero mean and finite variance.

- OLS: $\widehat{\beta} := \arg\min_\beta \frac{1}{n} \sum_{i=1}^n (y_i - \beta_i^T \cdot \boldsymbol{x}_i)^2$ which is consistent under standard conditions.
- Variant of OLS: $\widehat{\beta}^* := \arg\min_\beta \frac{1}{n} \sum_{i=1}^n (y_i - (\beta_i^T \cdot \boldsymbol{x}_i + \epsilon_i^*))^2$ where $\{\epsilon_i^*\}_{i=1}^n$ are independent of $\boldsymbol{X}$ and can be generated from any distribution with mean zero and finite variance.

$\widehat{\beta}^*$ is also consistent although $\widehat{\beta}$ would generally be more efficient.

Analogously, our DNN-based estimation $\widehat{H}^*$ converges to H_0 in the mean square sense even using the artificially generated $\{Z_i^*\}_{i=1}^n$.

Estimation ability of DNN

The estimation error of $\widehat{H}$ can be decomposed into two sources:

(1) The stochastic error, which measures the difference between $\widehat{H}$ and the best estimator H^* in a DNN class $\mathcal{F}_{\mathrm{DNN}}$; $H^* := \arg\min_{H \in \mathcal{F}_{\mathrm{DNN}}} \|H_0 - H\|_\infty$;

(2) The approximation error, which measures the difference between H_0 and H^* in a DNN class $\mathcal{F}_{\mathrm{DNN}}$.

Preliminary comparisons

Table 4: Comparison between different DNN-based methods

	$\widehat{H}$	$\widehat{G}_{\text{KL}}, \widehat{G}_{\text{WA}}$
Stability	The training process is more stable and directly due to the MSE-like loss function.	The training process is sensitive to the training setting and depends on D_ϕ being optimal given current step G_ρ.
Metrics	The optimization corresponds to minimizing the Kolmogorov distance between two distributions.	The optimization corresponds to minimizing KL-divergence and Wasserstein-1 distance[11].
Computability	Only one DNN need to be trained.	Two DNNs need to be trained adversarially.

[11] The "distance" between two distributions converges to 0 under the metric of Wasserstein-1 distance or KL-divergence implies the convergence measured by Kolmogorov distance.

Simulation setting for optimal L_2 point prediction

We take the below model from Zhou et al. (2023) to generate n training and T test data:

$$Y_i = X_{i,1}^2 + \exp\left(X_{i,2} + X_{i,3}/3\right) + X_{i,4} - X_{i,5} + \left(0.5 + X_{i,2}^2/2 + X_{i,5}^2/2\right) \cdot \varepsilon_i;$$

where $\boldsymbol{X}_i$ and ε_i come from $N(0, \boldsymbol{I}_5)$ and $N(0,1)$ truncated to $[-5,5]^5$ and $[-5,5]$, respectively.

To predict the mean of Y conditional on $\boldsymbol{X} = \boldsymbol{x}$, we rely on $\widehat{Y}_t = \frac{1}{S}\sum_{s=1}^{S} \widehat{\Pi}(\boldsymbol{x}_t, Z_s)$; $Z_s \sim N(0, I_p)$; $\boldsymbol{x}_t$ is the t-th observation of the test data; $\widehat{\Pi}$ represents trained model $\widehat{H}$, $\widehat{G}_{\text{KL}}$ or $\widehat{G}_{\text{WA}}$.

To measure different methods, we repeat the simulations K times and consider the metric:

$$\widetilde{L} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{K}\sum_{k=1}^{K}(Y_{t,L_2} - \widehat{Y}_{k,t})^2;$$

where Y_{t,L_2} is the oracle L_2 optimal value of Y conditional on $\boldsymbol{x}_t$; $\widehat{Y}_{k,t}$ is the conditional L_2 point prediction based on the k-th training data.

Simulation setting

We apply the same hyperparameter setting to train all DNN.

We take $n = 2000$, $T = 2000$, $S = 10000$, $K = 200$ to compute the error metric.

For the structure of DNN, we separate the simulation studies into two groups: (a) structures of $\widehat{H}$ and $\widehat{G}$ are [35,35] and [50][12], respectively; (b) structures of $\widehat{H}$ and $\widehat{G}$ are all [35,35]. For both groups, $\widehat{D}$ takes the same structure as the previous work, i.e., [50,25].

For the benchmark method, we apply the numerical integration $\int_{\mathcal{Y}} y \hat{f}_{y|\boldsymbol{x}_t} dy$ with 1000 subdivisions to approximate $E(Y|\boldsymbol{x}_t)$; $\hat{f}_{y|\boldsymbol{x}_t}$ is the kernel conditional density estimator of Y conditional on $\boldsymbol{x}_t$.

[12] [35,35] stands for a two layers DNN and each layer has 35 neurons; [50] is the DNN structure used in Zhou et al. (2023). To simplify notations, $\widehat{G}$ represents $\widehat{G}_{\mathrm{KL}}$ or $\widehat{G}_{\mathrm{WA}}$; $\widehat{D}$ represents $\widehat{D}_{\mathrm{KL}}$ or $\widehat{D}_{\mathrm{WA}}$.

Simulation results

Table 5: Point predictions of different methods under groups (a) and (b).

	Group (a)			Group (b)		
	$\widehat{H}$	$\widehat{G}_{\mathrm{KL}}$	$\widehat{G}_{\mathrm{WA}}$	$\widehat{H}$	$\widehat{G}_{\mathrm{KL}}$	$\widehat{G}_{\mathrm{WA}}$
SGD						
$p = 1$	0.309	3.931	10.39	0.292	3.827	82.97
$p = 3$	0.298	4.009	11.10	0.285	3.762	56644
$p = 5$	0.296	4.036	40.39	0.281	3.801	12843
$p = 10$	**0.294**	4.116	182.3	**0.280**	3.812	11378
Adam						
$p = 1$	1.608	1.838	3558	1.572	1.836	14322
$p = 3$	0.832	1.105	8.480	0.843	1.549	43.48
$p = 5$	0.604	0.820	43.85	0.591	1.166	43.84
$p = 10$	**0.412**	0.495	5.523	**0.422**	0.817	14.50
RMSProp						
$p = 1$	0.960	1.767	1.910	0.973	1.620	2.326
$p = 3$	0.601	1.049	1.248	0.597	0.964	1.263
$p = 5$	0.484	0.779	0.908	0.479	0.727	0.903
$p = 10$	**0.365**	0.463	0.598	**0.352**	0.494	0.508

Note: The error metric $\widetilde{L}$ of using conditional kernel density estimation is around 1.210.

Hyperparameter setting

We apply the same hyperparameter setting to train $\widehat{H}$, $\widehat{G}_{\mathrm{KL}}$ and $\widehat{G}_{\mathrm{WA}}$: $n = 2000$; $T = 2000$; $S = 10000$; $K = 200$; $p = 1, 3, 5, 10$, $m = 20$; Learning rate: 0.001; Number of epochs: 10000.

For the optimizer of the adversarial training process, Arjovsky et al. (2017) proposed using optimizer RMSProp with Wasserstein distance is more appropriate. However, Pang et al. (2020) argued that SGD-based optimizers are better. We consider three common optimizers, SGD, Adam and RMSProp.

Remark of Theorem 3

- **PPI can capture the estimation variability**: Since the distribution of R_0^* can approximate the distribution of R_0, PPI captures the estimation variability in finite sample cases to some extent.
- **A convolution implied in predictive root**: It comes from rewriting the predictive root as $R_0 := Y_0 - \mathbb{E}(Y_0|\boldsymbol{x}_0) + \mathbb{E}(Y_0|\boldsymbol{x}_0) - \widehat{Y}_{0,L_2}$; $Y_0 - \mathbb{E}(Y_0|\boldsymbol{x}_0)$ only depends on $P_{Y|\boldsymbol{x}_0}$ and $\mathbb{E}(Y_0|\boldsymbol{x}_0) - \widehat{Y}_{0,L_2}$ is a (asymptotically shrinking) Gaussian distribution. Thus the below inequality from the previous theorem reveals that we need less data to achieve the same accuracy of the distribution estimation under this convolution approach.

$$\sup_y \left|\widehat{F}_{\widehat{H}(\boldsymbol{x}_0,Z)} \star \phi_\sigma(y) - F_{Y|\boldsymbol{x}_0} \star \phi_\sigma(y)\right| \leq \sup_y \left|\widehat{F}_{\widehat{H}(\boldsymbol{x}_0,Z)}(y) - F_{Y|\boldsymbol{x}_0}(y)\right| \text{ with probability 1.}$$

KL-divergence and Wasserstein-1 distance

- KL-divergence: if f, g are densities of the measures μ, ν with respect to a dominating measure λ,

$$d_I(\mu, \nu) := \int_{S(\mu)} f \log(f/g) d\lambda.$$

  where $S(\mu)$ is the support of μ on Ω.

- Wasserstein-1 distance: for $\Omega = \mathbb{R}$, if F, G are the distribution functions of μ, ν respectively, the Kantorovich metric is defined by

$$\begin{aligned} d_W(\mu, \nu) &:= \int_{-\infty}^{\infty} |F(x) - G(x)| dx \\ &= \int_0^1 \left| F^{-1}(t) - G^{-1}(t) \right| dt. \end{aligned}$$

Remark on scalable subsampling method

The scalable subsampling method can be applied in making point estimations and developing the estimation inference:

- **For point estimation:** Take $h = b$ as an example, as the analysis in the previous example, $O\left(nb^{\zeta-1}\right)$ operations are needed to compute $\bar{\theta}_{b,n,\mathrm{SS}}$. Moreover, we have

  $$\mathbb{E}\left(\bar{\theta}_{b,n,\mathrm{SS}}\right) = \mathbb{E}\left(\hat{\theta}_{b,1}\right) \textit{ and } \mathrm{Var}\left(\bar{\theta}_{b,n,\mathrm{SS}}\right) \leq \frac{1}{q}\mathrm{Var}\left(\hat{\theta}_{b,1}\right); q = \lfloor n/h \rfloor.$$

  Hence, if the bias of $\hat{\theta}_{b,1}$ is tolerable, $\bar{\theta}_{b,n,\mathrm{SS}}$ yields a welcome variance reduction.
- **For estimation inference:** The subsampling distribution $L_{n,b,h}(x) = q^{-1} \sum_{i=1}^{q} \mathbb{1}\left\{\tau_b g\left(\hat{\theta}_{b,i} - \hat{\theta}_n\right) \leq x\right\}$ can be used to approximate the distribution of the estimation root $J_n(x) = P\left\{\tau_n g\left(\hat{\theta}_n - \theta\right) \leq x\right\}$ under mild conditions; where $g(\cdot)$ could be the identity function or the sup-norm.

Remark on scalable subsampling estimation of DNN

- The two assumptions regarding the bias order of the DNN estimator are feasible:

Remark on scalable subsampling estimation of DNN

- The two assumptions regarding the bias order of the DNN estimator are feasible:
  (1) As revealed in Yarotsky and Zhevnerchuk (2020), the approximation ability in the uniform sup-norm of a DNN can be as fast as $\Delta^{-2\xi/d}$; Δ is the total number of parameters in a DNN. Although this rate is not instructive in practice, the desired bias order is achievable.

Remark on scalable subsampling estimation of DNN

- The two assumptions regarding the bias order of the DNN estimator are feasible:
  (1) As revealed in Yarotsky and Zhevnerchuk (2020), the approximation ability in the uniform sup-norm of a DNN can be as fast as $\Delta^{-2\xi/d}$; Δ is the total number of parameters in a DNN. Although this rate is not instructive in practice, the desired bias order is achievable.
  (2) As revealed in Belkin et al. (2019), the double-descent of the risk exists for over-parameterized estimator. Thus, we may take $\Delta > n$ to meet the bias order requirement.

Remark on scalable subsampling estimation of DNN

- The two assumptions regarding the bias order of the DNN estimator are feasible:
  (1) As revealed in Yarotsky and Zhevnerchuk (2020), the approximation ability in the uniform sup-norm of a DNN can be as fast as $\Delta^{-2\xi/d}$; Δ is the total number of parameters in a DNN. Although this rate is not instructive in practice, the desired bias order is achievable.
  (2) As revealed in Belkin et al. (2019), the double-descent of the risk exists for over-parameterized estimator. Thus, we may take $\Delta > n$ to meet the bias order requirement.
- The saving of computational cost from applying scalable subagging will be more significant for executing estimation with a large DNN or with a large sample. Assume that a DNN with size $W = \Theta(n^{\phi})$. The total number of operations to train a DNN is $O(n \cdot W \cdot E)$; here E represents the number of epochs. When the size of the DNN is larger than the sample size, $O(n \cdot W \cdot E) \approx O(n^{\varphi})$; $\varphi > 2$. For our estimator, the number of operations is $O(n^{\beta\varphi} q) = O(n^{1+\beta(\varphi-1)})$. The ratio of $n^{1+\beta(\varphi-1)}$ over n^{φ} is $n^{-(\varphi-1)(1-\beta)}$.