Double Machine Learning with Gradient Boosting and Its Application to the Big N Audit Quality Effect

JUI-CHUNG (RAY) YANG (National Tsing Hua University)
Hui-Ching Chuang (Yuan Ze University)
Chung-Ming Kuan (National Taiwan University)

Workshop on Asset Pricing and Risk Management
National University of Singapore
August 29, 2019

Outline

1. Introduction
2. Double Machine Learning
3. Monte Carlo Experiments
4. Empirical Results
5. Conclusion

Double / Debiased Machine Learning for Treatment and Structural Parameters

- Various machine learning algorithms have been used in estimating the treatment effects.

  CART Su et al. (2009), Athey and Imbens (2016).
  RF Lu et al. (2018), Wager and Athey (2018)
  SVM Imai and Ratkovic (2013)

- Chernozhukov, Chetverikov, Demirer, Duflo, Hansen, Newey, and Robins (CCDDHNR, 2018, *Econometrics Journal*)

  Object inference on a low-dimensional parameter θ_0 in the presence of high-dimensional nuisance parameters η_0.
  - η_0 estimated by *some* ML algorithms.

  Method Double / debiased *machine learning* (**DML**):
  - *Neyman-orthogonal scores* + cross-fitting.

Big N Audit Quality Effect

- Big N effect: Big N audits provide better audit quality than non-Big N auditors.
- Lawrence et al. (2011) used propensity score matching (PSM), and claimed that Big N effect is mainly due to clients' characteristics.
- DeFond et al. (2016) argued result of Lawrence et al. (2011) was only one particular PSM design, and examined 3,000 combinations.
  - With/without replacement, ratios of control firms to treated firm, caliper widths, inclusion the non-linear covariates terms, etc.
- DeFond et al. (2016) concluded that Big N effect existed among majority of their designs.

Our Contributions

- By simulations, we demonstrate that the gradient boosting method is to be preferred to other learning methods, such as regression trees and random forests, in estimating the high-dimensional nuisance parameters η_0.
- We find that Big N auditors have a positive effect on audit quality and that this effect is not only statistically significant but also economically important.
- In contrast to the results of propensity score matching, our estimates of said effect are quite robust to the hyper-parameters in the gradient boosting algorithm.

Partially Linear Regression (PLR)

Robinson (1988, *Econometrica*)

$$
\begin{aligned}
Y &= \theta_0 D + f_0(\mathbf{X}) + U, \quad \mathbb{E}(U|\mathbf{X}, D) = 0. \\
D &= m_0(\mathbf{X}) + V, \quad \mathbb{E}(V|X) = 0.
\end{aligned}
$$

- Treatment effect: θ_0;
- Nuisance parameters: $\eta_0 = (f_0, m_0)$;
- $(Y, D, \mathbf{X}')'$: iid observations;
- D: treatment variable;
- $\mathbf{X} = (X_1, \ldots, X_p)'$: other controls.

A Naive Approach

$$\begin{aligned} Y &= \theta_0 D + f_0(\mathbf{X}) + U, \quad \mathbb{E}(U|\mathbf{X}, D) = 0. \\ D &= m_0(\mathbf{X}) + V, \quad \mathbb{E}(V|X) = 0. \end{aligned}$$

- Consider $\widehat{f}_N$, an ML estimator for f_0.
  - In practice, f_0 is extremely hard to estimate without any prior knowledge about θ_0.
  - Let's assume we can do it (for now).
- One naive approach to estimate θ_0:

$$\widehat{\theta}_N = \left(\frac{1}{N} \sum_{i=1}^{N} D_i D_i' \right)^{-1} \frac{1}{n} \sum_{i \in I} D_i \left(Y_i - \widehat{f}_N(\mathbf{X}_i) \right).$$

Regularization Bias

- However, in general, $\left|\sqrt{N}\left(\widehat{\theta}_N - \theta_0\right)\right| \xrightarrow{p} \infty$.

$$\sqrt{N}\left(\widehat{\theta}_N - \theta_0\right) \approx \underbrace{\left[\mathbb{E}\left(D_i D_i'\right)\right]^{-1} \frac{1}{\sqrt{N}} \sum_{i=1}^{N} D_i U_i}_{a}$$

$$\underbrace{+\left[\mathbb{E}\left(D_i D_i'\right)\right]^{-1} \frac{1}{\sqrt{N}} \sum_{i=1}^{N} m_0\left(\mathbf{X}_i\right)\left(f_0\left(\mathbf{X}_i\right) - \widehat{f}_N\left(\mathbf{X}_i\right)\right)}_{b} + r.m.$$

- Under mild conditions, $a \overset{A}{\sim} \mathcal{N}\left(0, \overline{\Sigma}\right)$ for some $\overline{\Sigma}$.
- However, in general, b, the *regularization bias*, $\xrightarrow{p} 0$ only if $\widehat{f}_N - f_0 = o_p\left(N^{-1/2}\right)$.

Bias-Variance Tradeoff and Convergence Rates

- In high-dimensional (or otherwise highly complex) settings, *regularized* estimators are need for informative learning to be feasible.
  - *E.g.*, lasso, ridge, boosting or penalized neural nets.
- The regularization in these estimators keeps the *variance* of the estimator from exploding, but also induces substantive *biases* in the estimator.
  - The rate of convergence of $\widehat{f}_N(\mathbf{x})$ to $f_0(\mathbf{x})$ is typically $N^{-\varphi_f}$ with $\varphi_f < 1/2$.
- Hence, b is of stochastic order $n^{1/2-\varphi_g} \to \infty$, as $m_0(\mathbf{X}_i) \neq 0$.

Orthogonalization and Sample Splitting

- $\widehat{\theta}_N$ is derived from the following moment condition:

$$\mathbb{E}\left[D\left(Y - f_0\left(\mathbf{X}\right) - D'\theta_0\right)\right] = \mathbb{E}\left[DU\right] = 0.$$

- Instead, CCDDHNR suggest to consider the **Neyman orthogonal** moment, which partial out the effect of $\mathbf{X}$ from D:

$$\begin{aligned}&\mathbb{E}\left[\left(D - m_0\left(\mathbf{X}\right)\right)\left(Y - f_0\left(\mathbf{X}\right) - D'\theta_0\right)\right] \\ &= \mathbb{E}\left[\left(D - m_0\left(\mathbf{X}\right)\right)\left(Y - l_0\left(\mathbf{X}\right) - V'\theta_0\right)\right] = \mathbb{E}\left[VU\right] = 0,\end{aligned}$$

  - $l_0\left(\mathbf{X}\right) = \mathbb{E}\left(Y|\mathbf{X}\right) = f_0\left(\mathbf{X}\right) + m_0\left(\mathbf{X}\right)'\theta_0.$

- CCDDHNR also suggest to split the sample into two:
  - The main part of size $n = N/2$, indexed by $i \in I$.
  - The auxiliary part of size $n = N/2$, indexed by $i \in I^c$.

Double Machine Learning

- Let $\widehat{l}_n$ and $\widehat{m}_n$ be ML estimators using I^c, and $\widehat{V} = D - \widehat{m}_n(\mathbf{X})$,

$$\tilde{\theta}_n = \left(\frac{1}{n} \sum_{i \in I} \widehat{V}_i \widehat{V}_i' \right)^{-1} \frac{1}{n} \sum_{i \in I} \widehat{V}_i \left(Y_i - \widehat{l}_n(\mathbf{X}_i) \right).$$

  Then $\sqrt{n}\left(\tilde{\theta}_n - \theta_0\right) \approx a^* + b^* + c^*$.

  - *"Double" machine learning*: ML for both f_0 and m_0.

- Under mild conditions,

$$a^* = \left[\mathbb{E}\left(V_i V_i' \right) \right]^{-1} \frac{1}{\sqrt{n}} \sum_{i \in I} V_i U_i \overset{A}{\sim} \mathcal{N}(0, \Sigma) \text{ for some } \Sigma.$$

Orthogonalization and Regularization Bias

- Say, the rates of convergence of $\widehat{m}_n(\mathbf{x})$ to $m_0(\mathbf{x})$ and $\widehat{l}_n(\mathbf{x})$ to $l_0(\mathbf{x})$ are respectively $n^{-\varphi_m}$ and $n^{-\varphi_l}$, then

$$b^* = \left[\mathbb{E}\left(V_i V_i'\right)\right]^{-1} \frac{1}{\sqrt{n}} \sum_{i \in I} \left(\widehat{m}_n(\mathbf{X}_i) - m_0(\mathbf{X}_i)\right)\left(\widehat{l}_n(\mathbf{X}_i) - l_0(\mathbf{X}_i)\right)$$

  is upper-bounded by $n^{1/2-(\varphi_m+\varphi_l)}$.

  - When $\varphi_m + \varphi_l > 1/2$, b^* vanishes.
  - When $\varphi_m = \varphi_l$, $\varphi_m + \varphi_f > 1/2 \Rightarrow \varphi_l > 1/4$.
  - *"Debiased" machine learning*: the regularization bias is removed by *orthogonalization*.

Sample Splitting and "Overfitting" Bias

$$c^* = \left[\mathbb{E}\left(V_i V_i'\right)\right]^{-1} \frac{1}{\sqrt{n}} \sum_{i \in I} V_i \left(\widehat{l}_n\left(\mathbf{X}_i\right) - l_0\left(\mathbf{X}_i\right)\right) + r.m.$$

- When $\widehat{l}_n$ is obtained using the *same* sample as $\widetilde{\theta}_n$, V_i and $\widehat{l}_n\left(\mathbf{X}_i\right) - l_0\left(\mathbf{X}_i\right)$ are generally related.
  - "*Overfitting*": $\widehat{l}_n$ extracts some of the error variation.
  - For c^* to vanish, strong conditions (*e.g.* Donsker properties) that limit complexity of the parameter space are usually assumed.
- But $\widehat{l}_n$ is obtained using I^c and $\{V_i\}$ are the errors from I!
  - They are independent (under $i.i.d.$ assumption)!
  - So c^* vanishes, as long as $\mathbb{E}\left(V|X\right) = 0$, and

$$\frac{1}{n} \sum_{i \in I} \left(\widehat{l}_n\left(\mathbf{X}_i\right) - l_0\left(\mathbf{X}_i\right)\right)^2 \xrightarrow{p} 0.$$

Cross-Fitting

- Drawback of sample splitting: estimation of θ_0 only use the main sample I.
  - A substantial loss of efficiency.
- "*Cross-Fitting*": flip the role of the main and auxiliary samples to obtain a second version of the estimator.
- By averaging the two resulting estimators, we can improve the efficiency.

Double/Debiased Machine Learning Estimation

- DML approach (Chernozhukov et al., 2018) has following steps:
  1. Randomly split observations into two disjoint subsets: $\mathcal{I}$ and $\mathcal{I}^c$.
  2. Apply a ML algorithm on $\mathcal{I}^c$ to estimate $\ell_0(\mathbf{X}) := \mathbb{E}(Y|\mathbf{X})$ and $m_0(\mathbf{X}) := \mathbb{E}(D|\mathbf{X})$ functions.
  3. Define $\tilde{\theta}_n$ as:

$$\tilde{\theta}_n = \left(\sum_{i\in\mathcal{I}} \tilde{V}_i^2\right)^{-1} \sum_{i\in\mathcal{I}} \tilde{V}_i[Y_i - \tilde{\ell}_n(\mathbf{X}_i)], \quad i \in \mathcal{I},$$

     where $\tilde{V}_i = D_i - \tilde{m}_n(\mathbf{X}_i)$. $n = N/2$, and N is the total observations.
  4. Switch $\mathcal{I}$ and $\mathcal{I}^c$ in step (2) and (3) to obtain $\tilde{\theta}_n^c$.
  5. Estimate θ_0 as

$$\tilde{\theta}_n^* = (\tilde{\theta}_n + \tilde{\theta}_n^c)/2.$$

Double/Debiased Machine Learning Estimation

- The asymptotic variance of $n^{1/2}(\tilde{\theta}_n^* - \theta_0)$ is estimated by three steps:
  1. Once $\tilde{\theta}_n$ is obtained, the asymptotic variance of $n^{1/2}(\tilde{\theta}_n - \theta_0)$ is estimated by the standard White-type estimator:

$$\tilde{\sigma}_n^2 = \left(\frac{1}{n}\sum_{i\in\mathcal{I}} \tilde{V}_i^2\right)^{-1} \left(\frac{1}{n}\sum_{i\in\mathcal{I}} \tilde{V}_i^2 \tilde{U}_i^2\right) \left(\frac{1}{n}\sum_{i\in\mathcal{I}} \tilde{V}_i^2\right)^{-1},$$

     with $\tilde{U}_i = Y_i - \tilde{\ell}_n(\mathbf{X}_i) - \tilde{V}_i\tilde{\theta}_n$, and $i \in \mathcal{I}$.
  2. Switch $\mathcal{I}$ and $\mathcal{I}^c$ in step (1) to obtain $(\tilde{\sigma}_n^c)^2$.
  3. We can estimate asymptotic variance of $n^{1/2}(\tilde{\theta}_n^* - \theta_0)$ as:

$$(\tilde{\sigma}_n^*)^2 = \frac{1}{2}\left[\tilde{\sigma}_n^2 + (\tilde{\sigma}_n^c)^2 + \left(\tilde{\theta}_n - \tilde{\theta}_n^*\right)^2 + \left(\tilde{\theta}_n^c - \tilde{\theta}_n^*\right)^2\right].$$

- Under some conditions, $n^{1/2}(\tilde{\theta}_n^* - \theta_0)$ is asymptotically normal distributed.

Double/Debiased Machine Learning Estimation

- Chernozhukov et al. (2018) show that to ensure the normality, it is needed to find a ML with the following convergence rates:

$$1/4 < \varphi_\ell, \varphi_m$$

where

$$\begin{aligned} \tilde{\ell}_n(\mathbf{X}_i) - \ell_0(\mathbf{X}_i) &= O_p\left(n^{-\varphi_\ell}\right), \\ \tilde{m}_n(\mathbf{X}_i) - m_0(\mathbf{X}_i) &= O_p\left(n^{-\varphi_m}\right), \end{aligned}$$

- The question is: Which ML algorithm satisfies this condition?
  - Limited literatures give us guidance on this aspect.
- Simulation studies are conducted in this paper.

Monte Carlo Experiments

- Y_i: outcome variable. D_i: policy/treatment variable.

$$\begin{aligned} Y_i &= \theta_0 D_i + f_0(\mathbf{X}_i) + U_i, \\ D_i &= m_0(\mathbf{X}_i) + V_i. \end{aligned}$$

- $\theta_0 = 1$.
- $U_i \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$. $\mathbf{X}_i = (X_{1i}, \ldots, X_{pi})'$ with $X_{ji} \overset{i.i.d.}{\sim} U(0,1)$.
- $V_i = 1 - m_0(\mathbf{X}_i)$ with probability $m_0(\mathbf{X}_i)$. $V_i = -m_0(\mathbf{X}_i)$ with probability $1 - m_0(\mathbf{X}_i)$.

$$f_0(\mathbf{X}_i) = \mu_f + \sum_{j=1}^{p} f_{0,j}(X_{ji}), \quad m_0(\mathbf{X}_i) = \mu_m + \sum_{j=1}^{p} m_{0,j}(X_{ji}),$$

with $\mu_f = 1$, and $\mu_m = 0.5$.

- $n = 2000$, 4000, or 8000. $p = 5$, 10, or 20.
- Number of replications $R = 2000$.

Simulation Study I: Linear Setting

	Bias×100			RMSE×100			Size (%, $\alpha = 5\%$)		
$N =$	2000	4000	8000	2000	4000	8000	2000	4000	8000
Regression Tree									
$p = 5$	64.04	96.24	114.04	68.86	98.10	114.98	79.30	99.95	100.00
$p = 10$	61.56	96.50	114.45	66.42	98.33	115.39	77.05	99.95	100.00
Random Forest									
$p = 5$	17.85	12.36	8.96	19.40	13.34	9.53	46.80	57.40	70.05
$p = 10$	38.33	27.02	19.54	39.30	27.53	19.85	97.80	99.45	100.00
GB with $d = 1$									
$p = 5$	2.18	1.26	0.61	5.62	3.85	2.60	8.15	6.50	6.65
$p = 10$	3.18	1.54	0.98	6.34	3.95	2.72	9.40	7.70	7.35
GB with $d = 2$									
$p = 5$	3.44	1.96	1.01	6.57	4.31	2.81	10.10	8.10	8.30
$p = 10$	4.91	2.48	1.51	7.71	4.60	3.06	16.45	10.45	9.35
LASSO									
$p = 5$	0.10	0.13	-0.03	4.60	3.30	2.35	5.15	5.25	4.75
$p = 10$	0.29	-0.03	0.11	4.78	3.24	2.33	5.80	4.10	5.10
SVRM									
$p = 5$	-9.80	-9.27	-9.18	10.78	9.79	9.44	60.10	84.15	98.80
$p = 10$	-9.85	-9.61	-9.17	10.94	10.12	9.44	59.80	86.10	98.50
Neural Net									
$p = 5$	31.68	21.79	7.74	33.46	38.54	13.72	0.05	2.70	0.30
$p = 10$	36.97	25.31	8.79	39.17	40.49	15.87	0.55	3.35	0.85
Nadaraya-Watson									
$p = 5$	12.93	9.64	7.14	14.97	10.69	7.75	37.90	47.25	59.65
$p = 10$	46.28	37.41	29.98	48.28	38.41	30.52	93.90	99.55	99.95

Simulation Study II: Smooth nonlinear setting

	Bias×100			RMSE×100			Size (%, $\alpha = 5\%$)		
$N =$	2000	4000	8000	2000	4000	8000	2000	4000	8000
Regression Tree									
$p = 5$	-76.32	-109.37	-128.00	81.00	111.30	128.91	84.35	100.00	100.00
$p = 10$	-75.35	-109.56	-127.56	80.21	111.35	128.52	83.15	100.00	100.00
Random Forest									
$p = 5$	-18.97	-12.55	-8.50	20.66	13.57	9.17	41.05	47.75	54.90
$p = 10$	-44.37	-31.69	-23.12	45.41	32.23	23.41	97.95	99.80	100.00
GB with $d = 1$									
$p = 5$	-1.48	-0.72	-0.43	5.64	3.78	2.62	6.75	6.50	5.45
$p = 10$	-3.11	-1.24	-0.65	6.55	3.98	2.64	10.40	7.35	6.50
GB with $d = 2$									
$p = 5$	-2.26	-1.04	-0.55	6.29	4.08	2.71	8.05	6.35	5.70
$p = 10$	-4.72	-1.99	-1.05	7.97	4.51	2.93	13.90	8.25	7.70
LASSO									
$p = 5$	5.54	5.51	5.22	8.89	7.43	6.31	11.45	19.35	31.55
$p = 10$	4.74	5.28	5.26	8.66	7.30	6.35	10.60	18.30	31.70
SVRM									
$p = 5$	-1.04	-0.94	-1.33	7.20	5.18	3.85	6.65	6.85	7.95
$p = 10$	-1.60	-1.11	-1.21	7.68	5.26	3.78	8.15	7.55	8.10
Neural Net									
$p = 5$	-34.60	-26.94	-6.78	37.13	46.90	16.29	0.00	3.35	0.45
$p = 10$	-41.79	-29.74	-6.98	44.30	50.38	19.96	0.55	3.95	1.30
Nadaraya-Watson									
$p = 5$	-14.14	-10.93	-7.99	16.43	12.11	8.67	34.00	47.35	56.90
$p = 10$	-51.11	-41.13	-33.15	53.72	42.42	33.77	90.65	98.20	99.95

Simulation Study III: Nonsmooth Nonlinear Setting

	Bias×100			RMSE×100			Size (%, $\alpha = 5\%$)		
$N =$	2000	4000	8000	2000	4000	8000	2000	4000	8000
Regression Tree									
$p = 5$	-88.98	-127.64	-151.04	94.71	130.02	152.35	85.20	100.00	100.00
$p = 10$	-87.60	-127.47	-150.57	93.26	129.82	151.82	86.25	100.00	100.00
Random Forest									
$p = 5$	-18.75	-10.98	-7.48	22.34	13.20	8.77	25.65	22.85	28.85
$p = 10$	-48.65	-32.61	-21.53	50.50	33.70	22.13	92.00	95.95	98.05
GB with $d = 1$									
$p = 5$	-0.20	-0.30	-0.13	6.39	4.05	2.69	5.95	5.45	5.35
$p = 10$	-0.83	-0.34	-0.27	6.96	4.22	2.76	6.75	5.25	5.80
GB with $d = 2$									
$p = 5$	-1.15	-0.56	-0.27	6.79	4.16	2.73	6.55	6.00	5.25
$p = 10$	-2.12	-0.94	-0.51	7.57	4.54	2.87	8.45	7.20	6.75
LASSO									
$p = 5$	-0.19	0.12	0.09	12.53	8.74	6.31	5.35	4.85	5.25
$p = 10$	-0.78	-0.35	-0.15	12.85	8.76	6.55	5.40	4.50	5.35
SVRM									
$p = 5$	-7.25	-6.85	-6.73	14.10	10.82	9.01	8.65	11.55	20.15
$p = 10$	-7.35	-7.03	-6.86	14.47	10.96	9.25	9.80	12.45	21.50
Neural Net									
$p = 5$	-41.42	-33.54	-11.68	45.85	65.04	20.84	0.65	4.25	0.30
$p = 10$	-50.79	-38.82	-13.69	55.90	70.52	29.96	3.60	5.25	1.70
Nadaraya-Watson									
$p = 5$	-10.95	-6.16	-2.67	18.01	11.57	6.99	14.10	12.65	8.00
$p = 10$	-62.80	-48.25	-36.80	66.44	50.34	37.92	89.35	95.25	98.90

Simulation Study IV: Nonsmooth Nonlinear Setting with 2-Way Interactions

	Bias×100			RMSE×100			Size (%, $\alpha = 5\%$)		
$N =$	2000	4000	8000	2000	4000	8000	2000	4000	8000
Regression Tree									
$p = 5$	-69.21	-92.90	-109.74	75.79	95.59	111.08	63.70	98.85	100.00
$p = 10$	-69.26	-93.27	-108.93	75.85	96.10	110.28	64.50	98.65	100.00
Random Forest									
$p = 5$	-13.15	-8.13	-4.65	18.42	11.01	6.34	13.15	13.50	14.20
$p = 10$	-29.69	-19.78	-12.07	32.93	21.27	12.95	43.75	60.30	66.00
GB with $d = 1$									
$p = 5$	112.73	112.67	113.06	114.84	113.61	113.53	99.95	100.00	100.00
$p = 10$	108.96	110.49	112.45	111.07	111.41	112.90	99.95	100.00	100.00
GB with $d = 2$									
$p = 5$	2.77	1.47	0.94	9.35	5.53	3.41	6.55	6.60	5.45
$p = 10$	6.73	3.50	2.29	11.47	6.42	4.16	10.35	8.80	10.45
LASSO									
$p = 5$	211.64	210.97	211.17	213.17	211.71	211.54	100.00	100.00	100.00
$p = 10$	208.75	210.03	211.37	210.35	210.76	211.74	100.00	100.00	100.00
SVRM									
$p = 5$	199.29	199.17	199.66	200.86	199.92	200.03	100.00	100.00	100.00
$p = 10$	196.59	198.38	199.83	198.28	199.12	200.21	100.00	100.00	100.00
Neural Net									
$p = 5$	39.34	22.77	12.37	45.68	28.36	19.40	0.50	0.25	0.80
$p = 10$	46.24	33.21	22.95	54.88	38.71	27.93	2.80	2.45	3.80
Nadaraya-Watson									
$p = 5$	2.21	2.31	3.03	18.68	12.91	8.73	5.70	5.45	5.95
$p = 10$	-25.11	-14.83	-8.72	38.60	23.73	15.14	21.45	18.20	15.55

Why Random Forests Don't Work?

- Need a rate of convergence faster than $1/4$.
- Rates of convergence of trees and random forests are not clear.
  - The same training data is used for both *model selection* and *model fitting*.
- Intuition: Lin and Jeon (2006) showed that random forests can be viewed as the *adaptive* version of nearest neighbors.

Rate of Convergence of Random Forests

Biau (2012, *JMLR*)

- Biau (2012, *JMLR*) studied the simplied version of random forests (Breiman, 2004), and showed that

$$\mathbb{E}\left(\left[\bar{r}_n\left(\mathbf{X}\right)-r\left(\mathbf{X}\right)\right]^2\right)=O\left(n^{\frac{-0.75}{S\log 2+0.75}}\right),$$

  where $r\left(\mathbf{X}\right)=\mathbb{E}\left(Y|\mathbf{X}\right)$ is the unknown function, $\bar{r}_n$ is the simplified random forest, and S is # of Strong features, *i.e.*, Y only depends on these S features.

  - In Biau (2012), # of features is finite.

- The rate of convergence is $n^{-0.375/\left(S\log 2+0.75\right)}$.

  - When $S=1$, $0.375/\left(S\log 2+0.75\right)=0.2598>1/4$.
  - When $S=2$, $0.375/\left(S\log 2+0.75\right)=0.1755<1/4$....

Algorithm Gradient Boosting (Efron and Hastie, 2016)

Randomly split the sample into $\mathcal{I}$ and $\mathcal{I}^c$. Set the number of trees to be grown as B, the number of splits (interaction depth) d, and the shrinkage parameter ϵ.

1. *Initialize the fitted model as $\widehat{G}_0 = 0$ and set the initial residuals $r_{0,i} = Y_i$, $i = 1, \ldots, n$, where these Y_i are from $\mathcal{I}^c$.*
2. *For $b = 1, 2, \ldots, B$,*
    1. *Grow a regression tree $\tilde{g}_b$ with depth d by minimizing the RSS: $\sum_{i=1}^{n} [r_{b-1,i} - g_b(\mathbf{X}_i)]^2$, based on the sub-sample $\mathcal{I}^c$.*
    2. *Compute a shrunken version of $\tilde{g}_b$: $\hat{g}_b = \epsilon \cdot \tilde{g}_b$ and update the fitted model for Y_i as*

$$\widehat{G}_b(\mathbf{X}_i) = \widehat{G}_{b-1}(\mathbf{X}_i) + \hat{g}_b(\mathbf{X}_i), \quad i = 1, 2, \ldots, n.$$

    3. *Update the residuals as*

$$r_{b,i} = r_{b-1,i} - \hat{g}_b(\mathbf{X}_i), \quad i = 1, 2, \ldots, n.$$

3. *Return the final, fitted function $\widehat{G}_B(\mathbf{X}_i)$, $i = 1, 2, \ldots, n$.*

Why Gradient Boosting Works?

- The convergence rate of gradient boosting is not clear.
- Efron et al. (2004) point out the similarity between gradient boosting and LASSO.
- The L_2 boosting (not the same as gradient boosting) has the convergence rate slower than but close to that of LASSO (Luo and Spindler, 2016).
- Gradient boosting with interaction depth d works best if $\mathbb{E}(Y|X)$ is with up to d-way interactions of covariates.

Empirical setting

$$\begin{aligned} \text{AbsPDAC}_{it} &= \theta_0 \, \text{BigN}_{it} + f\left(\mathbf{X}_{it}\right) + U_{it}, \\ \text{BigN}_{it} &= m\left(\mathbf{X}_{it}\right) + V_{it}. \end{aligned}$$

- AbsPDAC: absolute value of the performance-matched discretionary accruals. Proxy for inferior audit quality.
- BigN: dummy of the client of a Big N auditor.
- $\mathbf{X}$: logarithm of total assets, asset turnover ratios (ATURN, sales divided by one-year lagged total assets), market values of equity (MKT), return on assets ratios (ROA), leverages (LEV, debts scaled by the one-year lagged total assets), current assets (CURR, scaled by current liabilities), and year and industry categorical variables (two-digit SIC codes).

Empirical Results

- We replicate the data used in Lawrence et al. (2011). There are 69,354 Big-N audited firms and 17,877 non-Big-N audited firms during 1988 to 2006 in U.S.
- DML with gradient boosting results support the existence of the Big N effect. The Big N auditors are negatively significantly associated to lower AbsPDAC at 1% level.
- Also find the non-linear relations between firm characteristics and AbsPDAC using Friedman (2001)'s partial dependence as, for a specific value x_1^* of the covariate X_1 is computed as

$$\frac{1}{n}\sum_{i=1}^{n}\widehat{G}_B(x_1^*, x_{i2}, \ldots, x_{ip}),$$

where $\widehat{G}_B$ is obtained from Step 3 of the gradient boosting algorithm.

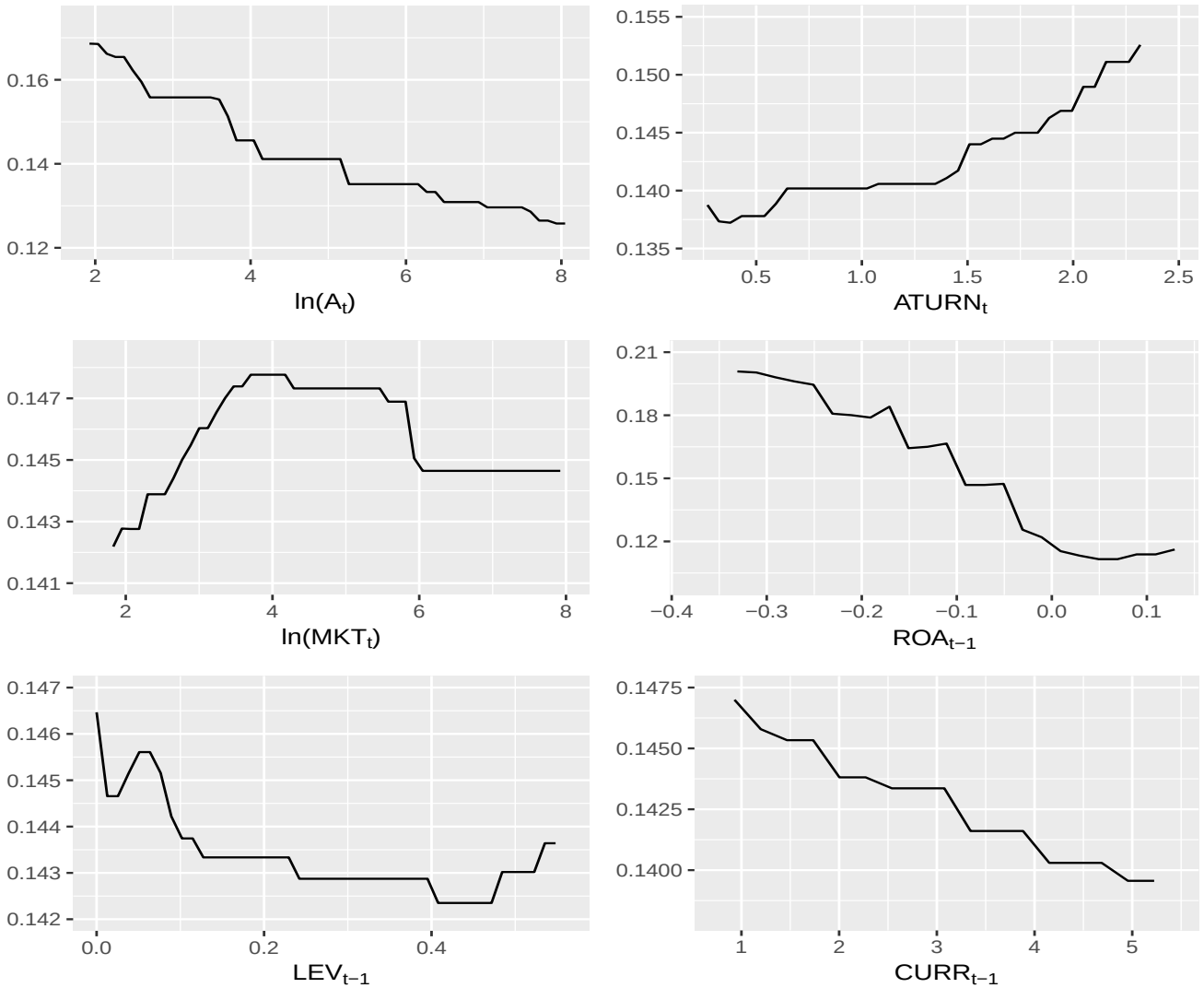
Big N Effect Revisit: PSM

	Dependent Variable: AbsPDAC	
	(1) Without replacement	(2) With replacement
BigN	-0.0010	-0.0092***
s.e.	(0.0017)	(0.0015)
Year	Yes	Yes
Industry	Yes	Yes

The estimation results based on DML-GB

	Dependent variable: AbsPDAC			
Year	No	Yes	No	Yes
Industry	No	No	Yes	Yes
Panel A: $d = 1$				
BigN	-0.0101^{***}	-0.0099^{***}	-0.0098^{***}	-0.0097^{***}
s.e.	(0.0014)	(0.0014)	(0.0014)	(0.0014)
t ratio	-7.29	-6.96	-7.13	-6.85
Panel B: $d = 2$				
BigN	-0.0078^{***}	-0.0063^{***}	-0.0072^{***}	-0.0070^{***}
s.e.	(0.0014)	(0.0015)	(0.0014)	(0.0015)
t ratio	-5.61	-3.30	-5.19	-4.77

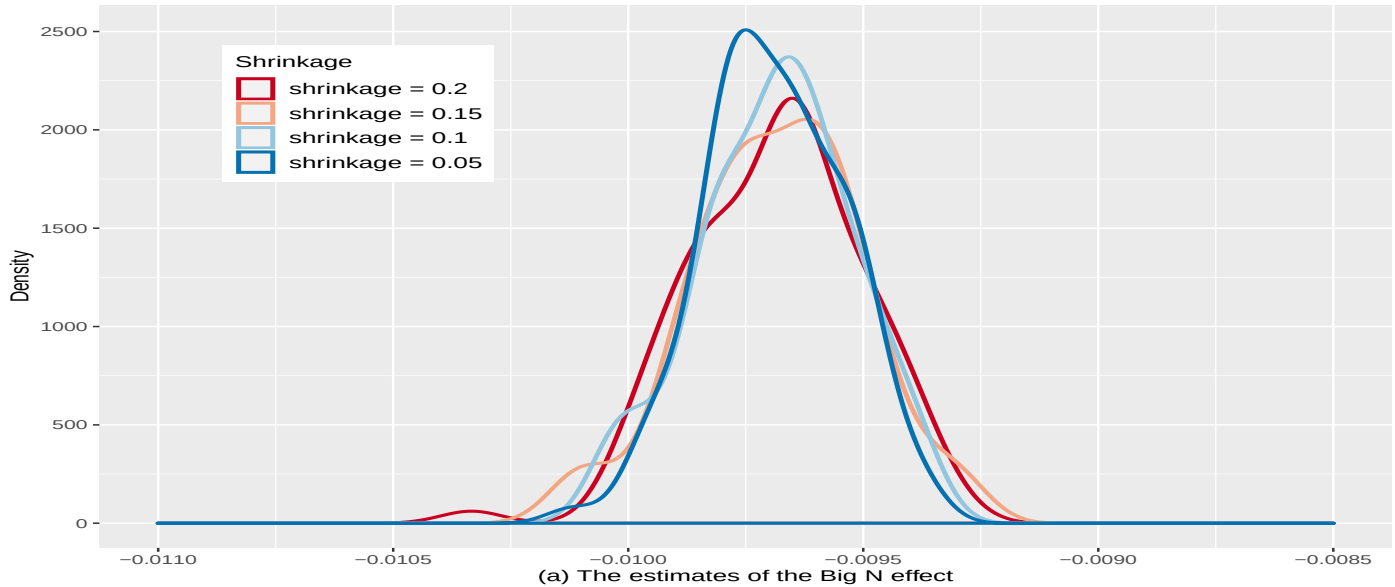
Partial dependence on client firm characteristics



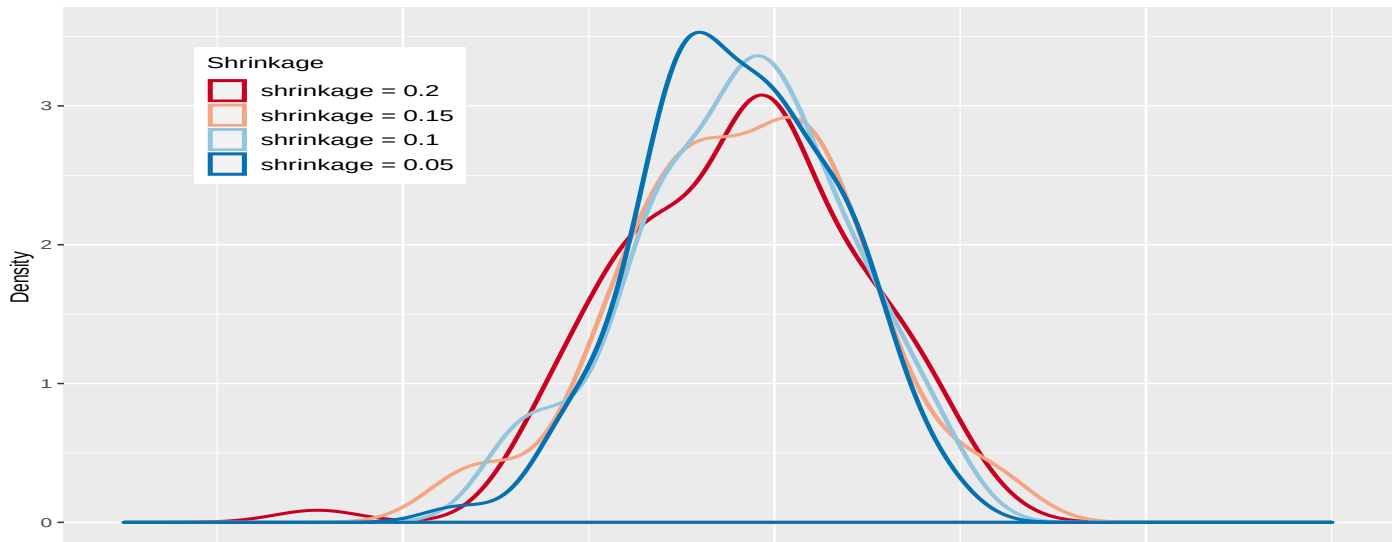
Sensitivity of Big N effect to GB hyper-parameters

- We set the following GB hyper-parameters:
  - 100 sets of randomly generated subs-samples for each shrinkage parameter.
  - Four shrinkage parameters, ϵ: 0.05, 0.1, 0.15, and 0.2.
  - The bag fraction is 0.5, and the number of trees is determined by 10-fold cross-validation.
- Results:
  - These densities of estimates are quite similar and concentrated between -0.0090 and -0.0105
  - DML-GB estimates are quite robust to the shrinkage parameters and sample splitting schemes.
  - The corresponding densities of the t ratios from DML-GB estimates are similar and all below the 1% critical value, -2.33.
  - This result is in contrast to the density plot of DeFond et al. (2016), who show that the PSM estimates are more sensitive to design settings.

Figure: Density of the Big N effect: The DML-GB with $d = 1$



(a) The estimates of the Big N effect



(b) The t ratio of the Big N effect

Conclusion

- We find that Big N auditors have a positive effect on audit quality and that this effect is not only statistically significant but also economically important.
- We demonstrate by simulations that, for the DML approach, the gradient boosting method is to be preferred to other learning methods, such as regression tree and random forest.
- In contrast to the results of propensity score matching, our estimates of said effect are quite robust to the hyper-parameters in the gradient boosting algorithm.

Key Reference

Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W., Robins, J., 2018. Double/debiased machine learning for treatment and structural parameters. Econometrics Journal 21, 1–68.

DeFond, M., Erkens, D. H., Zhang, J., 2016. Do client characteristics really drive the Big N audit quality effect? New evidence from propensity score matching. Management Science 63 (11), 3628–3649.

Efron, B., Hastie, T., 2016. Computer Age Statistical Inference: Algorithms, Evidence and Data Science. Cambridge Press.

Friedman, J. H., 2001. Greedy function approximation: A gradient boosting machine. Annals of Statistics 29, 1189–1232.

Lawrence, A., Minutti-Meza, M., Zhang, P., 2011. Can Big 4 versus non-Big 4 differences in audit-quality proxies be attributed to client characteristics? The Accounting Review 86 (1), 259–286.