Estimation and Inference of Heterogenous Treatment Effects via Boosting

October 27, 2025

Abstract

Gradient boosting is widely popular due to its flexibility and predictive accuracy. However, statistical inference and uncertainty quantification for gradient boosting remain challenging and under-explored. We propose a unified framework for statistical inference in gradient boosting regression. Our framework integrates dropout or parallel training with a recently proposed regularization procedure that allows for a central limit theorem (CLT) for boosting. With these enhancements, we surprisingly find that *increasing* the dropout rate and the number of trees grown in parallel at each iteration substantially enhances signal recovery and overall performance. Our resulting algorithms enjoy similar CLTs, which we use to construct built-in confidence intervals, prediction intervals, and rigorous hypothesis tests for assessing variable importance. Numerical experiments demonstrate that our algorithms perform well, interpolate between regularized boosting and random forests, and confirm the validity of their built-in statistical inference procedures.

1 Introduction

Consider the common setup where we have i.i.d. examples $i = 1, ..., n$, such that

$$Y_i = \tau(\mathbf{x}_i)A_i + \boldsymbol{f}(\mathbf{X}_i) + \epsilon_i, \qquad \mathbb{E}[\epsilon_i | A_i, \mathbf{X}_i] = 0,$$

where $A_i \in \{0, 1\}$ is a binary treatment indicator and τ is the (heterogenous) treatment effect at $\mathbf{x}$:

$$\tau(\mathbf{x}) = \mathbb{E}\left[Y_i^{(1)} - Y_i^{(0)} \middle| \mathbf{X}_i = \mathbf{x} \right].$$

We assume unconfoundedness, that is, that the potential outcomes are independent of the treatment conditional on the covariates: $\{Y_i^{(1)}, Y_i^{(0)} \perp\!\!\!\perp A_i \mid \mathbf{X}_i\}$. Under unconfoundedness, it can be shown that

$$\tau(\mathbf{x}) = \mathbb{E}\left[Y_i \left(\frac{A_i}{\pi(\mathbf{x})} - \frac{1 - A_i}{1 - \pi(\mathbf{x})} \right) \;\middle|\; \mathbf{X}_i = \mathbf{x} \right], \text{ where } \pi(\mathbf{x}) = \mathbb{E}[A_i \mid \mathbf{X}_i = \mathbf{x}] \text{ is the propensity score.}$$

This can be written in another way:

$$Y_i - \mathbf{m}(\mathbf{x}) = \tau(\mathbf{x}) \cdot (A_i - \pi(\mathbf{x})) + \epsilon_i, \text{ where } \mathbf{m}(\mathbf{x}) = \mathbb{E}[Y_i | \mathbf{X}_i = \mathbf{x}] = \boldsymbol{f}(\mathbf{x}) + \tau(\mathbf{x}) \cdot \pi(\mathbf{x}) \text{ is the outcome model.}$$

This implies that we can regress the demeaned outcome on the demeaned treatment to recover the CATE. Now say that we have a reasonable outcome model given by kernel regression:

$$\mathbf{m}(\mathbf{x}) \approx \widehat{\boldsymbol{r}}_n^P(\mathbf{x})^\top \mathbf{Y}_n = \widehat{\mathbf{m}}(\mathbf{x}) = \widehat{\boldsymbol{k}}_n^P(\mathbf{x})^\top (\mathbf{I} + (K-1)\widehat{\mathbf{K}}_n^P)^{-1} K \mathbf{Y}_n.$$

If the weights $\widehat{\boldsymbol{r}}_n^P(\mathbf{x})$ can be shared between outcome prediction and propensity score prediction, the propensity score model can also be given by kernel regression:

$$\pi(\mathbf{x}) \approx \widehat{\pi}(\mathbf{x}) = \widehat{\boldsymbol{r}}_n^P(\mathbf{x})^\top \mathbf{A}_n.$$

Why are the propensity score and outcome model computed with the same weights? In a tree, the treatment propensity is the proportion of units in the same leaf that are treated, while the outcome is the average outcome within the same leaf. This is straightforward within an adaptive nearest neighbors interpretation. Within a kernel regression, we maybe have to do a little more work to justify this.

1.1 Estimation via an outcome model

The g-formula[1] states that

$\widehat{\text{ATE}} = \mathbb{E}[\mathbf{m}^{(a)}(\mathbf{x})] = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}(\mathbf{X})}\left[Y \mid A = a, \mathbf{X} = \mathbf{x}\right]$, where the expectation is taken over the density of the covariates.

We can estimate the CATE $\mathbf{m}^{(a)}(\mathbf{x})$ via boosting, under unconfoundedness. This is given by

$$\widehat{\mathbf{m}}^{(1)}(\mathbf{x}) - \widehat{\mathbf{m}}^{(0)}(\mathbf{x}), \text{ where } \widehat{\mathbf{m}}^{(a)}(\mathbf{x}) \text{ is the boosting prediction,}$$

with variance given by the Delta method:

$$\text{Var}(\widehat{\mathbf{m}}^{(1)}(\mathbf{x})) + \text{Var}(\widehat{\mathbf{m}}^{(0)}(\mathbf{x})) - 2\text{Cov}\left(\widehat{\mathbf{m}}^{(1)}(\mathbf{x}), \widehat{\mathbf{m}}^{(0)}(\mathbf{x})\right).$$

Both the first and second quantities can be estimated by $\widehat{\sigma}^2\|\widehat{\boldsymbol{r}}_n^{(a)}(\mathbf{x})\|_2^2$, and the cross-term is given by $2 \cdot \widehat{\sigma}^2 \cdot \widehat{\boldsymbol{r}}_n^{(1)}(\mathbf{x})^\top \widehat{\boldsymbol{r}}_n^{(0)}(\mathbf{x})$. We then have a built in variance estimate for the CATE:

$$\widehat{\text{Cov}}(\widehat{\tau}(\mathbf{x})) = \widehat{\sigma}^2\|\widehat{\boldsymbol{r}}_n^{(1)}(\mathbf{x})\|_2^2 + \widehat{\sigma}^2\|\widehat{\boldsymbol{r}}_n^{(0)}(\mathbf{x})\|_2^2 - 2 \cdot \widehat{\sigma}^2 \cdot \widehat{\boldsymbol{r}}_n^{(1)}(\mathbf{x})^\top \widehat{\boldsymbol{r}}_n^{(0)}(\mathbf{x}).$$

2 EBMs

Now say we have an additive model:

$$y_i = \tau(\mathbf{x}_i)a_i + \sum_{j=1}^{d} f(\mathbf{x}_{i,j}) + \epsilon.$$

[Kevin: Defer EBMs to separate paper]

3 Yihui comment

[Kevin: Alternate algorithm: Run boosting as usual, and then take average of boosters as weak learners in a GRF.]
Suppose we have

$$\sigma^{-1}\sum_{i=1}^{n}\alpha_i\psi_{\theta_0,\nu_0,i} \Rightarrow N(0, I),$$

and since we have

$$\sum_{i=1}^{n}\alpha_i\psi_{\hat{\theta},\hat{\nu},i} = 0,$$

[Kevin: This only holds on the training set.] we have

$$\sigma^{-1}(\sum_{i=1}^{n}\alpha_i\psi_{\theta_0,\nu_0,i} - \sum_{i=1}^{n}\alpha_i\psi_{\hat{\theta},\hat{\nu},i}) \Rightarrow N(0, I).$$

If we view θ as a function of $\sum_{i=1}^{n}\alpha_i\psi_{\theta,\nu,i}$, then we can use delta method to obtain

$$\sigma^{-1}(\widehat{\theta} - \theta_0) \Rightarrow N(0, f'^2)$$

for f' as a derivative

[Kevin: Do we need to even fit gradient trees here if we are gradient boosting?]

[1] The g stands for great.

The outcome variable ψ has parameter θ and ν within it, so I think we should only use the weights of previous trees, rather than the previous estimation of parameters, in boosting. I will provide a very rough idea about how we can construct the boosting tree. Suppose the tree weight function of unit i in the bth round of boosting is $\alpha_i^{(b)}(x)$. In the $(k+1)$th round of boosting, we randomly pick $\mathcal{G}_{k+1} \subset \{1, \cdots, k\}$ with the probability of q and pick a random subsample S_{k+1} to train the new tree. In each splitting in the $(k+1)$th tree we should maximize

$$\sum_{i \in S_{k+1}} \left(\hat{\theta}^{\text{after splitting}}(x_i) - \hat{\theta}^{\text{before splitting}}(x_i) \right)^2,$$

where the two $\hat{\theta}$ are obtained from

$$\sum_{i \in S_{k+1}} \left(\sum_{b \in \mathcal{G}_{k+1}} \alpha_i^{(b)}(x) + \alpha_i^{(k+1)}(x) \right) \psi_{\hat{\theta}, \hat{\nu}, i} = 0$$

[Kevin: This looks like a regular boosting estimating equation. Does this incorporate Boulevard regularization?] [Kevin: May not have to use Boulevard! Because we are not interested in the residuals.]

[Kevin: This is effectively taking an average of the tree weights. But because of the term on the left, which is absent in the generalized random forest setting, this is still a boosting procedure!] Then, similar as Wager 2019, we should approximately maximize

$$\sum_{j=1}^{2} |C_j| \left(\sum_{x_i \in C_j} \rho_{ij}(x_i) \right)^2$$

where

$$\rho_{ij}(x_i) = -\xi^T \left(\nabla \left(\lambda \sum_{b \in \mathcal{G}_{k+1}} \alpha_i^{(b)}(x_i) \psi_{\hat{\theta}, \hat{\nu}, i} \right) + |C_j| A_P \right)^{-1} \psi_{\hat{\theta}, \hat{\nu}, i}.$$

To prove the asymptotic property, maybe we only need to prove that $\alpha_i^{1:B}(x) := \frac{1}{B} \sum_{b=1}^{B} \alpha_i^b(x)$ converges in finite sample to a limit function $\alpha_i(x)$ as $B \to \infty$. Then, as long as $\alpha_i(x)$ is supported on a neighborhood of x_i with width of a certain order, we have

$$\sigma^{-1} \sum_{i=1}^{n} \alpha_i \psi_{\theta_0, \nu_0, i} \Rightarrow N(0, I).$$

[Kevin: Perform boosting to get the tree structures, then throw away the leaf weights. Refit, and take average of tree weights after.]

Questions:

1. How adaptive can trees be in Boulevard framework? What's the advantage of using Boulevard framework over incorporating boosting into base tree? [Kevin: Compared to taking an average of boosters, this is significantly faster and requires three orders of magnitudes less trees.]

3.1 Updates on 7/23

A big problem: the violation of structure-value isolation.

Solutions:

1. Split the data, use one part to learn tree structures by gradient algorithm with boosting. After sufficient boosting, we randomly draw the tree structures here to run standard grf on the other part. (It's a little trivial in methodology but may be useful in practice)
2. Run Boulevard for residual-based parameters.
3. Cross-fitting! Good luck everybody!

Algorithm 1 Boosting algorithm only for tree structure

Input: Tree-structure learning sample and grf sample (D_1, D_2) with features $\mathbf{X}$, treatment variables A, and labels Y. Number of boosting rounds B for tree-structure learning stage, tree depth T, dropout rate $p = 1 - q$ in tree-structure learning stage, subsample rate ξ, and regularization parameter λ

for $b = 0, \ldots, B-1$ **do**

Subsample data $G_b \subseteq D_1$ with probability ξ. We'll learn the bth gradient tree with G_b.

Subsample $S_b \subseteq \{0, \ldots, b-1\}$ with probability q. For arbitrary $b' \leq b$ and $i \in G_b$, define tree weight

$$\alpha_i^{(b')}(X) = \frac{1(X_i \in L_X)}{\sum_{i \in G_{b'}} 1(X_i \in L_X)},$$

where L_X is the leaf containing X in the (b')th tree. Let $\alpha_i^{(b')}(X) = 0$ if the denominator is 0.

Fit the plug-in parameters before we grow the bth tree. If S_b is not null, then for any $i \in G_b$, calculate

$$(\hat{\theta}(X_i), \hat{\nu}(X_i)) = argmin_{\theta,\nu} \left\{ \| \sum_{i' \in G_b} \left(\sum_{b' \in S_b} \alpha_{i'}^{(b')}(X_i) \right) \psi_{\theta,\nu}(O_{i'}; X_i, \lambda) \| \right\}.$$

Plug in this set of $(\hat{\theta}(X_i), \hat{\nu}(X_i))$ in the splitting of the whole tree. If S_b is null, then in the splitting of each node P we need to fit a new set of parameters. However, $(\hat{\theta}(X_i), \hat{\nu}(X_i)$ will be the same for all $X_i \in P$, and

$$(\hat{\theta}(X_i), \hat{\nu}(X_i)) = argmin_{\theta,\nu} \left\{ \| \sum_{i' \in G_b, X_{i'} \in P} \psi_{\theta,\nu}(O_{i'}; X_i, \lambda) \| \right\}.$$

for $t = 0, \ldots, T-1$ **do**

Based on the plug-in parameters and the weights from previous trees and the first t layers of this tree, calculate the gradient ρ_i for all $i \in G_b$. Prior to splitting any node at a given depth of the tree, assume that $\alpha_i^{(b)}(X)$ denotes the current tree weight for the new tree. For any $i \in G_b$, calculate

$$\rho_i = -\xi^T \left(\sum_{i' \in G_b} \left(\sum_{b' \in S_b} \alpha_{i'}^{(b')}(X_i) + \alpha_{i'}^{(b)}(X_i) \right) \nabla \psi_{\hat{\theta}(X_i), \hat{\nu}(X_i)}(O_{i'}; X_i, \lambda) \right)^{-1} \psi_{\hat{\theta}(X_i), \hat{\nu}(X_i)}(O_i; X_i, \lambda).$$

Here, ξ is a vector with 1 on the digits corresponding to θ and 0 on the digits corresponding to ν.

Based on ρ_i, split all nodes in the tth layer of the tree. Suppose that the parent node contains $\{i : X_i \in P\}$, to decide how we split P into C_1 and C_2, we maximize

$$\sum_{j=1}^{2} \frac{1}{|i : X_i \in C_j, i \in G_b|} \left(\sum_{X_i \in C_j, i \in G_b} \rho_i \right)^2.$$

end for

Update $\beta_i^{(b)}(x) = \frac{b-1}{b} \beta_i^{(b-1)}(x) + \frac{\lambda}{b} (\sum_{b' \in S_b} \alpha_i^{(b')}(x) + \alpha_i^{(b)}(x))$. Start from $\beta_i^{(0)}(x) = 0$.

end for

Return for all interested X

$$(\hat{\theta}(X), \hat{\nu}(X)) = argmin_{\theta,\nu} \left\{ \| \sum_{i \in D_2} \beta_i(X) \psi_{\theta,\nu}(O_i; X, \lambda) \| \right\}$$

For standard CATE estimation, we have 1-dimensional θ and 1-dimensional ν, and the formula of ψ is

$$\psi_{\theta,\nu}(O_i; X) = (Y_i - A_i\theta - \nu)(A_i, 1)^T.$$

For CATE in local random forest, we have 1-dimensional θ and $2d+1$-dimensional ν, where d is the dimensional of X. Let ν_1, ν_2 and ν_3 be the first 1 dimension, the following d dimensions, and the last d dimensions of ν,

respectively. There would be regularization terms with coefficient λ for ν_2 and ν_3, and the formula of ψ is

$$\psi_{\theta,\nu}(O_i; X, \lambda) = (Y_i - A_i(\theta + \nu_2(X_i - X)) - (\nu_1 + \nu_3(X_i - X)))(A_i, 1, A_i(X_i - X), X_i - X)^T + 2\lambda(0, 0, \nu_2, \nu_3)^T.$$

In these two cases, as well as in many other cases, we can solve the linear equations to get θ and ν for different X_i instead of doing optimization.

The final algorithm in Algorithm 1 degenerates into a random forest where the tree structures are chosen adaptively. The leaf weights are not adaptive! [Yihui: Also, Algorithm 1 is expensive in computation because we need to fit local parameters for all units before we split the tree. A main reason why we use subsampling and dropout in Algorithm 1 is to cut the computation expense.] Hence we may wish to adopt Algorithm 2, when we are looking at estimating linear functionals of the labels. [Yihui: In Algorithm 2, we can still use the tree structure learning algorithm as in Algorithm 1, but instead of running ensemble optimization we'll run Boulevard boosting and ensemble the parameters obtained in separate optimizations.]

Algorithm 2 Boosting algorithm for both tree structure and residuals

1: **Input:** Tree-structure learning sample and Boulevard learning sample (D_1, D_2) with features $\mathbf{X}$, treatment variables A, and labels Y. Number of boosting rounds (B, M) for tree-structure learning stage and Boulevard stage, dropout rate $(p_1 = 1 - q_1, p_2 = 1 - q_2)$ for tree-structure learning stage and Boulevard stage, subsample rate (ξ_1, ξ_2) in the two stage, and regularization parameter λ (optional)

2: Do everything in the tree-structure learning stage as in 1.

3: **for** m=1,...,M **do**

4: Subsample data $H_m \subseteq D_2$ with probability ξ_2.

5: Draw a tree from D_1. The probability of choosing the bth tree proportional to $(1+k_b)$, where k_b is the number of trees in D_1 boosting on the bth tree. For all $i \in D_2$, let L_X be the leaf containing X in the selected tree, and we have

$$\beta_i^{(m)}(X) = \frac{1(X_i \in L_X, i \in H_m)}{\sum_{i \in D_2} 1(X_i \in L_X, i \in H_m)}.$$

6: Subsample $T_m \subseteq \{0, \ldots, m-1\}$ with probability q_2. For $m' < m$, let

$$\hat{Y}_i^{(m')} = \sum_{i'} \beta_{i'}^{(m')}(X_i) Y_{i'}$$

and let

$$Z_i = Y_i - \frac{1}{m-1} \sum_{m' \in T_m} \hat{Y}_i^{(m')}$$

for all $i \in H_m$. Let $\tilde{O}_i$ be the vector of observable variables where Y_i is replaced with Z_i.

7: For any interested X, calculate

$$(\hat{\theta}(X)^{(m)}, \hat{\nu}(X)^{(m)}) = argmin_{\theta,\nu} \left\{ \| \sum_{i \in D_2} \beta_i^{(m)}(X) \psi_{\theta,\nu}(\tilde{O}_i; X, \lambda) \| \right\}$$

8: Let $\hat{\theta}_m(X) = \frac{1+q_2}{M} \sum_{m' \leq m} \hat{\theta}(X)^{(m')}$

9: **end for**

10: **Return for all interested** X

$$\hat{\theta}_M(X)$$

[Kevin: Problem: Trees here are not fit on residuals. It is possible to run standard (L2) Boulevard, get the propensity scores from the Boulevard kernel, and outcome model from L2 Boulevard. With extra work, perhaps we can make leaf updates according to the score equation.]

In this algorithm, it matters how we define parameters as linear functional of Y. The easiest way is to assume that the grf estimator of $\theta(X)$ has a closed-form formula

$$\hat{\theta}(X) = K_n(X)\mathbf{Y}$$

for some K_n. This holds for the two examples I mentioned before. Under this assumption, we can easily prove the anology of Theorem 1 in the Boulevard paper.

Proof. To show Algorithm 2 is a stochastic contraction mapping, define $\mathbf{u}_t := \hat{\theta}_t(X) - \hat{\theta}^*(X)$. And define $\widetilde{\mathbf{Y}}_t := \frac{1}{t-1}\sum_{k=1}^{t-1} B_k \hat{t}_k$, $B_k \overset{i.i.d.}{\sim} \text{Ber}(q_2)$. To check mean contraction, notice that

$$\begin{aligned}
\|\mathbb{E}[\mathbf{u}_t \mid \mathcal{F}_{t-1}]\| &= \left\| \mathbb{E}\left[\frac{t-1}{t}\hat{\theta}_{t-1}(X) + \frac{1}{t}K_n(\mathbf{Y} - \widetilde{\mathbf{Y}}_t) - \hat{\theta}^*(X) \,\middle|\, \mathcal{F}_{b-1} \right] \right\| \\
&= \left\| \frac{t-1}{t}(\hat{\theta}_{t-1}(X) - \hat{\theta}^*(X)) + \frac{1}{t}E[K_n](\mathbf{Y} - E[\widetilde{\mathbf{Y}}_t|\mathcal{F}_{t-1}]) - \frac{1}{t}\hat{\theta}^*(X) \right\| \\
&\leq \frac{t-1}{t}\left\| \hat{\theta}_{t-1}(X) - \hat{\theta}^*(X) \right\| + \left\| \frac{1}{t}E[K_n](\mathbf{Y} - E[\widetilde{\mathbf{Y}}_t|\mathcal{F}_{t-1}]) - \frac{1}{t}E[K_n](\mathbf{Y} - q_2\hat{\theta}^*(X)) \right\| \\
&= \frac{t-1}{t}\|\mathbf{u}_{t-1}\| + \left\| \frac{1}{t}\mathbb{E}[K_n](q_2\hat{\theta}^*(X) - E[\widetilde{\mathbf{Y}}_t|\mathcal{F}_{t-1}]) \right\| \\
&\leq \frac{t-1+q_2}{t}\|\mathbf{u}_{t-1}\|
\end{aligned}$$

Next we check for bounded deviations.

$$\begin{aligned}
\|\varepsilon_t\| &= \|\mathbf{u}_t - \mathbb{E}[\mathbf{u}_t \mid \mathcal{F}_{t-1}]\| \\
&= \left\| \frac{1}{t}(K_n - \mathbb{E}[K_n])(\mathbf{Y} - \widetilde{\mathbf{Y}}_t) + \frac{1}{t}\mathbb{E}[K_n](-E[\widetilde{\mathbf{Y}}_t|\mathcal{F}_{t-1}] \mid \mathcal{F}_{b-1}]) \right\| \\
&\leq \frac{\lambda}{b}\|\mathbb{E}[S_n] - S_n\| \cdot \|Y - \Gamma_M(\widetilde{\mathbf{y}}_{b-1}) + \mathbb{E}[\Gamma_M(\widetilde{\mathbf{y}}_{b-1}) \mid \mathcal{F}_{b-1}]\|
\end{aligned}$$

Now, by Zhou and Hooker (2022), we know $\mathbb{E}[S_n]$ has both row sum and column sum no greater than 1, hence we see that

$$\|\mathbb{E}[S_n]\| \leq \sqrt{\|\mathbb{E}[S_n]\|_1 \cdot \|\mathbb{E}[S_n]\|_\infty} \leq \sqrt{1 \cdot n} = \sqrt{n}.$$

Then, it can be seen that

$$\|\varepsilon_t\| \leq \frac{1}{t} \cdot (1 + \sqrt{n}) \cdot 2M$$

Then we can show that

$$\sum_{b=1}^{\infty} \|\varepsilon_b\|^2 = \sum_{b=1}^{\infty} \mathcal{O}\left(\frac{1}{b^2}\right) < \infty.$$

□

4 8/20 meeting

4.1 Methodology

A problem in Algorithm 2 is that we're not using leaf values in previous rounds to update the residual in a new round. This means that our method is approximately equivalent to using a large m and only calculating one $\hat{\theta}$ as the final estimate.

To deal with this problem, we can turn to a new algorithm, in which we first run a standard Boulevard, and then use the residual in each round to learn $\hat{\theta}$ and average them.

4.2 Application dataset

We can use our method on the bird-watching dataset. Each bird photo is an observation, and we know information about the bird species, the photo-taking area, and the photo taker. The interested outcome variable is a dummy variable of whether the bird in the photo belongs to a certain species, which indicates its level of endangerment. We try to estimate the heterogeneous time trend in the level of endangerment, where the treatment variable is year and the covariates include variables such as the photo-taking area, the climate variables in the area, the frequency of other birds in the area, and the photographer expertise.

5 9/17 meeting

Key point: low-dimensional analysis for high-dimension confounder (e.g. check the treatment effect correlation between location and expertise, while controlling for a lot more variables). When we marginalize the effect over other variables, we can use DML to achieve better accuracy.

5.1 A version of algorithm

(i) Run Boulevard on treatment model and outcome model respectively, and possibly use grf to get more sensible outcome and treatment estimators (ii) Run orthogonal random forest with the nuisance functions from Boulevard. (use cross-fitting and orthogonal score to ensure that the randomness mainly comes from U-statistics) For example, we can use the score

$$\phi = \{Y - q(X, W) - \theta(T - g(X, W))\}(T - g(X, W))$$

with kernel weights $\alpha_i(X)$. (iii) Do inference in this level by estimating the final kernel

5.2 Questions and remarks

(i) How do we calculate h^* for V-statistic in Zhou 2021, are we going to calculate h^*? (ii) Can Boulevard be seen as V-statistics or U-statistics? (iii) Can Boulevard handle heteroskedasticity well and used in treatment nuisance function regression? (iv) It seems that Boulevard may not be optimal both in nuisance estimation and in result integration.

5.3 Takeaways

In the score estimation stage, need new score function incorporating the treatment and outcome models as well as the final model for $\theta(x)$. We will be doing three boosting procedures. It would be nice to not have to make any assumptions on the treatment and outcome models.

6 Algorithms based on Boulevard

Algorithm 3 Boosting algorithm based on Boulevard

Input: Tree-structure learning sample, main learning sample, and nuisance learning sample (D_1, D_2, D_3) with important features $\mathbf{X}$, other features $\mathbf{W}$, treatment variables A, and labels Y. Number of boosting rounds (B, M) for tree-structure learning stage and Boulevard stage, dropout rate $(p_1 = 1 - q_1, p_2 = 1 - q_2)$ for tree-structure learning stage and Boulevard stage, subsample rate (ξ_1, ξ_2) in the two stage, and regularization parameter λ (optional)

Do everything in the tree-structure learning stage as in 1.

Use Boulevard to learn nuisance function $\hat{\nu} = (\hat{Y}(x, w), \hat{A}(x, w))$ on D_3

for m=1,...,M **do**

Subsample data $H_m \subseteq D_2$ with probability ξ_2.

Draw a tree from D_1. The probability of choosing the bth tree proportional to $(1+k_b)$, where k_b is the number of trees in D_1 boosting on the bth tree. For all $i \in D_2$, let L_X be the leaf containing X in the selected tree, and we have

$$\beta_i^{(m)}(X) = \frac{1(X_i \in L_X, i \in H_m)}{\sum_{i \in D_2} 1(X_i \in L_X, i \in H_m)}.$$

Subsample $T_m \subseteq \{0, \ldots, m-1\}$ with probability q_2. For $m' < m$, let $\hat{f}^{(m')}$ be the result of outcome Boulevard in the (m')th round and let

$$Z_i = Y_i - \frac{1}{m-1} \sum_{m' \in T_m} \hat{f}_i^{(m')}$$

for all $i \in H_m$. Let $\tilde{O}_i$ be the vector of observable variables where Y_i is replaced with Z_i.

For any interested X, calculate

$$(\hat{\theta}(X)^{(m)}, \hat{\nu}(X)^{(m)}) = argmin_{\theta,\nu} \left\{ \| \sum_{i \in D_2} \beta_i^{(m)}(X) \psi_{\theta,\nu}(\tilde{O}_i; X, \lambda, \hat{\nu}) \| \right\}.$$

Also, obtain the outcome predictor $\hat{f}^{(m)}(W, X)$ to update Boulevard.

Let $\hat{\theta}_m(X) = \frac{m-1}{m} \hat{\theta}_{m-1}(X) + \frac{\lambda}{m} \hat{\theta}(X)^{(m)}$.

Exchange D_2 with D_3 to get another θ estimator, use the mean of the two θ to get the final result.

end for

Return for all interested X

$$\hat{\theta}_M(X)$$

Remark: We need four trees in total, which are the two Boulevard trees in nuisance estimation, the outcome Boulevard tree in main estimation, and the X kernel tree in main estimation. The splitting rules of the outcome/treatment predictor trees can be the same as grf, but the X-kernel tree may have different gradient form.

References