An Introduction to Machine Learning and Its Application in Estimating Treatment Effects

Presented by Zibin Huang [1]

[1]Department of Economics, University of Rochester

April 23, 2020

Machine Learning: What and Why

- Reference
  The Elements of Statistical Learning, *by Hastie, Tibshirani and Friedman*
  Machine Learning Methods That Economists Should Know About, *by Athey and Imbens*
  Big Data: New Tricks for Econometrics, *by Varian*
- A traditional linear model

$$y = x'\beta + \epsilon \tag{1}$$

- A non-parametric model

$$y = f(x, \epsilon) \tag{2}$$

- Why not always the second one?

Machine Learning: What and Why

- Reference
  The Elements of Statistical Learning, *by Hastie, Tibshirani and Friedman*
  Machine Learning Methods That Economists Should Know About, *by Athey and Imbens*
  Big Data: New Tricks for Econometrics, *by Varian*
- A traditional linear model

$$y = x'\beta + \epsilon \tag{1}$$

- A non-parametric model

$$y = f(x, \epsilon) \tag{2}$$

- Why not always the second one?

Machine Learning: What and Why

- Reference
  The Elements of Statistical Learning, *by Hastie, Tibshirani and Friedman*
  Machine Learning Methods That Economists Should Know About, *by Athey and Imbens*
  Big Data: New Tricks for Econometrics, *by Varian*
- A traditional linear model

$$y = x'\beta + \epsilon \tag{1}$$

- A non-parametric model

$$y = f(x, \epsilon) \tag{2}$$

- Why not always the second one?

Machine Learning: What and Why

- Reference
  The Elements of Statistical Learning, *by Hastie, Tibshirani and Friedman*
  Machine Learning Methods That Economists Should Know About, *by Athey and Imbens*
  Big Data: New Tricks for Econometrics, *by Varian*
- A traditional linear model

$$y = x'\beta + \epsilon \tag{1}$$

- A non-parametric model

$$y = f(x, \epsilon) \tag{2}$$

- Why not always the second one?

Machine Learning: What and Why

- Model A

$$y = x_1'\beta + \epsilon \tag{3}$$

- Model B

$$y = x_1'\beta_1 + x_2'\beta_2 + \epsilon \tag{4}$$

- Why not always the second one?
- Always better to have a more complicated model?

Machine Learning: What and Why

- Model A

$$y = x_1'\beta + \epsilon \tag{3}$$

- Model B

$$y = x_1'\beta_1 + x_2'\beta_2 + \epsilon \tag{4}$$

- Why not always the second one?
- Always better to have a more complicated model?

Machine Learning: What and Why

- Model A

$$y = x_1'\beta + \epsilon \tag{3}$$

- Model B

$$y = x_1'\beta_1 + x_2'\beta_2 + \epsilon \tag{4}$$

- Why not always the second one?
- Always better to have a more complicated model?

Machine Learning: What and Why

- Model A

$$y = x_1'\beta + \epsilon \tag{3}$$

- Model B

$$y = x_1'\beta_1 + x_2'\beta_2 + \epsilon \tag{4}$$

- Why not always the second one?
- Always better to have a more complicated model?

Machine Learning: What and Why

- Model Selection: Bias vs. Variance
  Assume that:

$$Y = f(X) + \epsilon$$

  The prediction error can be written as:

$$E[(Y - \hat{f}(x_0))^2 | X = x_0] = \sigma_\epsilon^2 + Bias^2 + Variance$$

- Too complicated model $\Rightarrow$ Over-fitting

Machine Learning: What and Why

- Model Selection: Bias vs. Variance
  Assume that:

$$Y = f(X) + \epsilon$$

  The prediction error can be written as:

$$E[(Y - \hat{f}(x_0))^2 | X = x_0] = \sigma_\epsilon^2 + Bias^2 + Variance$$

- Too complicated model $\Rightarrow$ Over-fitting

Machine Learning: What and Why

- Consider a data generating process

$$Y = 1 + 1.5X + \epsilon$$
$$\epsilon \sim N(0, 100)$$

  It is a noisy process.
- Simulate 30 samples from this process
- Let's start to fit it with different polynomials

Machine Learning: What and Why

- Consider a data generating process

$$Y = 1 + 1.5X + \epsilon$$
$$\epsilon \sim N(0, 100)$$

  It is a noisy process.
- Simulate 30 samples from this process
- Let's start to fit it with different polynomials

Machine Learning: What and Why

- Consider a data generating process

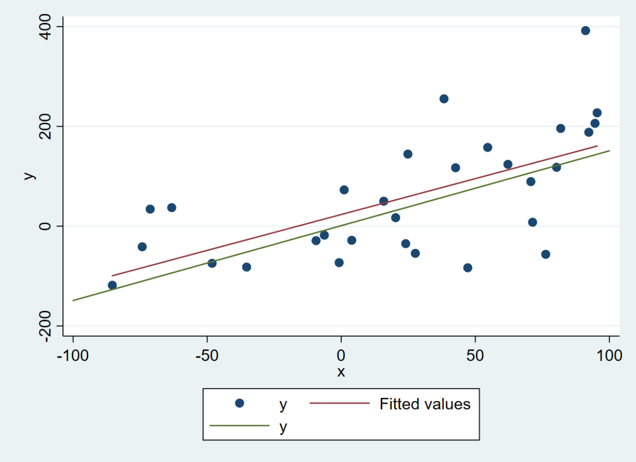
$$Y = 1 + 1.5X + \epsilon$$
$$\epsilon \sim N(0, 100)$$

  It is a noisy process.

- Simulate 30 samples from this process
- Let's start to fit it with different polynomials

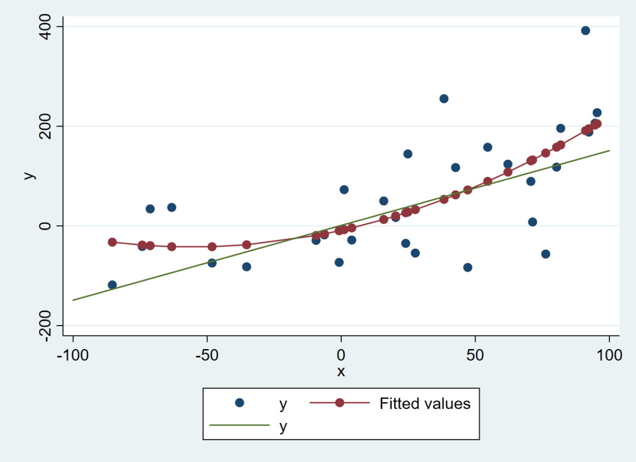
Machine Learning: What and Why

Figure: First Order (Linear) Fitting



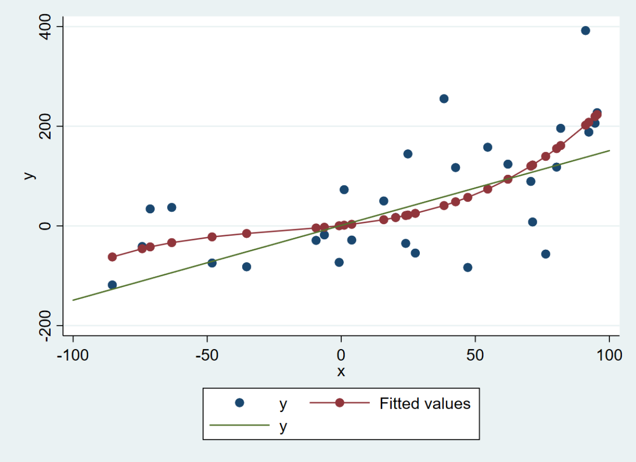
Machine Learning: What and Why

Figure: Second Order (Quadratic) Fitting



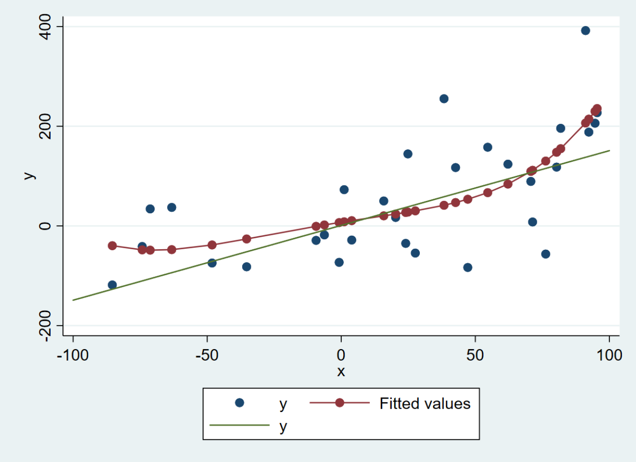
Machine Learning: What and Why

Figure: Third Order (Cubic) Fitting



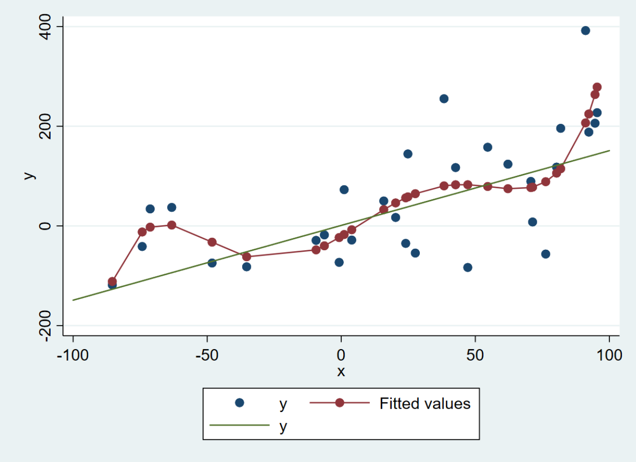
Machine Learning: What and Why

Figure: Fourth Order Fitting



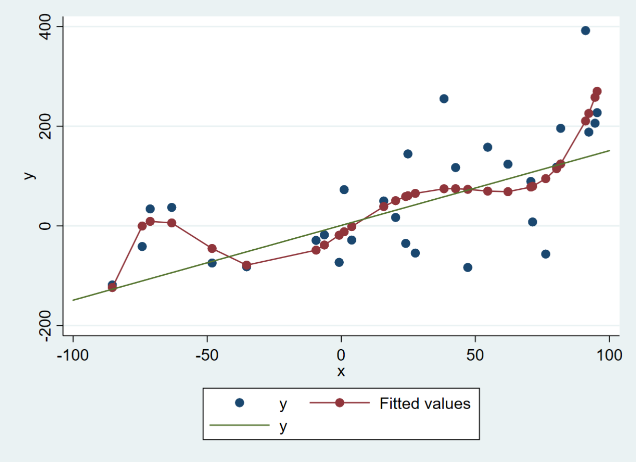
Machine Learning: What and Why

Figure: Fifth Order Fitting



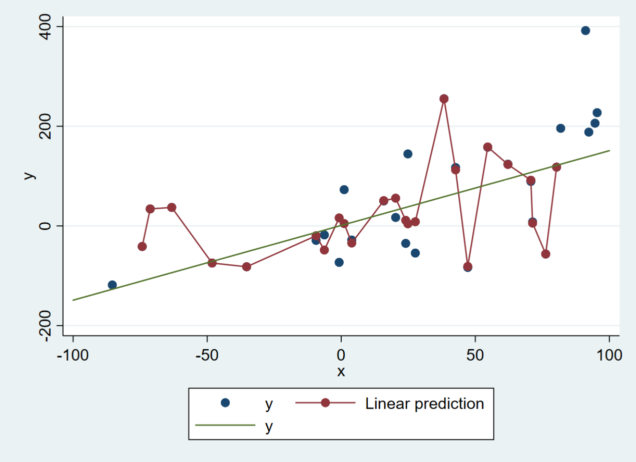
Machine Learning: What and Why

Figure: Sixth Order Fitting



Machine Learning: What and Why

Figure: Twentieth Order Fitting



Machine Learning: What and Why

Machine Learning: What and Why

- Main target: How complicated the model should be? How to *predict* Y given X?
- When Y is discrete: Classification
- When Y is continuous: Prediction

Machine Learning: What and Why

- Main target: How complicated the model should be? How to *predict* Y given X?
- When Y is discrete: Classification
- When Y is continuous: Prediction

Machine Learning: What and Why

- Main target: How complicated the model should be? How to *predict* Y given X?
- When Y is discrete: Classification
- When Y is continuous: Prediction

Machine Learning: Penalized Regressions

- Linear function: $y_i = x_i'\beta + \epsilon_i$
- OLS: $\hat{\beta}^{OLS} = argmin \sum_i (y_i - x_i'\beta)^2$
  All features x play roles.
- Penalized: $\hat{\beta}^{Pen} = argmin \sum_i (y_i - x_i'\beta)^2 + \lambda(\|\beta\|_p)^p$
  p=1: Lasso regression, drop some x with small prediction power
  p=2: Ridge regression, shrink some x with small prediction power
- λ: tuning parameter
- Combination: Elastic Net
  $\hat{\beta}^{Pen} = argmin \sum_i (y_i - x_i'\beta)^2 + \lambda(\alpha\|\beta\|_1 + (1-\alpha)(\|\beta\|_2)^2)$

Machine Learning: Penalized Regressions

- Linear function: $y_i = x_i'\beta + \epsilon_i$
- OLS: $\hat{\beta}^{OLS} = argmin \sum_i (y_i - x_i'\beta)^2$
  All features x play roles.
- Penalized: $\hat{\beta}^{Pen} = argmin \sum_i (y_i - x_i'\beta)^2 + \lambda(\|\beta\|_p)^p$
  p=1: Lasso regression, drop some x with small prediction power
  p=2: Ridge regression, shrink some x with small prediction power
- λ: tuning parameter
- Combination: Elastic Net
  $\hat{\beta}^{Pen} = argmin \sum_i (y_i - x_i'\beta)^2 + \lambda(\alpha\|\beta\|_1 + (1-\alpha)(\|\beta\|_2)^2)$

Machine Learning: Penalized Regressions

- Linear function: $y_i = x_i'\beta + \epsilon_i$
- OLS: $\hat{\beta}^{OLS} = argmin \sum_i (y_i - x_i'\beta)^2$
  All features x play roles.
- Penalized: $\hat{\beta}^{Pen} = argmin \sum_i (y_i - x_i'\beta)^2 + \lambda(\|\beta\|_p)^p$
  p=1: Lasso regression, drop some x with small prediction power
  p=2: Ridge regression, shrink some x with small prediction power
- λ: tuning parameter
- Combination: Elastic Net
  $\hat{\beta}^{Pen} = argmin \sum_i (y_i - x_i'\beta)^2 + \lambda(\alpha\|\beta\|_1 + (1-\alpha)(\|\beta\|_2)^2)$

Machine Learning: Penalized Regressions

- Linear function: $y_i = x_i'\beta + \epsilon_i$
- OLS: $\hat{\beta}^{OLS} = argmin \sum_i (y_i - x_i'\beta)^2$
  All features x play roles.
- Penalized: $\hat{\beta}^{Pen} = argmin \sum_i (y_i - x_i'\beta)^2 + \lambda(\|\beta\|_p)^p$
  p=1: Lasso regression, drop some x with small prediction power
  p=2: Ridge regression, shrink some x with small prediction power
- λ: tuning parameter
- Combination: Elastic Net
  $\hat{\beta}^{Pen} = argmin \sum_i (y_i - x_i'\beta)^2 + \lambda(\alpha\|\beta\|_1 + (1-\alpha)(\|\beta\|_2)^2)$

Machine Learning: Penalized Regressions

- Linear function: $y_i = x_i'\beta + \epsilon_i$
- OLS: $\hat{\beta}^{OLS} = argmin \sum_i (y_i - x_i'\beta)^2$
  All features x play roles.
- Penalized: $\hat{\beta}^{Pen} = argmin \sum_i (y_i - x_i'\beta)^2 + \lambda(\|\beta\|_p)^p$
  p=1: Lasso regression, drop some x with small prediction power
  p=2: Ridge regression, shrink some x with small prediction power
- λ: tuning parameter
- Combination: Elastic Net
  $\hat{\beta}^{Pen} = argmin \sum_i (y_i - x_i'\beta)^2 + \lambda(\alpha\|\beta\|_1 + (1-\alpha)(\|\beta\|_2)^2)$

Machine Learning: Tree Based Method

- Tree-based methods partition the feature (X) space into a set of rectangles, and then fit a simple model (constant) in each one.
- Classification and Regression Tree (CART)
- Partition into regions $R_1, R_2 ... R_M$, assign average value in a region as the predicted value
  $\hat{f}(x_i) = \sum_{m=1}^{M} c_m I(x \in R_m)$
- How to partition (Grow the tree)?

Machine Learning: Tree Based Method

- Tree-based methods partition the feature (X) space into a set of rectangles, and then fit a simple model (constant) in each one.
- Classification and Regression Tree (CART)
- Partition into regions $R_1, R_2 ... R_M$, assign average value in a region as the predicted value
  $\hat{f}(x_i) = \sum_{m=1}^{M} c_m I(x \in R_m)$
- How to partition (Grow the tree)?

Machine Learning: Tree Based Method

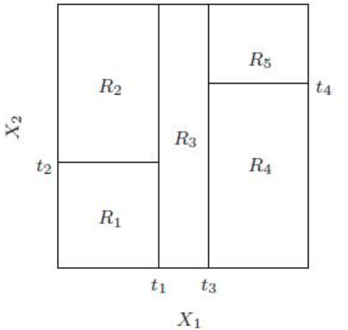
- Tree-based methods partition the feature (X) space into a set of rectangles, and then fit a simple model (constant) in each one.
- Classification and Regression Tree (CART)
- Partition into regions $R_1, R_2 ... R_M$, assign average value in a region as the predicted value
  $\hat{f}(x_i) = \sum_{m=1}^{M} c_m I(x \in R_m)$
- How to partition (Grow the tree)?

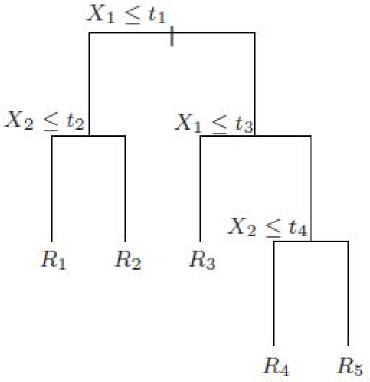
Machine Learning: Tree Based Method

- Tree-based methods partition the feature (X) space into a set of rectangles, and then fit a simple model (constant) in each one.
- Classification and Regression Tree (CART)
- Partition into regions $R_1, R_2 \ldots R_M$, assign average value in a region as the predicted value
  $\hat{f}(x_i) = \sum_{m=1}^{M} c_m I(x \in R_m)$
- How to partition (Grow the tree)?

Machine Learning: Tree Based Method

- Recursive binary partitions
- $(X_1, t_1) \rightarrow ((X_2, t_2), (X_1, t_3)) \rightarrow (X_2.t_4)$





Machine Learning: Tree Based Method

For each region R_m:

$$N_m = \{x_i \in R_m\}$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{x \in R_m} y_i$$

$$Q_m(T) = \frac{1}{N_m} \sum_{x \in R_m} (y_i - \hat{c}_m)^2$$

Two choices: where to partition + continue partitioning or stop
Greedy algorithm

Machine Learning: Tree Based Method

- Conditional on continuing grow, how to determine partition?
- How to find (j,s) in each branch? Minimize SSE (Easy)

$$\min_{j,s}[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$$

- How to grow the tree? (Continue growing or stop?)
- Too large $\to$ Overfitting; Too small $\to$ Losing information
- Grow a big tree T_0, then prune it!

Step 1: Grow T_0 when some minimum node size is reached (say 10)
Step 2: Pruning. Choose the tree $T \subset T_0$ with the lowest cost function $C_\alpha(T)$.

Machine Learning: Tree Based Method

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha|T|$$

α as the tuning parameter; $|T|$ as number of terminal nodes

- Total SSE + Size penalty
- α determines how hard to penalize tree size

Random Forests: Definition

- Using sub-sampling or bagging to reduce variance of a single tree
- Draw a lot of different samples (1,2,...B) with sub-sampling ($n < N$) (Jackknife) or bagging ($n = N$) (Bootstrap)

$$\hat{f}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

- Random Forests = Tree Method + Sampling average (Many Trees)

Random Forests: Definition

- Using sub-sampling or bagging to reduce variance of a single tree
- Draw a lot of different samples (1,2,...B) with sub-sampling ($n < N$) (Jackknife) or bagging ($n = N$) (Bootstrap)

$$\hat{f}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

- Random Forests = Tree Method + Sampling average (Many Trees)

Random Forests: Definition

- Using sub-sampling or bagging to reduce variance of a single tree
- Draw a lot of different samples (1,2,...B) with sub-sampling ($n < N$) (Jackknife) or bagging ($n = N$) (Bootstrap)

$$\hat{f}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

- Random Forests = Tree Method + Sampling average (Many Trees)

Machine Learning: What is Random Forests?

- A machine learning method similar to kernels and nearest-neighbor method
  Making predictions using weighted averages of "nearby" observations
- Difference: Weighting scheme
  NN: Not adaptive; RF: Adaptive

Machine Learning: What is Random Forests?

- A machine learning method similar to kernels and nearest-neighbor method
  Making predictions using weighted averages of "nearby" observations
- Difference: Weighting scheme
  NN: Not adaptive; RF: Adaptive

Causal Forests: Motivation

- Athey and Imbens(2016) Recursive Partitioning for Heterogeneous Causal Effects
- Wager and Athey(2016) Estimation and Inference of Heterogeneous Treatment Effects Using Random Forests

Causal Forests: Motivation

- Athey and Imbens(2016) Recursive Partitioning for Heterogeneous Causal Effects
- Wager and Athey(2016) Estimation and Inference of Heterogeneous Treatment Effects Using Random Forests

Causal Forests: Motivation

- Main topic in causal inference: Treatment effect
  Mostly ATE, LATE etc.
- Heterogeneous Treatment Effect
  Cherry picking? $\Rightarrow$ Institutional restrictions on trials
- Unexpected heterogeneity

Causal Forests: Motivation

- Main topic in causal inference: Treatment effect
  Mostly ATE, LATE etc.
- Heterogeneous Treatment Effect
  Cherry picking? $\Rightarrow$ Institutional restrictions on trials
- Unexpected heterogeneity

Causal Forests: Motivation

- Main topic in causal inference: Treatment effect
  Mostly ATE, LATE etc.
- Heterogeneous Treatment Effect
  Cherry picking? $\Rightarrow$ Institutional restrictions on trials
- Unexpected heterogeneity

Causal Forests: Main Contribution

- Developing a machine learning tool, Causal Forests (An extension of Random Forests), to reveal the true underlying heterogeneous treatment effects
- Prove the consistency and asymptotic normality $\Rightarrow$ valid CI and inference

Causal Forests: Main Contribution

- Developing a machine learning tool, Causal Forests (An extension of Random Forests), to reveal the true underlying heterogeneous treatment effects
- Prove the consistency and asymptotic normality $\Rightarrow$ valid CI and inference

Causal Forests: Definition

- Why RF?: tells us how to divide groups to get the "real" heterogeneous TE
- Data of (X_i, Y_i, W_i), W_i is treatment assignment. L as a leaf (region).
- Treatment effect: $\tau(x) = E[Y_i^{(1)} - Y_i^{(0)} | X_i = x]$
- Unconfoundness: $\{Y_i^{(0)}, Y_i^{(1)}\} \perp W_i | X_i$

Causal Forests: Definition

- Why RF?: tells us how to divide groups to get the "real" heterogeneous TE
- Data of (X_i, Y_i, W_i), W_i is treatment assignment. L as a leaf (region).
- Treatment effect: $\tau(x) = E[Y_i^{(1)} - Y_i^{(0)} | X_i = x]$
- Unconfoundness: $\{Y_i^{(0)}, Y_i^{(1)}\} \perp W_i | X_i$

Causal Forests: Definition

- Why RF?: tells us how to divide groups to get the "real" heterogeneous TE
- Data of (X_i, Y_i, W_i), W_i is treatment assignment. L as a leaf (region).
- Treatment effect: $\tau(x) = E[Y_i^{(1)} - Y_i^{(0)} | X_i = x]$
- Unconfoundness: $\{Y_i^{(0)}, Y_i^{(1)}\} \perp W_i | X_i$

Causal Forests: Definition

- Why RF?: tells us how to divide groups to get the "real" heterogeneous TE
- Data of (X_i, Y_i, W_i), W_i is treatment assignment. L as a leaf (region).
- Treatment effect: $\tau(x) = E[Y_i^{(1)} - Y_i^{(0)} | X_i = x]$
- Unconfoundness: $\{Y_i^{(0)}, Y_i^{(1)}\} \perp W_i | X_i$

Causal Forests: Definition

- Estimation of TE: Given x in leaf $L(x)$, the difference of the average outcome Y for treated/non-treated group

$$\hat{\tau}(x) = \frac{1}{|\{i: W_i = 1, X_i \in L\}|} \sum_{\{i: W_i = 1, X_i \in L\}}^{Y_i} Y_i - \frac{1}{|\{i: W_i = 0, X_i \in L\}|} \sum_{\{i: W_i = 0, X_i \in L\}}^{Y_i} Y_i$$

- Implement the Random Forests using a splitting rule: maximize variance of $\hat{\tau}(X_i)$
- The criterion is a natural analogue of the original one (min SSE)
- Causal Forests is a Nearest Neighbor with *adaptive neighborhood*

Causal Forests: Definition

- Estimation of TE: Given x in leaf $L(x)$, the difference of the average outcome Y for treated/non-treated group

$$\hat{\tau}(x) = \frac{1}{|\{i:W_i=1, X_i \in L\}|} \sum_{\{i:W_i=1, X_i \in L\}}^{Y_i} Y_i - \frac{1}{|\{i:W_i=0, X_i \in L\}|} \sum_{\{i:W_i=0, X_i \in L\}}^{Y_i} Y_i$$

- Implement the Random Forests using a splitting rule: maximize variance of $\hat{\tau}(X_i)$
- The criterion is a natural analogue of the original one (min SSE)
- Causal Forests is a Nearest Neighbor with *adaptive neighborhood*

Causal Forests: Definition

- Estimation of TE: Given x in leaf $L(x)$, the difference of the average outcome Y for treated/non-treated group
  $\hat{\tau}(x) = \frac{1}{|\{i: W_i = 1, X_i \in L\}|} \sum_{\{i: W_i = 1, X_i \in L\}}^{Y_i} Y_i - \frac{1}{|\{i: W_i = 0, X_i \in L\}|} \sum_{\{i: W_i = 0, X_i \in L\}}^{Y_i} Y_i$
- Implement the Random Forests using a splitting rule: maximize variance of $\hat{\tau}(X_i)$
- The criterion is a natural analogue of the original one (min SSE)
- Causal Forests is a Nearest Neighbor with *adaptive neighborhood*

Causal Forests: Definition

- Estimation of TE: Given x in leaf $L(x)$, the difference of the average outcome Y for treated/non-treated group

$$\hat{\tau}(x) = \frac{1}{|\{i: W_i=1, X_i \in L\}|} \sum_{\{i: W_i=1, X_i \in L\}}^{Y_i} Y_i - \frac{1}{|\{i: W_i=0, X_i \in L\}|} \sum_{\{i: W_i=0, X_i \in L\}}^{Y_i} Y_i$$

- Implement the Random Forests using a splitting rule: maximize variance of $\hat{\tau}(X_i)$
- The criterion is a natural analogue of the original one (min SSE)
- Causal Forests is a Nearest Neighbor with *adaptive neighborhood*

Causal Forests: Honest Trees and Forests

- A tree is honesty, if for each training example i, it is either used to estimate τ or used to decide splits
- Double-Sample Trees: Averagely divide samples into two parts I and J. Grow the tree using I and then estimate τ in each leaf using J.
- Honest Causal Forests is consistent and asymptotically normal

Causal Forests: Honest Trees and Forests

- A tree is honesty, if for each training example i, it is either used to estimate τ or used to decide splits
- Double-Sample Trees: Averagely divide samples into two parts I and J. Grow the tree using I and then estimate τ in each leaf using J.
- Honest Causal Forests is consistent and asymptotically normal

Causal Forests: Honest Trees and Forests

- A tree is honesty, if for each training example i, it is either used to estimate τ or used to decide splits
- Double-Sample Trees: Averagely divide samples into two parts I and J. Grow the tree using I and then estimate τ in each leaf using J.
- Honest Causal Forests is consistent and asymptotically normal