SIRCSS – Machine Learning for Social Science

Lecture 3

Martin Arvidsson | Institute of Analytical Sociology, Linköping University

2025-12-03

Recap of yesterday

Recap of yesterday

Neural networks

Recap of yesterday

Neural networks

- 0 hidden layers $+$ softmax $\rightarrow$ standard logit

Recap of yesterday

Neural networks

- 0 hidden layers + softmax → standard logit
- Add hidden layer → learn new representations (linear combos of X)

Recap of yesterday

Neural networks

- 0 hidden layers + softmax $\rightarrow$ standard logit
- Add hidden layer $\rightarrow$ learn new representations (linear combos of X)
- Non-linear transformation $\rightarrow$ capture interactions & non-linearities

Recap of yesterday

Neural networks

- 0 hidden layers + softmax $\rightarrow$ standard logit
- Add hidden layer $\rightarrow$ learn new representations (linear combos of X)
- Non-linear transformation $\rightarrow$ capture interactions & non-linearities
- More layers $\rightarrow$ richer representations (complex patterns)

Recap of yesterday

Neural networks

- 0 hidden layers + softmax → standard logit
- Add hidden layer → learn new representations (linear combos of X)
- Non-linear transformation → capture interactions & non-linearities
- More layers → richer representations (complex patterns)
- Usually need lots of data (/w complex patterns) to be competitive

Recap of yesterday

Neural networks

- 0 hidden layers + softmax → standard logit
- Add hidden layer → learn new representations (linear combos of X)
- Non-linear transformation → capture interactions & non-linearities
- More layers → richer representations (complex patterns)
- Usually need lots of data (/w complex patterns) to be competitive
- Best performance — specialized architectures which facilitate feature learning of complex data

Recap of yesterday

Neural networks

- 0 hidden layers + softmax $\rightarrow$ standard logit
- Add hidden layer $\rightarrow$ learn new representations (linear combos of X)
- Non-linear transformation $\rightarrow$ capture interactions & non-linearities
- More layers $\rightarrow$ richer representations (complex patterns)
- Usually need lots of data (/w complex patterns) to be competitive
- Best performance — specialized architectures which facilitate feature learning of complex data

Convolutional Neural Networks (CNN)

Recap of yesterday

Neural networks

- 0 hidden layers + softmax → standard logit
- Add hidden layer → learn new representations (linear combos of X)
- Non-linear transformation → capture interactions & non-linearities
- More layers → richer representations (complex patterns)
- Usually need lots of data (/w complex patterns) to be competitive
- Best performance — specialized architectures which facilitate feature learning of complex data

Convolutional Neural Networks (CNN)

- Specialized architecture for *image* prediction

Recap of yesterday

Neural networks

- 0 hidden layers + softmax → standard logit
- Add hidden layer → learn new representations (linear combos of X)
- Non-linear transformation → capture interactions & non-linearities
- More layers → richer representations (complex patterns)
- Usually need lots of data (/w complex patterns) to be competitive
- Best performance — specialized architectures which facilitate feature learning of complex data

Convolutional Neural Networks (CNN)

- Specialized architecture for *image* prediction
- *Convolutional layers* learn spatial patterns through *filters*

Recap of yesterday

Neural networks

- 0 hidden layers + softmax → standard logit
- Add hidden layer → learn new representations (linear combos of X)
- Non-linear transformation → capture interactions & non-linearities
- More layers → richer representations (complex patterns)
- Usually need lots of data (/w complex patterns) to be competitive
- Best performance — specialized architectures which facilitate feature learning of complex data

Convolutional Neural Networks (CNN)

- Specialized architecture for *image* prediction
- *Convolutional layers* learn spatial patterns through *filters*
- Early layers: *basic* features (edges); deeper layers: *complex* objects

Recap of yesterday

Neural networks

- 0 hidden layers + softmax → standard logit
- Add hidden layer → learn new representations (linear combos of X)
- Non-linear transformation → capture interactions & non-linearities
- More layers → richer representations (complex patterns)
- Usually need lots of data (/w complex patterns) to be competitive
- Best performance — specialized architectures which facilitate feature learning of complex data

Convolutional Neural Networks (CNN)

- Specialized architecture for *image* prediction
- *Convolutional layers* learn spatial patterns through *filters*
- Early layers: *basic* features (edges); deeper layers: *complex* objects
- Conv-outputs feed into *regular* (*dense*) layers for prediction

Today

- Recap causal inference basics + limitations of standard paradigm
- Machine learning for causal inference
    - Addressing non-linearity
    - Heterogeneous treatment effects

Causal inference basics

Causal inference basics

Type of question: What is the causal effect of X on Y?

$$\boxed{X} \longrightarrow \boxed{Y}$$

Causal inference basics

Type of question: What is the causal effect of X on Y?

Causal inference basics

Type of question: What is the causal effect of X on Y?

How to estimate the (causal) effect X on Y?

How to estimate the (causal) effect X on Y?

Just regress X on Y?

$$Y = \beta_0 + \beta_1 X$$

How to estimate the (causal) effect X on Y?

Just regress X on Y?

$$Y = \beta_0 + \beta_1 X$$

$$\boxed{X} \xrightarrow{\beta_1} \boxed{Y}$$

How to estimate the (causal) effect X on Y?

Just regress X on Y?

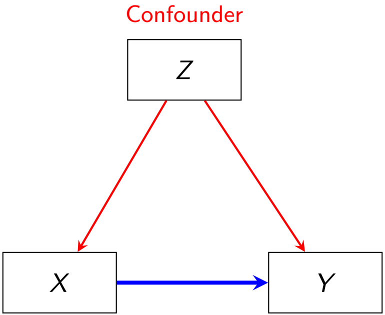
$$Y = \beta_0 + \beta_1 X$$

$$\boxed{X} \xrightarrow{\beta_1} \boxed{Y}$$

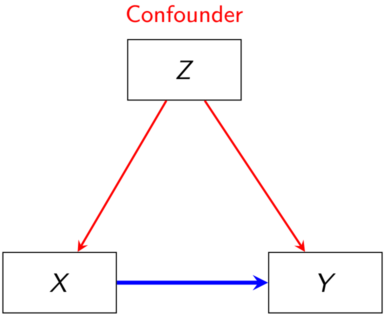
This can work — sometimes. When:

- No other variable Z that affects both X and Y (no confounder)
- Or, X is randomized.

When we have confounders



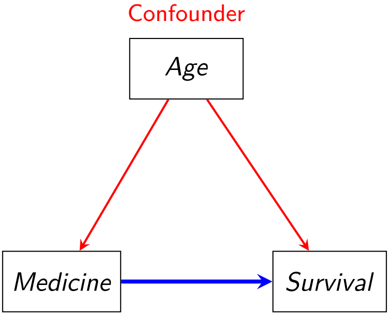
When we have confounders



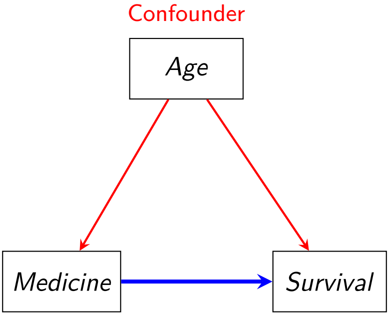
- If we have a confounder (Z), we must account for it somehow.
- Commonly — include as ‘control variable’ in regression.

$$Y = \beta_0 + \beta_1 X + \beta_2 Z$$

When we have confounders



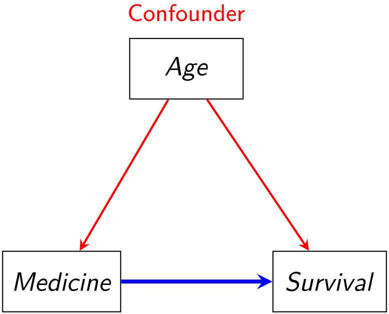
When we have confounders



Older people:

- More likely to take medicine
- Lower survival rate

When we have confounders



Older people:

- More likely to take medicine
- Lower survival rate

$\rightarrow$ If we don't control for age, it may look like medicine hurts survival.

Illustration – not controlling for age

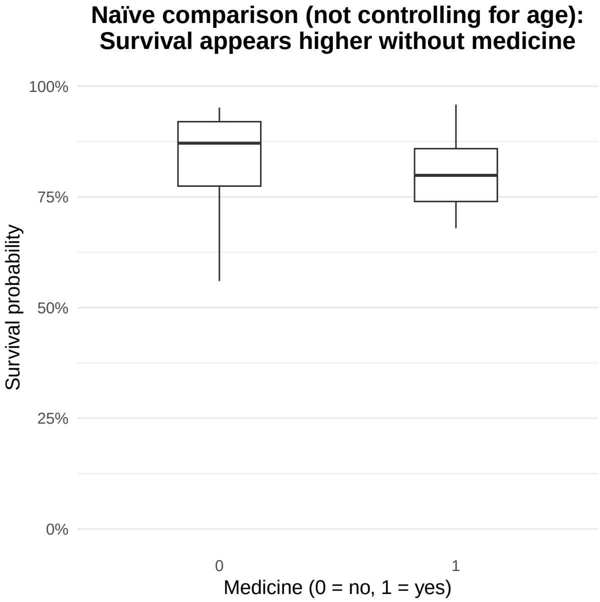
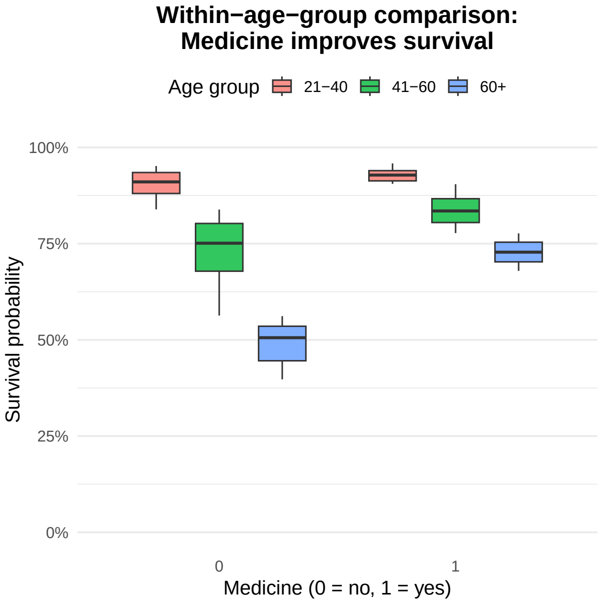


Illustration – controlling for age



Why move beyond standard paradigm? Why this lecture?

Why move beyond standard paradigm? Why this lecture?

The standard paradigm. . .

(A) Assumes linearity.

- When $Z \rightarrow X$ and/or $Z \rightarrow Y$ is non-linear: biased estimates.

Why move beyond standard paradigm? Why this lecture?

The standard paradigm. . .

(A) Assumes linearity.

- When $Z \to X$ and/or $Z \to Y$ is non-linear: biased estimates.

(B) Assumes we know which variables that moderate effect beforehand.

- Often not the case.
- Especially when we have many variables.

Why move beyond standard paradigm? Why this lecture?

The standard paradigm. . .

(A) Assumes linearity.

- When $Z \rightarrow X$ and/or $Z \rightarrow Y$ is non-linear: biased estimates.

(B) Assumes we know which variables that moderate effect beforehand.

- Often not the case.
- Especially when we have many variables.

(C) Struggles with 'complex' data like images, text, and networks.

- Often contain rich latent structures.
- Not accounted for by standard paradigm.

Non-linearity

Why is non-linearity a problem?

- Adjusting for a confounder is about comparing "like with like".[1]

[1] E.g., comparing two similarly aged people taking a medicine.

Why is non-linearity a problem?

- Adjusting for a confounder is about comparing “like with like”.[1]

[1] E.g., comparing two similarly aged people taking a medicine.

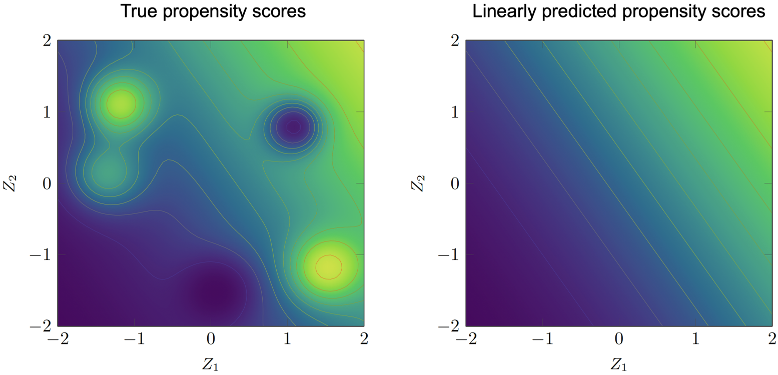
Why is non-linearity a problem?

- Adjusting for a confounder is about comparing “like with like”.[1]
- If $Z \rightarrow X$ and $Z \rightarrow Y$ are **curved**, but we adjust for a **straight line**; we don’t actually compare like with like.

[1] E.g., comparing two similarly aged people taking a medicine.

Why is non-linearity a problem?

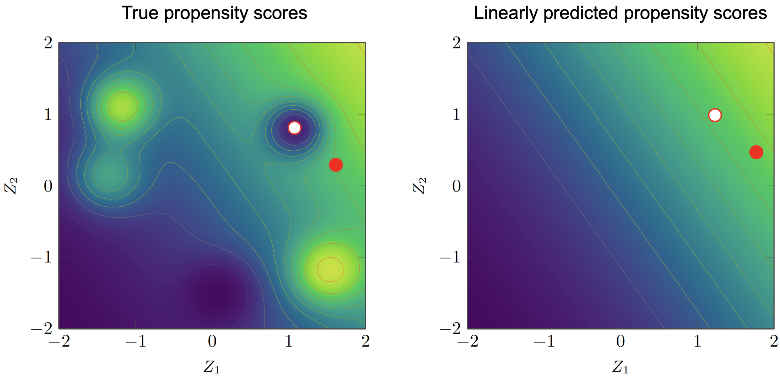
- Adjusting for a confounder is about comparing "like with like".[1]
- If $Z \rightarrow X$ and $Z \rightarrow Y$ are **curved**, but we adjust for a **straight line**; we don't actually compare like with like.



[1] E.g., comparing two similarly aged people taking a medicine.

Why is non-linearity a problem?

- Adjusting for a confounder is about comparing "like with like".[2]
- If $Z \to X$ and $Z \to Y$ are curved, but we adjust for a **straight line**; we don't actually compare like with like.



[2] E.g., comparing two similarly aged people taking a medicine.

How can machine learning help?

- **ML**: powerful for learning non-linear mappings: $f(X) \approx Y$.
  - In particular — *non-parametric models*.
  - Do not impose functional form on f; learn from data.

How can machine learning help?

- **ML**: powerful for learning non-linear mappings: $f(X) \approx Y$.
  - In particular — *non-parametric models*.
  - Do not impose functional form on f; learn from data.
- But — we also said that these models were bad at inference. . . ?
  - Not able to say: controlling for Z, X affects Y in so and so way.
  - E.g., ridge penalty intentionally *adds bias* to *reduce variance*.

How can machine learning help?

- **ML**: powerful for learning non-linear mappings: $f(X) \approx Y$.
  - In particular — *non-parametric models*.
  - Do not impose functional form on f; learn from data.
- But — we also said that these models were bad at inference. . . ?
  - Not able to say: controlling for Z, X affects Y in so and so way.
  - E.g., ridge penalty intentionally *adds bias* to *reduce variance*.
- Can ML still be useful?
  —Yes, by reframing causal inference as prediction problems.

Meta-learners

Meta-learners

Class of models that:

- Reduce causal estimation to *prediction tasks*
- Are not tied to any particular prediction method
- Different learners (S,T,X,R) combine pred-models in different ways

Meta-learners

Class of models that:

- Reduce causal estimation to *prediction tasks*
- Are not tied to any particular prediction method
- Different learners (S,T,X,R) combine pred-models in different ways

Today, focus on two key ones:

- T-learner
- Orthogonal learner

T-learner

Builds on the classic **potential outcomes framework**:

$$\text{ATE} = \frac{1}{N} \sum_{i}^{N} Y_i^1 - Y_i^0$$

T-learner

Builds on the classic **potential outcomes framework**:

$$\text{ATE} = \frac{1}{N} \sum_{i}^{N} Y_i^1 - Y_i^0$$

- Where Y_i^1, Y_i^0 are the potential outcomes for individual i.

T-learner

Builds on the classic **potential outcomes framework**:

$$\text{ATE} = \frac{1}{N}\sum_{i}^{N} Y_i^1 - Y_i^0$$

- Where Y_i^1, Y_i^0 are the potential outcomes for individual i.
- **Challenge**: we only observe *one potential outcome* per individual.

T-learner

Builds on the classic **potential outcomes framework**:

$$\text{ATE} = \frac{1}{N}\sum_{i}^{N} Y_i^1 - Y_i^0$$

- Where Y_i^1, Y_i^0 are the potential outcomes for individual i.
- **Challenge**: we only observe *one potential outcome* per individual.

Individual	X_i	Y	Y_i^0	Y_i^1	Z_i
1	1	7	?	7	13
2	1	9	?	9	9
3	1	6	?	6	16
4	0	3	3	?	8
5	0	4	4	?	12

T-learner

Builds on the classic **potential outcomes framework**:

$$\text{ATE} = \frac{1}{N}\sum_{i}^{N} Y_i^1 - Y_i^0$$

- Where Y_i^1, Y_i^0 are the potential outcomes for individual i.
- **Challenge**: we only observe *one potential outcome* per individual.

Individual	X_i	Y	Y_i^0	Y_i^1	Z_i
1	1	7	?	7	13
2	1	9	?	9	9
3	1	6	?	6	16
4	0	3	3	?	8
5	0	4	4	?	12

T-learner: predict potential outcomes ("fill the gaps")

T-learner procedure:

1. Train prediction model $f_1(Z) \approx Y$ on the treated $(X = 1)$

T-learner procedure:

1. Train prediction model $f_1(Z) \approx Y$ on the treated $(X = 1)$
2. Train prediction model $f_0(Z) \approx Y$ on the untreated $(X = 0)$

T-learner procedure:

1. Train prediction model $f_1(Z) \approx Y$ on the treated $(X = 1)$
2. Train prediction model $f_0(Z) \approx Y$ on the untreated $(X = 0)$
3. For each individual, predict $\hat{Y}^1, \hat{Y}^0$ using f_1, f_0

T-learner procedure:

1. Train prediction model $f_1(Z) \approx Y$ on the treated $(X = 1)$
2. Train prediction model $f_0(Z) \approx Y$ on the untreated $(X = 0)$
3. For each individual, predict $\hat{Y}^1, \hat{Y}^0$ using f_1, f_0
4. Compute $\text{ATE} = \frac{1}{N} \sum_i^N \hat{Y}_i^1 - \hat{Y}_i^0$

T-learner procedure:

1. Train prediction model $f_1(Z) \approx Y$ on the treated $(X = 1)$
2. Train prediction model $f_0(Z) \approx Y$ on the untreated $(X = 0)$
3. For each individual, predict $\hat{Y}^1, \hat{Y}^0$ using f_1, f_0
4. Compute ATE $= \frac{1}{N} \sum_i^N \hat{Y}_i^1 - \hat{Y}_i^0$

What's neat is:

- Adressing non-linearity — we can use highly flexible $f_1(Z)$, $f_0(Z)$

T-learner procedure:

1. Train prediction model $f_1(Z) \approx Y$ on the treated $(X = 1)$
2. Train prediction model $f_0(Z) \approx Y$ on the untreated $(X = 0)$
3. For each individual, predict $\hat{Y}^1, \hat{Y}^0$ using f_1, f_0
4. Compute ATE $= \frac{1}{N} \sum_i^N \hat{Y}_i^1 - \hat{Y}_i^0$

What's neat is:

- Adressing non-linearity — we can use highly flexible $f_1(Z)$, $f_0(Z)$
- No concern about inference/interpretability of f_1, f_2 — they are only used to predict components of a well-defined, interpretable effect.

T-learner procedure:

1. Train prediction model $f_1(Z) \approx Y$ on the treated $(X = 1)$
2. Train prediction model $f_0(Z) \approx Y$ on the untreated $(X = 0)$
3. For each individual, predict $\hat{Y}^1, \hat{Y}^0$ using f_1, f_0
4. Compute $\text{ATE} = \frac{1}{N} \sum_i^N \hat{Y}_i^1 - \hat{Y}_i^0$

What's neat is:

- Adressing non-linearity — we can use highly flexible $f_1(Z)$, $f_0(Z)$
- No concern about inference/interpretability of f_1, f_2 — they are only used to predict components of a well-defined, interpretable effect.

Note: the name 'T-learner' stems from it learning *two* prediction models.[3]

[3] There exists an S-learner which only learns a single prediction model.

T-learner applied to the toy data

Individual	X_i	Y	$\hat{Y}_i^0$	$\hat{Y}_i^1$	Z_i
1	1	7	5	7	13
2	1	9	8	10	9
3	1	6	4	5	16
4	0	3	3	5	8
5	0	4	4	6	12

T-learner applied to the toy data

Individual	X_i	Y	$\hat{Y}_i^0$	$\hat{Y}_i^1$	Z_i
1	1	7	5	7	13
2	1	9	8	10	9
3	1	6	4	5	16
4	0	3	3	5	8
5	0	4	4	6	12

$$\text{ATE} = \frac{1}{5}[(7-5)+(10-8)+(5-4)+(5-3)+(6-4)] = 1.8$$

Summary, T-learner

- Causal inference as prediction of potential outcomes.
- Makes use of supervised ML.
- Can *improve estimation* considerably when data is complex.
- But note: still relies on the assumption of no unobserved confounders.
- Limitation: splits data by *treatment status* $\Rightarrow$ data inefficiency[4]

[4] Less data to fit each model, and thus less able to learn patterns.

Orthogonal learner

Orthogonal learner

Basic idea: predict treatment (X) and outcome (Y) based on confounders (Z) — and extract residuals to estimate causal effect.

Orthogonal learner

Basic idea: predict treatment (X) and outcome (Y) based on confounders (Z) — and extract residuals to estimate causal effect.

1. Predict treatment X from Z: $\hat{X} = f_X(Z)$

Orthogonal learner

Basic idea: predict treatment (X) and outcome (Y) based on confounders (Z) — and extract residuals to estimate causal effect.

1. Predict treatment X from Z: $\hat{X} = f_X(Z)$
2. Predict outcome Y from Z: $\hat{Y} = f_Y(Z)$

Orthogonal learner

Basic idea: predict treatment (X) and outcome (Y) based on confounders (Z) — and extract residuals to estimate causal effect.

1. Predict treatment X from Z: $\hat{X} = f_X(Z)$
2. Predict outcome Y from Z: $\hat{Y} = f_Y(Z)$
3. Calculate residuals:
   - $\bar{X} = X - \hat{X}$
   - $\bar{Y} = Y - \hat{Y}$

Orthogonal learner

Basic idea: predict treatment (X) and outcome (Y) based on confounders (Z) — and extract residuals to estimate causal effect.

1. Predict treatment X from Z: $\hat{X} = f_X(Z)$
2. Predict outcome Y from Z: $\hat{Y} = f_Y(Z)$
3. Calculate residuals:
    - $\bar{X} = X - \hat{X}$
    - $\bar{Y} = Y - \hat{Y}$
4. Estimate ATE by regressing: $\bar{Y} = \beta\bar{X}$

Orthogonal learner

Basic idea: predict treatment (X) and outcome (Y) based on confounders (Z) — and extract residuals to estimate causal effect.

1. Predict treatment X from Z: $\hat{X} = f_X(Z)$
2. Predict outcome Y from Z: $\hat{Y} = f_Y(Z)$
3. Calculate residuals:
   - $\bar{X} = X - \hat{X}$
   - $\bar{Y} = Y - \hat{Y}$
4. Estimate ATE by regressing: $\bar{Y} = \beta \bar{X}$

Intuition for why this works:

- Subtracting predictions based on $Z \Rightarrow$ removes association with Z

Orthogonal learner

Basic idea: predict treatment (X) and outcome (Y) based on confounders (Z) — and extract residuals to estimate causal effect.

1. Predict treatment X from Z: $\hat{X} = f_X(Z)$
2. Predict outcome Y from Z: $\hat{Y} = f_Y(Z)$
3. Calculate residuals:
   - $\bar{X} = X - \hat{X}$
   - $\bar{Y} = Y - \hat{Y}$
4. Estimate ATE by regressing: $\bar{Y} = \beta\bar{X}$

Intuition for why this works:

- Subtracting predictions based on $Z \Rightarrow$ removes association with Z
- As a result, $\hat{X}, \hat{Y}$ becomes independent of Z.

Orthogonal learner

Basic idea: predict treatment (X) and outcome (Y) based on confounders (Z) — and extract residuals to estimate causal effect.

1. Predict treatment X from Z: $\hat{X} = f_X(Z)$
2. Predict outcome Y from Z: $\hat{Y} = f_Y(Z)$
3. Calculate residuals:
   - $\bar{X} = X - \hat{X}$
   - $\bar{Y} = Y - \hat{Y}$
4. Estimate ATE by regressing: $\bar{Y} = \beta\bar{X}$

Intuition for why this works:

- Subtracting predictions based on $Z \Rightarrow$ removes association with Z
- As a result, $\hat{X}, \hat{Y}$ becomes independent of Z.
- And the treatment as good as random.

Orthogonal learner

Basic idea: predict treatment (X) and outcome (Y) based on confounders (Z) — and extract residuals to estimate causal effect.

1. Predict treatment X from Z: $\hat{X} = f_X(Z)$
2. Predict outcome Y from Z: $\hat{Y} = f_Y(Z)$
3. Calculate residuals:
   - $\bar{X} = X - \hat{X}$
   - $\bar{Y} = Y - \hat{Y}$
4. Estimate ATE by regressing: $\bar{Y} = \beta \bar{X}$

Intuition for why this works:

- Subtracting predictions based on $Z \Rightarrow$ removes association with Z
- As a result, $\hat{X}, \hat{Y}$ becomes independent of Z.
- And the treatment as good as random.

Again, what's neat is:

- Adressing non-linearity — we can use highly flexible $f_X(Z)$, $f_Y(Z)$
- No concern about inference/interpretability.

Orthogonal learner applied to toy data

Orthogonal learner applied to toy data

Step 1: Predict X and Y from Z using flexible ML models

Orthogonal learner applied to toy data

Step 1: Predict X and Y from Z using flexible ML models

Individual	X_i	Y_i	$\hat{X}_i$	$\hat{Y}_i$	Z_i
1	1	7	0.7	6.5	13
2	1	9	0.6	8.2	9
3	1	6	0.8	4.8	16
4	0	3	0.4	3.5	8
5	0	4	0.5	4.1	12

Orthogonal learner applied to toy data

Step 2: Compute residuals

Orthogonal learner applied to toy data

Step 2: Compute residuals

Individual	$\tilde{X}_i = X_i - \hat{X}_i$	$\tilde{Y}_i = Y_i - \hat{Y}_i$	Z_i
1	$1 - 0.7 = 0.3$	$7 - 6.5 = 0.5$	13
2	$1 - 0.6 = 0.4$	$9 - 8.2 = 0.8$	9
3	$1 - 0.8 = 0.2$	$6 - 4.8 = 1.2$	16
4	$0 - 0.4 = -0.4$	$3 - 3.5 = -0.5$	8
5	$0 - 0.5 = -0.5$	$4 - 4.1 = -0.1$	12

Orthogonal learner applied to toy data

Step 2: Compute residuals

Individual	$\tilde{X}_i = X_i - \hat{X}_i$	$\tilde{Y}_i = Y_i - \hat{Y}_i$	Z_i
1	$1 - 0.7 = 0.3$	$7 - 6.5 = 0.5$	13
2	$1 - 0.6 = 0.4$	$9 - 8.2 = 0.8$	9
3	$1 - 0.8 = 0.2$	$6 - 4.8 = 1.2$	16
4	$0 - 0.4 = -0.4$	$3 - 3.5 = -0.5$	8
5	$0 - 0.5 = -0.5$	$4 - 4.1 = -0.1$	12

Step 3: Estimate effect by regressing $\tilde{Y}$ on $\tilde{X}$

Overfitting and solution

Problem: ML models can overfit, contaminating in-sample predictions.[5]

[5] Note: this problem applies to all meta-learners.

Overfitting and solution

Problem: ML models can overfit, contaminating in-sample predictions.[5]

[5] Note: this problem applies to all meta-learners.

Overfitting and solution

Problem: ML models can overfit, contaminating in-sample predictions.[5]

- Orthogonal learner: if $f_Y(Z)$ is overfitted, it captures not only *confounding* ($Z \rightarrow Y$) but also part of *treatment signal* ($X \rightarrow Y$).

[5] Note: this problem applies to all meta-learners.

Overfitting and solution

Problem: ML models can overfit, contaminating in-sample predictions.[5]

- Orthogonal learner: if $f_Y(Z)$ is overfitted, it captures not only *confounding* ($Z \rightarrow Y$) but also part of *treatment signal* ($X \rightarrow Y$).
- This leads to *residuals* (variation in Y due to X) becoming *too small*, biasing effect towards 0.

[5] Note: this problem applies to all meta-learners.

Overfitting and solution

Problem: ML models can overfit, contaminating in-sample predictions.[5]

- Orthogonal learner: if $f_Y(Z)$ is overfitted, it captures not only *confounding* ($Z \to Y$) but also part of *treatment signal* ($X \to Y$).
- This leads to *residuals* (variation in Y due to X) becoming *too small*, biasing effect towards 0.

Solution: Cross-fitting — predict out-of-fold to block contamination.

[5] Note: this problem applies to all meta-learners.

Overfitting and solution

Problem: ML models can overfit, contaminating in-sample predictions.[5]

- Orthogonal learner: if $f_Y(Z)$ is overfitted, it captures not only *confounding* ($Z \to Y$) but also part of *treatment signal* ($X \to Y$).
- This leads to *residuals* (variation in Y due to X) becoming *too small*, biasing effect towards 0.

Solution: Cross-fitting — predict out-of-fold to block contamination.

- Orthogonal learner: residuals in hold-out not made smaller.

[5]Note: this problem applies to all meta-learners.

Orthogonal learning procedure — with cross-fitting

0) Split data into K folds.

Orthogonal learning procedure — with cross-fitting

0) Split data into K folds.
1) For each fold i:
    - Train model $f_X(Z) \approx X$ on folds $\neq i$
    - Train model $f_Y(Z) \approx Y$ on folds $\neq i$
    - For every individual in fold i, predict & compute residuals:
        - $\tilde{X}_i = X_i - f_X(Z_i)$
        - $\tilde{Y}_i = Y_i - f_Y(Z_i)$

Orthogonal learning procedure — with cross-fitting

0) Split data into K folds.
1) For each fold i:
   - Train model $f_X(Z) \approx X$ on folds $\neq i$
   - Train model $f_Y(Z) \approx Y$ on folds $\neq i$
   - For every individual in fold i, predict & compute residuals:
     - $\tilde{X}_i = X_i - f_X(Z_i)$
     - $\tilde{Y}_i = Y_i - f_Y(Z_i)$
2) Stack residuals from all folds

Orthogonal learning procedure — with cross-fitting

0) Split data into K folds.
1) For each fold i:
   - Train model $f_X(Z) \approx X$ on folds $\neq i$
   - Train model $f_Y(Z) \approx Y$ on folds $\neq i$
   - For every individual in fold i, predict & compute residuals:
     - $\tilde{X}_i = X_i - f_X(Z_i)$
     - $\tilde{Y}_i = Y_i - f_Y(Z_i)$
2) Stack residuals from all folds
3) Estimate ATE by regressing: $\tilde{Y} = \beta \tilde{X}$

Demonstration using simulated data

To examine whether we can get improvements using ML $\Rightarrow$ simulate "ground truth" data and try to recover true effect.

Demonstration using simulated data

To examine whether we can get improvements using ML $\Rightarrow$ simulate "ground truth" data and try to recover true effect.

Basic setup

- Simulate data varying two dimensions:
  - Complexity: (1) *linear*, (2) *non-linear*
  - Size: (1) *small*, (2) *large*

Demonstration using simulated data

To examine whether we can get improvements using ML $\Rightarrow$ simulate "ground truth" data and try to recover true effect.

Basic setup

- Simulate data varying two dimensions:
  - Complexity: (1) *linear*, (2) *non-linear*
  - Size: (1) *small*, (2) *large*
- Compare: OLS, PS-matching, and the Orthogonal learner.[6]

[6] Using random forest.

Demonstration using simulated data

To examine whether we can get improvements using ML $\Rightarrow$ simulate "ground truth" data and try to recover true effect.

Basic setup

- Simulate data varying two dimensions:
  - Complexity: (1) *linear*, (2) *non-linear*
  - Size: (1) *small*, (2) *large*
- Compare: OLS, PS-matching, and the Orthogonal learner.[6]

[6] Using random forest.

Demonstration using simulated data

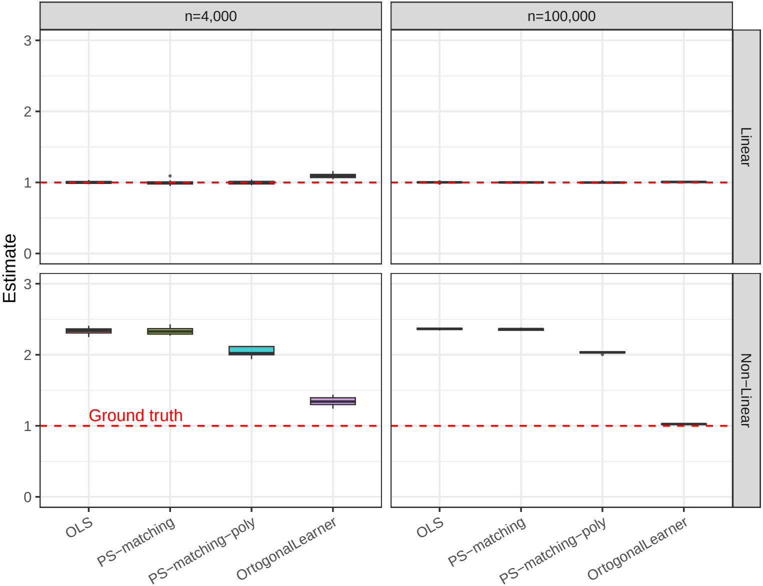
To examine whether we can get improvements using ML ⇒ simulate "ground truth" data and try to recover true effect.

Basic setup

- Simulate data varying two dimensions:
  - Complexity: (1) *linear*, (2) *non-linear*
  - Size: (1) *small*, (2) *large*
- Compare: OLS, PS-matching, and the Orthogonal learner.[6]
- Repeat 50 times, and examine closeness to ground truth.

[6] Using random forest.

Simulation results



Summary, orthogonal learners

- Like the T-learner:
  - Reframes causal inference as a prediction task.
  - Enables the use of supervised ML and account for non-linearities.
  - Can improve estimation of causal effects substantially.
  - Assumes all confounders accounted for.
- But instead of predicting potential outcomes — it predicts the treatment (X) and the outcome (Y) based on confounders (Z) in order to remove their influence.
- More robust than the T-learner[7].

[7] Uses all data; connects with classic analytical results in economics

Heterogenous effects

Heterogenous effects

Research frequently find that the effect of *events*, *exposures*, *policies* (etc) vary across individuals.

Heterogenous effects

Research frequently find that the effect of *events*, *exposures*, *policies* (etc) vary across individuals.

Examples:

- A *medical drug*'s effectiveness often vary by *health condition*.
- *Returns to college* often vary by *social background*.
- *Policy interventions* often have varying impacts across *demographics*.

Traditional approach

- Include interactions representing subgroups of interest:

$$Y = \beta_0 + \beta_1 X + \beta_2 Z + \beta_3 X \times Z$$

Traditional approach

- Include interactions representing subgroups of interest:

$$Y = \beta_0 + \beta_1 X + \beta_2 Z + \beta_3 X \times Z$$

- Selection of subgroups usually based on theory or convention.

Traditional approach

- Include interactions representing subgroups of interest:

$$Y = \beta_0 + \beta_1 X + \beta_2 Z + \beta_3 X \times Z$$

- Selection of subgroups usually based on theory or convention.

Limitations:

- Assumes we know moderators beforehand.
- We can do exploratory analysis—trying out interactions—but this (1) creates risk of p-hacking[8], and (2) is not feasible when data contain many variables.

[8]Unprincipled search for significance $\Rightarrow$ false positives.

Can we use ML to address these limitations?

Can we use ML to address these limitations?

- Any supervised learning method that discovers important, interpretable interactions?

Can we use ML to address these limitations?

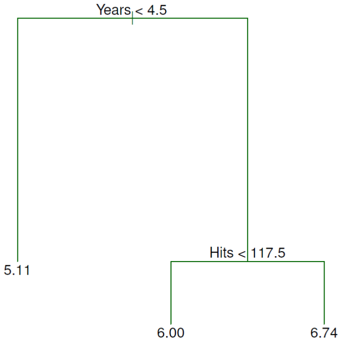
- Any supervised learning method that discovers important, interpretable interactions?
- Yes! Recall **Decision trees**.

Can we use ML to address these limitations?

- Any supervised learning method that discovers important, interpretable interactions?
- Yes! Recall **Decision trees**.
  - Learns *tree-structure* of *variable-combinations* important for *prediction* — using recursive binary splitting.

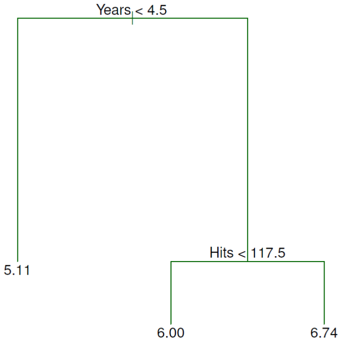
Can we use ML to address these limitations?

- Any supervised learning method that discovers important, interpretable interactions?
- Yes! Recall **Decision trees**.
    - Learns *tree-structure* of *variable-combinations* important for *prediction* — using recursive binary splitting.



Can we use ML to address these limitations?

- Any supervised learning method that discovers important, interpretable interactions?
- Yes! Recall **Decision trees**.
  - Learns *tree-structure* of *variable-combinations* important for *prediction* — using recursive binary splitting.



- But — not good at *inference* (also: leaves are about Y, not X).

Causal trees

From "decision trees" to "causal trees"

Clever idea (Athey et al): change the splitting criterion.

From "decision trees" to "causal trees"

Clever idea (Athey et al): change the splitting criterion.

- Standard tree: Splits to minimize *variance in outcomes* in leaves

From “decision trees” to “causal trees”

Clever idea (Athey et al): change the splitting criterion.

- Standard tree: Splits to minimize *variance in outcomes* in leaves
- Causal tree: Splits to maximize *difference in treatment effects* across leaves

From "decision trees" to "causal trees"

Clever idea (Athey et al): change the splitting criterion.

- Standard tree: Splits to minimize *variance in <u>outcomes</u>* in leaves
- Causal tree: Splits to maximize *difference in <u>treatment effects</u>* across leaves

Of course:

- We do not *observe* treatment effects in our data $\Rightarrow$ need to *estimate*.

From "decision trees" to "causal trees"

Clever idea (Athey et al): change the splitting criterion.

- Standard tree: Splits to minimize *variance in <u>outcomes</u>* in leaves
- Causal tree: Splits to maximize *difference in <u>treatment effects</u>* across leaves

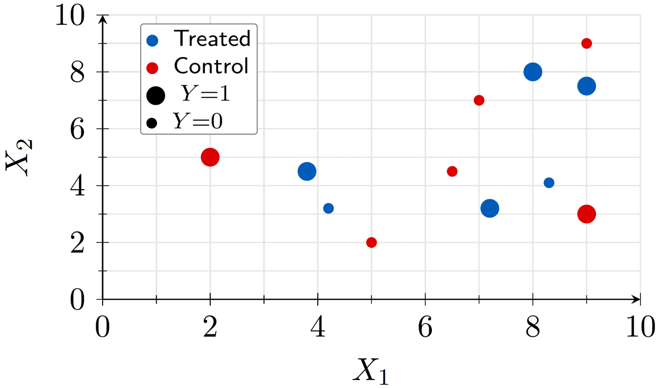
Of course:

- We do not *observe* treatment effects in our data ⇒ need to *estimate*.

So, more precisely, causal trees:

- Identify splits that maximize across-leaf variation in the within-leaf treated–untreated outcome difference.

Illustration: causal tree splitting



Suppose we have the following dataset:

- 12 observations
- 2 variables: X_1,X_2

Illustration: causal tree splitting

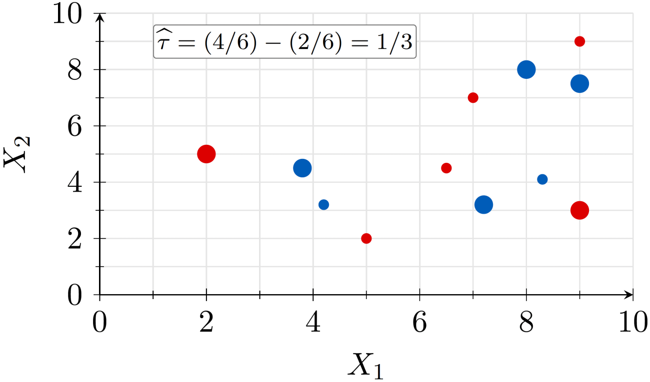
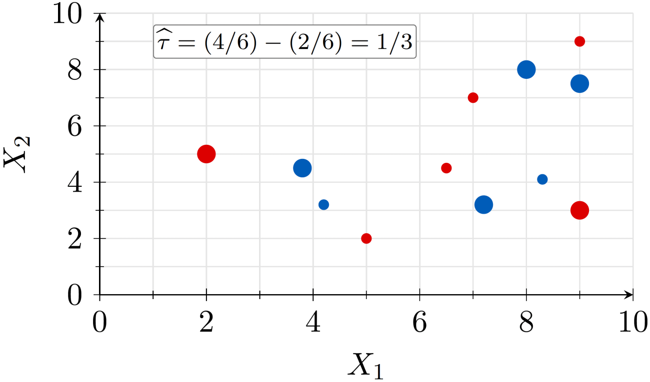
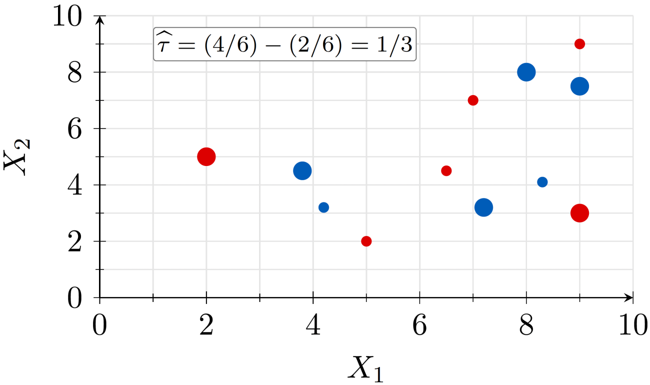


Illustration: causal tree splitting



- Average treatment effect: 1/3

Illustration: causal tree splitting



- Average treatment effect: $1/3$
- How can we split this data space (X_1, X_2) to maximize *difference treatment effect across regions*?

Illustration: causal tree splitting

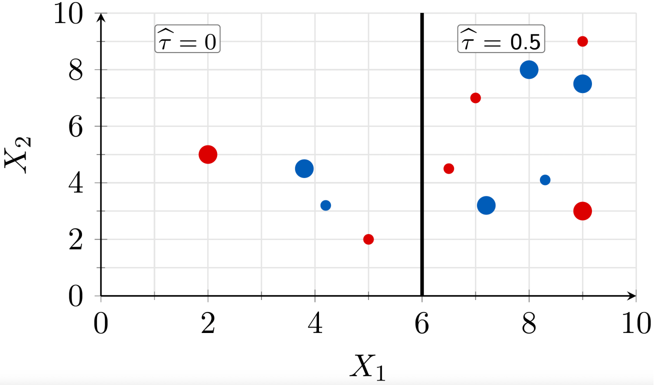
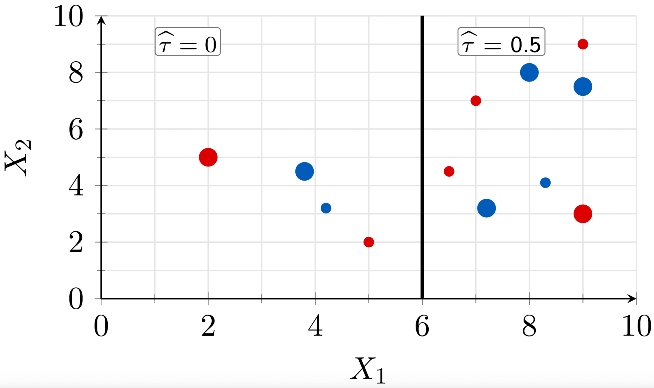


Illustration: causal tree splitting



- Splitting at $X_1 = 6$ divides up the data into a "no effect" and "moderate effect" region.

Illustration: causal tree splitting

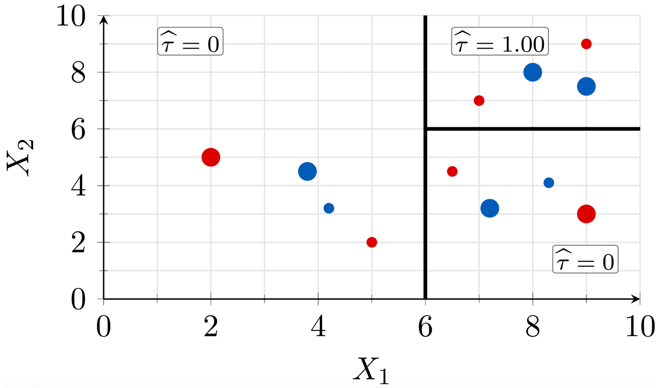
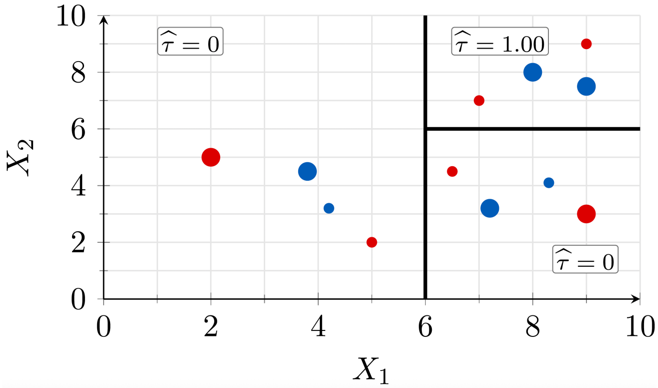


Illustration: causal tree splitting



- If we additionally split the right region ($X_1 > 6$) at $X_2 = 6$, we further refine the regions, getting an even stronger treatment effect in the top-right corner, and 0 in the rest.

What about "overfitting"?

- We want the tree to reflect robust causal patterns.

What about "overfitting"?

- We want the tree to reflect robust causal patterns.
- But—trees are known high-variance learners, tending to *overfit*.[9].

[9] The structure learned depending on which data point in sample it was trained on.

What about "overfitting"?

- We want the tree to reflect robust causal patterns.
- But—trees are known high-variance learners, tending to *overfit*.[9].

[9] The structure learned depending on which data point in sample it was trained on.

What about "overfitting"?

- We want the tree to reflect robust causal patterns.
- But—trees are known high-variance learners, tending to *overfit*.[9].

Causal trees use several strategies to mitigate overfitting:

[9] The structure learned depending on which data point in sample it was trained on.

What about "overfitting"?

- We want the tree to reflect robust causal patterns.
- But—trees are known high-variance learners, tending to *overfit*.[9].

Causal trees use several strategies to mitigate overfitting:

1. Minimum leaf size — e.g., 10 treated and 10 control.

[9] The structure learned depending on which data point in sample it was trained on.

What about "overfitting"?

- We want the tree to reflect robust causal patterns.
- But—trees are known high-variance learners, tending to *overfit*.[9].

Causal trees use several strategies to mitigate overfitting:

1. Minimum leaf size — e.g., 10 treated and 10 control.
2. Penalty for variance (within treatment categories) within leaves.[10]

[9] The structure learned depending on which data point in sample it was trained on.
[10] Which also is dependent on the *number of obs in the leaves*.

What about "overfitting"?

- We want the tree to reflect robust causal patterns.
- But—trees are known high-variance learners, tending to *overfit*.[9].

Causal trees use several strategies to mitigate overfitting:

1. Minimum leaf size — e.g., 10 treated and 10 control.
2. Penalty for variance (within treatment categories) within leaves.[10]

[9] The structure learned depending on which data point in sample it was trained on.
[10] Which also is dependent on the *number of obs in the leaves*.

What about "overfitting"?

- We want the tree to reflect robust causal patterns.
- But—trees are known high-variance learners, tending to *overfit*.[9].

Causal trees use several strategies to mitigate overfitting:

1. Minimum leaf size — e.g., 10 treated and 10 control.
2. Penalty for variance (within treatment categories) within leaves.[10]
3. 'Honest estimation'—learn tree on 50% of data; estimate on rest.[11]

[9] The structure learned depending on which data point in sample it was trained on.
[10] Which also is dependent on the *number of obs in the leaves*.
[11] Ensuring that any overfitting does not spillover to estimates.

What about “overfitting”?

- We want the tree to reflect robust causal patterns.
- But—trees are known high-variance learners, tending to *overfit*.[9].

Causal trees use several strategies to mitigate overfitting:

1. Minimum leaf size — e.g., 10 treated and 10 control.
2. Penalty for variance (within treatment categories) within leaves.[10]
3. 'Honest estimation'—learn tree on 50% of data; estimate on rest.[11]

[9] The structure learned depending on which data point in sample it was trained on.
[10] Which also is dependent on the *number of obs in the leaves*.
[11] Ensuring that any overfitting does not spillover to estimates.

What about "overfitting"?

- We want the tree to reflect robust causal patterns.
- But—trees are known high-variance learners, tending to *overfit*.[9].

Causal trees use several strategies to mitigate overfitting:

1. Minimum leaf size — e.g., 10 treated and 10 control.
2. Penalty for variance (within treatment categories) within leaves.[10]
3. 'Honest estimation'—learn tree on 50% of data; estimate on rest.[11]
4. Pruning — sufficient gain by split to justify increased tree size?

[9] The structure learned depending on which data point in sample it was trained on.
[10] Which also is dependent on the *number of obs in the leaves*.
[11] Ensuring that any overfitting does not spillover to estimates.

What about confounding?

- The *original causal tree* (Athey & Wager, 2016) assumes randomized treatments.

What about confounding?

- The *original causal tree* (Athey & Wager, 2016) assumes randomized treatments.
- Brand et al. (2021) proposed an *extension* using inverse probability weighting (IPW).

What about confounding?

- The *original causal tree* (Athey & Wager, 2016) assumes randomized treatments.
- Brand et al. (2021) proposed an *extension* using inverse probability weighting (IPW).
  - IPW: popular alternative to *propensity score matching*.

What about confounding?

- The *original causal tree* (Athey & Wager, 2016) assumes randomized treatments.
- Brand et al. (2021) proposed an *extension* using inverse probability weighting (IPW).
  - IPW: popular alternative to *propensity score matching*.
  - IPW: weights observations by their *propensity score* of being treated.

What about confounding?

- The *original causal tree* (Athey & Wager, 2016) assumes randomized treatments.
- Brand et al. (2021) proposed an *extension* using inverse probability weighting (IPW).
  - IPW: popular alternative to *propensity score matching*.
  - IPW: weights observations by their *propensity score* of being treated.
  - This "balances" the data across treated and untreated.

What about confounding?

- The *original causal tree* (Athey & Wager, 2016) assumes randomized treatments.
- Brand et al. (2021) proposed an *extension* using inverse probability weighting (IPW).
  - IPW: popular alternative to *propensity score matching*.
  - IPW: weights observations by their *propensity score* of being treated.
  - This "balances" the data across treated and untreated.
- Brand et al. (2021): check balance of confounders within leaves.

Variable selection

Brand et al. (2021):

Variable selection

Brand et al. (2021):

- Input variables should be pre-treatment.

Variable selection

Brand et al. (2021):

- Input variables should be pre-treatment.
- **Include** all vars used in estimation of the *propensity of treatment*
  - Variables presumed to cause the treatment, the outcome, or both.

Variable selection

Brand et al. (2021):

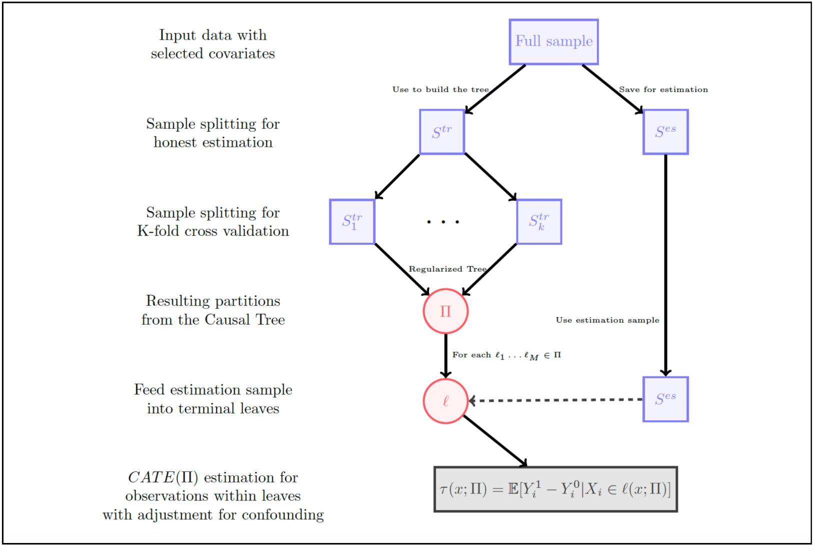
- Input variables should be pre-treatment.
- **Include** all vars used in estimation of the *propensity of treatment*
  - Variables presumed to cause the treatment, the outcome, or both.
- **Exclude** known instrumental variables.

Variable selection

Brand et al. (2021):

- Input variables should be pre-treatment.
- **Include** all vars used in estimation of the *propensity of treatment*
  - Variables presumed to cause the treatment, the outcome, or both.
- **Exclude** known instrumental variables.
- **Exclude** non-significant predictors of p(treatment).

Causal tree pipeline (Brand et al. 2021)



Empirical illustration (Brand et al. 2021)

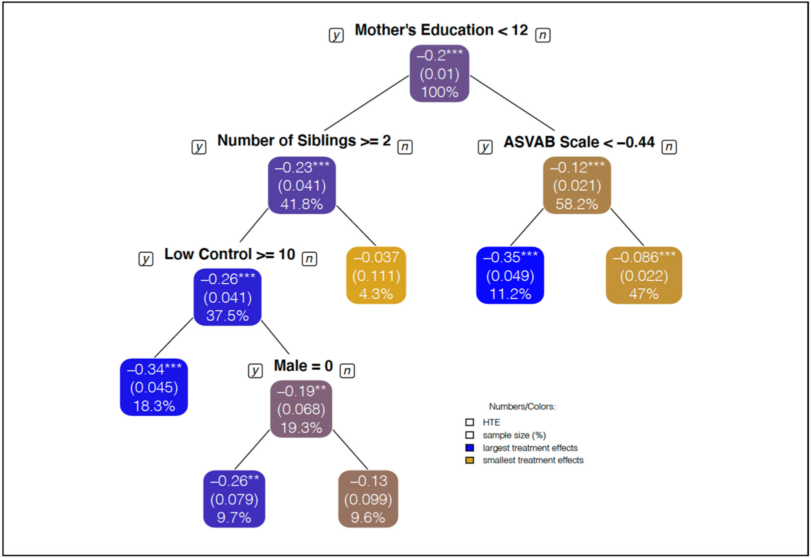
- **Data**: US longitudinal survey data (NLSY79)
- **Treatment**: college attendence
- **Outcome**: amount of low-wage work over a career.
- **Objective**: explore treatment effect heterogeneity
- **Method**: causal trees with IPW.

Empirical illustration (Brand et al. 2021)

Average treatment effect

Wage Outcome	Unadjusted	IPW
Proportion of time in low-wage work	–.223*** (.013)	–.189*** (.016)

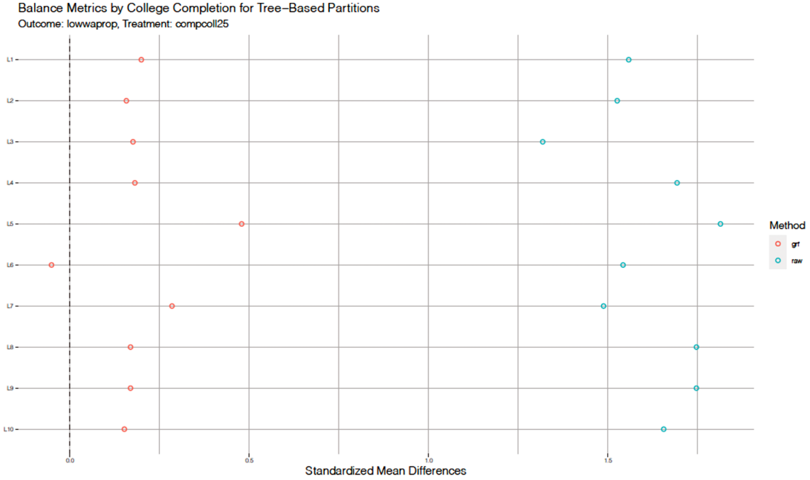
Empirical illustration (Brand et al. 2021)



Empirical illustration (Brand et al. 2021)

Balance in the different leaves:

- *Std mean difference* in *propensity of treatment* — by leaf.



Summary, causal trees

Pros

- Rigorous framework for discovering varying treatment effects[12]
- Highly interpretable tree-structure.<.->Consider leaves in terms of all variables, not just splits.
- Analytic guarantees for standard errors/CIs.

[12] For social science: potential identification of new, important moderators.

Summary, causal trees

Pros

- Rigorous framework for discovering varying treatment effects[12]
- Highly interpretable tree-structure.[13]
- Analytic guarantees for standard errors/CIs.

[12] For social science: potential identification of new, important moderators.
[13] Consider leaves in terms of all variables, not just splits.

Summary, causal trees

Pros

- Rigorous framework for discovering varying treatment effects[12]
- Highly interpretable tree-structure.[13]
- Analytic guarantees for standard errors/CIs.

[12] For social science: potential identification of new, important moderators.
[13] Consider leaves in terms of all variables, not just splits.

Summary, causal trees

Pros

- Rigorous framework for discovering varying treatment effects[12]
- Highly interpretable tree-structure.[13]
- Analytic guarantees for standard errors/CIs.

Cons

- Causal trees, like decision trees, are high-variance estimators.
- Tree-structure can be sensitive to the sample used to train it.

[12] For social science: potential identification of new, important moderators.
[13] Consider leaves in terms of all variables, not just splits.

Causal forest

Causal forest

Causal forest is the random-forest-equivalent of (causal) trees.

Causal forest

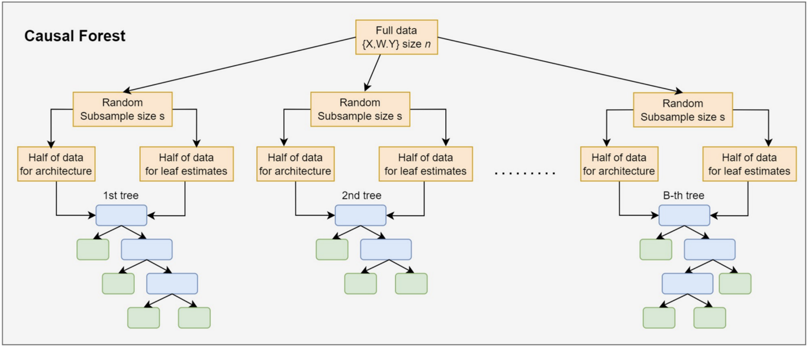
Causal forest is the random-forest-equivalent of (causal) trees.

- It learns a collection of (decorrelated) causal trees.

Causal forest

Causal forest is the random-forest-equivalent of (causal) trees.

- It learns a collection of (decorrelated) causal trees.



Two notable tweaks/additions

1. Incorporates orthogonal learning to remove confounding bias.

Two notable tweaks/additions

1. Incorporates orthogonal learning to remove confounding bias.
    - Before learning the trees, it estimates $f_Y(Z) \approx Y$ and $f_X(Z) \approx X$

Two notable tweaks/additions

1. Incorporates orthogonal learning to remove confounding bias.
   - Before learning the trees, it estimates $f_Y(Z) \approx Y$ and $f_X(Z) \approx X$
   - Subtract predictions to attain "residualized" $\bar{X}$, $\bar{Y}$

Two notable tweaks/additions

1. Incorporates orthogonal learning to remove confounding bias.
    - Before learning the trees, it estimates $f_Y(Z) \approx Y$ and $f_X(Z) \approx X$
    - Subtract predictions to attain "residualized" $\bar{X}$, $\bar{Y}$
    - Treatment effects calculated by regressing $\bar{X}$ on $\bar{Y}$ in leaves.

Two notable tweaks/additions

1. Incorporates orthogonal learning to remove confounding bias.
   - Before learning the trees, it estimates $f_Y(Z) \approx Y$ and $f_X(Z) \approx X$
   - Subtract predictions to attain "residualized" $\bar{X}$, $\bar{Y}$
   - Treatment effects calculated by regressing $\bar{X}$ on $\bar{Y}$ in leaves.
2. Uses forest-defined neighbors to estimate CATE

Two notable tweaks/additions

1. Incorporates orthogonal learning to remove confounding bias.
    - Before learning the trees, it estimates $f_Y(Z) \approx Y$ and $f_X(Z) \approx X$
    - Subtract predictions to attain "residualized" $\bar{X}$, $\bar{Y}$
    - Treatment effects calculated by regressing $\bar{X}$ on $\bar{Y}$ in leaves.
2. Uses forest-defined neighbors to estimate CATE
    - To estimate a CATE for a target profile z (e.g., age=20 & income>100K), we identify observations "close" to z in the forest.

Two notable tweaks/additions

1. Incorporates orthogonal learning to remove confounding bias.
   - Before learning the trees, it estimates $f_Y(Z) \approx Y$ and $f_X(Z) \approx X$
   - Subtract predictions to attain "residualized" $\bar{X}$, $\bar{Y}$
   - Treatment effects calculated by regressing $\bar{X}$ on $\bar{Y}$ in leaves.
2. Uses forest-defined neighbors to estimate CATE
   - To estimate a CATE for a target profile z (e.g., age=20 & income>100K), we identify observations "close" to z in the forest.
   - Closeness is captured by a weight $\alpha_i(z)$—how often observation i and z fall in the same leaf *across trees*.

Two notable tweaks/additions

1. Incorporates orthogonal learning to remove confounding bias.
    - Before learning the trees, it estimates $f_Y(Z) \approx Y$ and $f_X(Z) \approx X$
    - Subtract predictions to attain "residualized" $\bar{X}$, $\bar{Y}$
    - Treatment effects calculated by regressing $\bar{X}$ on $\bar{Y}$ in leaves.
2. Uses forest-defined neighbors to estimate CATE
    - To estimate a CATE for a target profile z (e.g., age$=$20 & income$>$100K), we identify observations "close" to z in the forest.
    - Closeness is captured by a weight $\alpha_i(z)$—how often observation i and z fall in the same leaf *across trees*.
    - The CATE for z is obtained by a *weighted* regression of the residualized outcome on the residualized treatment (weights $\alpha_i(z)$).

What can do with the output?

Two basic questions we can investigate are:[14]

- Which vars account for most heterogeneity? Variable importance.
- How do they moderate the treatment effect? CATE by subset.

[14] There is a lot more one can do; see reading list.

Demonstration via simulation

Setup

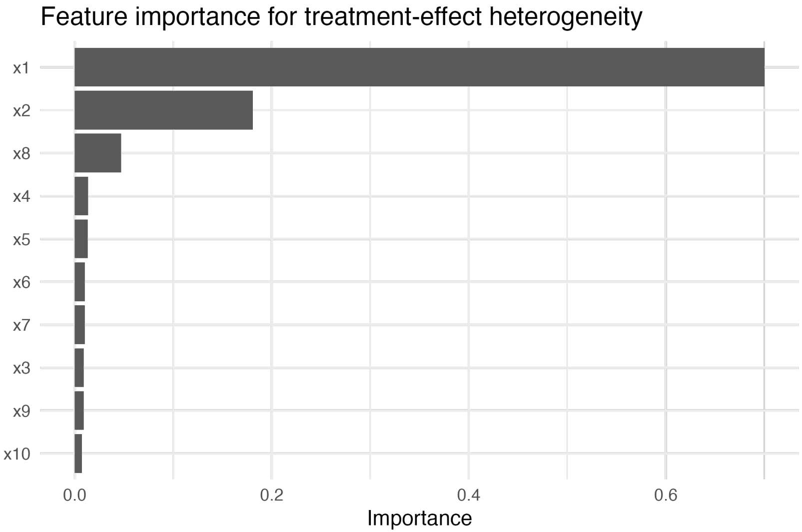
- Simulate data:
  - $n = 10,000$,
  - 10 features $X_1, \ldots, X_{10}$
  - Binary treatment W.
- True heterogeneity:

$$\tau(X) = 1.2 \sin(1.5\, X_1) + 0.8\, \mathbf{1}\{X_2{=}1\} + 0.2 X_8.$$

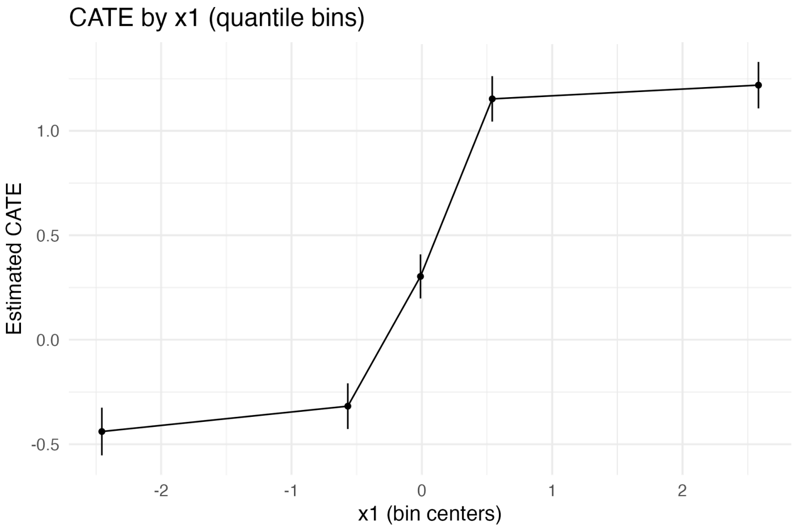
Objectives

- Identify key moderators.
- Quantify CATE by value ranges of top variables:

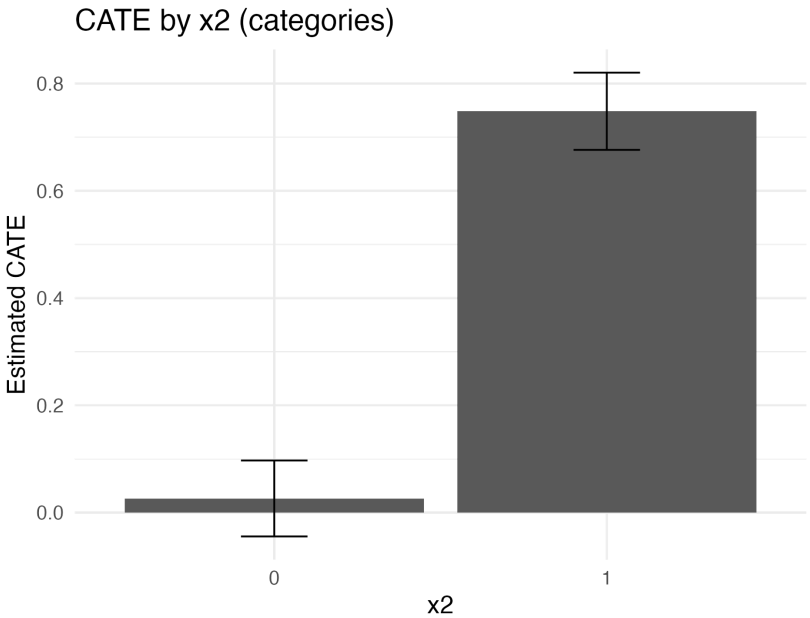
Variable importance



CATE for X_1


CATE by x1 (quantile bins)
Estimated CATE
1.0
0.5
0.0
-0.5
-2
-1
0
1
2
x1 (bin centers)

CATE for X_2



Summary

...

Further readings

Main:

- Brand, J. E., Xu, J., Koch, B., & Geraldo, P. (2021). Uncovering sociological effect heterogeneity using tree-based machine learning. Sociological Methodology, 51(2), 189-223.
- Chapter 22 in Alves, M. F., (2022) "Causal Inference for The Brave and True". https://matheusfacure.github.io/python-causality-handbook/22-Debiased-Orthogonal-Machine-Learning.html (until "CATE Estimation with Double-ML")
- Daoud, A., & Dubhashi, D. (2021). Melting together prediction and inference. Observational Studies, 7(1), 1-7.

Extra:

- Chapter 21 in Alves, M. F., (2022) "Causal Inference for The Brave and True". https://matheusfacure.github.io/python-causality-handbook/21-Meta-Learners.html
- Rehill, P. (2025). How Do Applied Researchers Use the Causal Forest? A Methodological Review. International Statistical Review.
- https://chenxing.space/blog/a-walkthrough-of-how-causal-forest-works/
- https://grf-labs.github.io/grf/articles/grf_guide.html