EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Causal Machine Learning

Supervised ML: predicting outcomes

Michael Knaus

WiSe 23/24

Plan for today

Introduce/recap supervised ML with a focus on tools that were driven by causal ML and will be important as we proceed

1. What we would like to have
2. OLS based prediction
3. Prediction based on shrinkage methods (OLS 2.0)
4. Botanical prediction
5. Why causal ML is needed

What we would like to have

Supervised ML

The supervised ML methods that we will consider in this course aim to predict

- an *outcome/response/signal/output/...* denoted by Y
- using *covariates/predictors/features/inputs/...* denoted by X

The resulting *predictions/fitted values* are denoted by $\hat{Y} = \hat{m}(x)$

The convenient feature of supervised ML is that we can compare observed outcomes with their predictions to assess prediction quality $(Y - \hat{Y})$

This is often referred to as "**The ground truth is known**" in the prediction setting

This narrative has intuitive appeal and we will revisit when and why it actually works

The unreachable goal of supervised ML

As we will do also later in the causal setting, let's first describe the target in unobservable population quantities

For predictions, we would like to know the conditional expectation function (CEF):

$$m(X) := \mathbb{E}[Y|X] \tag{1}$$

Recall from last lecture that the CEF minimizes the expected squared error:

$$m(X) = \underset{g(X)}{\arg\min}\, \mathbb{E}[(Y - g(X))^2] \tag{2}$$

and that

$$Y = m(X) + \varepsilon; \;\; \mathbb{E}[\varepsilon|X] = 0 \tag{3}$$

Bad news and good news (1/2)

Problem

The CEF is the best we can do BUT it is unobserved and needs to be estimated/learned using observable data.

Positive side

The Expected Squared Error (ESE) of an estimated CEF $\hat{m}(X)$ has a direct link to the squared CEF approximation error (derivation in two slides):

$$\mathbb{E}[(Y - \hat{m}(X))^2] = \underbrace{Var[\varepsilon]}_{\text{irreducible}} + \underbrace{\mathbb{E}[(m(X) - \hat{m}(X))^2]}_{\text{reducible}} - 2 \underbrace{Cov[\hat{m}(X), \varepsilon]}_{\text{= 0 if out-of-sample}} \tag{4}$$

Bad news and good news (2/2)

Equation (4) provides three important insights:

- 😿 The ESE does not become zero for the true CEF because $Var[\varepsilon] > 0$ in the usual case that the outcome is not a deterministic function of X
  $\Rightarrow$ We will never know if we found the true CEF
- 💡 If we want a representative picture for out-of-sample prediction, we need to test the model in an independent test set
- 🥳 We know that a function $\hat{m}(X)$ provides a better approximation of the CEF compared to alternative $\tilde{m}(X)$ if in an independent test set

$$\mathbb{E}[(Y - \hat{m}(X))^2] < \mathbb{E}[(Y - \tilde{m}(X))^2]$$

because it implies that

$$\mathbb{E}[(m(X) - \hat{m}(X))^2] < \mathbb{E}[(m(X) - \tilde{m}(X))^2]$$

Derivation expected squared error

We suppress (X)

$$
\begin{aligned}
ESE &= \mathbb{E}[(Y-\hat{m})^2] \\
&= \mathbb{E}[(m+\varepsilon-\hat{m})^2] \\
&= \mathbb{E}[m^2 + m\varepsilon - m\hat{m} + m\varepsilon + \varepsilon^2 - \hat{m}\varepsilon - m\hat{m} - \hat{m}\varepsilon + \hat{m}^2] \\
&= \mathbb{E}[m^2] + \mathbb{E}[m\varepsilon] - \mathbb{E}[m\hat{m}] + \mathbb{E}[m\varepsilon] + \mathbb{E}[\varepsilon^2] - \mathbb{E}[\hat{m}\varepsilon] - \mathbb{E}[m\hat{m}] - \mathbb{E}[\hat{m}\varepsilon] + \mathbb{E}[\hat{m}^2] \\
&= \mathbb{E}[m^2] + \mathbb{E}[m]\underbrace{\mathbb{E}[\varepsilon]}_{=0} - 2\,\mathbb{E}[m\hat{m}] + \mathbb{E}[m]\underbrace{\mathbb{E}[\varepsilon]}_{=0} + \mathbb{E}[\varepsilon^2] - 2\,\mathbb{E}[\hat{m}\varepsilon] + \mathbb{E}[\hat{m}^2] \\
&= \underbrace{\mathbb{E}[m^2] - 2\,\mathbb{E}[m\hat{m}] + \mathbb{E}[\hat{m}^2]}_{\mathbb{E}[(m-\hat{m})^2]} + \underbrace{\mathbb{E}[\varepsilon^2]}_{=Var[\varepsilon]} - 2 \underbrace{\mathbb{E}[\hat{m}\varepsilon]}_{=\mathbb{E}[\hat{m}]\,\mathbb{E}[\varepsilon]+Cov[\hat{m},\varepsilon]} \\
&= \mathbb{E}[(m-\hat{m})^2] + Var[\varepsilon] - 2Cov[\hat{m},\varepsilon]
\end{aligned}
$$

Practical value

The previous slides motivate the standard procedure to find the best predictor:

1. Randomly split your sample with N observations into a training set with N_{tr} and a test set with N_{te} observations
2. Estimate different CEF functions in the training sample $\hat{m}(X)$
3. Calculate and compare their out-of-sample Mean Squared Error (MSE) in the test sample:

$$MSE_{te} = \frac{1}{N_{te}} \sum_{i=1}^{N_{te}} (Y_i - \hat{m}(X_i))^2 \tag{5}$$

4. Pick the function with the lowest MSE

Why not in-sample MSE #Overfitting? (1/2)

Intuition by extreme example:

It is easy to fit a function that perfectly fits the outcomes in the training data

This is an extreme form of overfitting and means that $\hat{m}_o(X) = Y$ in the training sample and thus

$$MSE_{tr} = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} (Y_i - \hat{m}_o(X_i))^2 = \frac{1}{N_{tr}} \sum_{i=1}^{N_{tr}} (Y_i - Y_i)^2 = 0 \tag{6}$$

🤔 *But isn't it great that we found a perfect fit?*

☝ *No! We want to perfectly fit the CEF, not the observed outcomes.*

Why not in-sample MSE #Overfitting? (2/2)

Consider the sample analogue of the unobservable CEF approximation error in the training sample to see why fitting the outcome implies a bad fit of the CEF:

$$\frac{1}{N_{tr}}\sum_{i=1}^{N_{tr}}(m(X_i) - \hat{m}_o(X_i))^2 = \frac{1}{N_{tr}}\sum_{i=1}^{N_{tr}}(m(X_i) - Y_i)^2 \overset{(3)}{=} \frac{1}{N_{tr}}\sum_{i=1}^{N_{tr}}\varepsilon_i^2 > 0 \tag{7}$$

$\Rightarrow$ We do a bad job in approximating the CEF $\Rightarrow$ bad out-of-sample predictions

The discrepancy between equation (6) suggesting the perfect fit and equation (7) illustrates why **we should not use in-sample fit to judge prediction quality**

The same intuition holds also for estimators that overfit not that extreme

See also this Tweetorial as an excellent illustration/refresher of the related Bias-Variance trade-off

Summary

Supervised ML is so powerful because we can assess *unobservable* approximation quality of candidate prediction functions using *observed* outcomes

Overfitting can result in bad prediction performance (some musical nerd humor)

Independent test sample is required for reliable assessment of prediction quality

Successful predictors need to trade-off bias and variance

OLS based prediction

OLS as predictor

Standard OLS models the CEF as linear function

$$m^{ols}(X_i) = \beta_0 + \sum_{j=1}^{p} X_{ij}\beta_j \tag{8}$$

where the p covariates are handpicked (potentially including interactions and polynomials)

OLS estimates the parameters by solving the minimization problem

$$\hat{\beta}^{ols} = \underset{\beta}{\arg\min} \sum_{i=1}^{N} \left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j \right)^2 \tag{9}$$

$\Rightarrow \hat{Y} = x'\hat{\beta}^{ols}$ produces a fitted value for an observation with $X = x$

Why OLS is not the ultimate predictor

We learn that OLS is BLUE (Best Linear *Unbiased* Estimator) following the Gauss-Markov theorem BUT

- Unbiasedness is important for parameter estimation, but the price to pay in terms of variance might be too large for prediction #BiasVarianceTradeOff
- OLS tends to overfit
- OLS is not possible if $p + 1 > N$ and very instable for $(p + 1)/N$ close to one (this thinking is most productive for us, for bonus 🤯 #DoubleDecent)
- The CEF might not be linear and OLS provides ”only” the best linear prediction of CEF
- It is hard and (at least for me) unpleasant to manually specify an OLS model, i.e. to select the covariates and their functional form

Simulation notebook: Overfitting of OLS and value of training vs. test sample

Prediction based on shrinkage methods (OLS 2.0)

Lasso

OLS overfits and does not address the Bias-Variance trade-off

Shrinkage methods address this problem by penalizing the size of the coefficients

The most important method for this course is the Lasso (Tibshirani, 1996):

$$\hat{\beta}^{lasso} = \underset{\beta}{\arg\min} \sum_{i=1}^{N} \left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \tag{10}$$

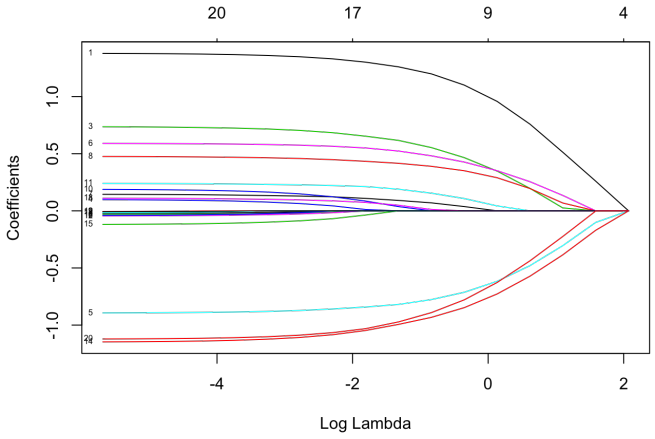
λ is the so-called tuning parameter or penalty term

$\lambda = 0$ is equivalent to OLS

$\lambda \to \infty$ results in an empty model including only a constant (β_0 is not penalized)

Selection property of Lasso

The Lasso is especially popular because it sets some coefficients to zero, which implies variable selection



Source: `glmnet` vignette

A note of caution

Selection property seems nice because it suggests that we can learn something about the ”important” variables

BUT recovering the structural model requires strong assumptions like limited correlation between covariates (see Hastie et al., 2015, Ch. 11.4)

However, predictors are often correlated, especially in social sciences

Example #SocioEconomicStatus

Education of mother and father are usually highly correlated. Most likely Lasso selects only one. This does not mean that the other is not relevant. Rather, Lasso leverages the ”omitted variable bias” in the coefficient of the selected variable to get good predictions with few predictors.

Mullainathan and Spiess (2017) forcefully illustrate this point ⇒ next slide

(Not) bar code graph of Mullainathan and Spiess (2017)

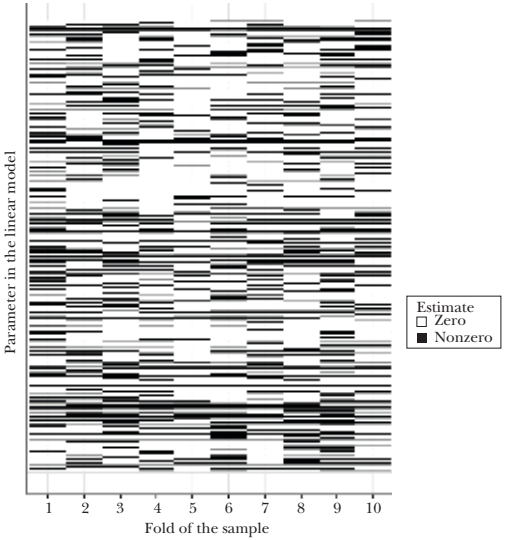


Figure 2

Selected Coefficients (Nonzero Estimates) across Ten LASSO Regressions

Note: We repeated the house-value prediction exercise on subsets of our sample from the American Housing Survey. First, we randomly cut the sample into ten partitions of approximately 5,000 units each. On each partition, we re-estimate the LASSO predictor, with LASSO regularization parameter fixed. The figure shows how the variables that are used vary from partition to partition. Each row represents one of x variables used. Each column represents a different partition. We color each cell black if that variable is used by the LASSO model (has a nonzero coefficient) in that partition. The figure documents a fundamental problem: a variable used in one partition may be unused in another. In fact, there are few stable patterns overall. For details, see discussion in text and online appendix available with this paper at http://e-jep.org.

Post-Lasso

Lasso has many extensions like adaptive/relaxed/fused/group/... Lasso

However, Post-Lasso is the most important for our purposes and works as follows

1. Run Lasso as in (10)
2. Identify the s variables with non-zero coefficients X^{sel}
3. Run an unpenalized OLS using only the selected variables:

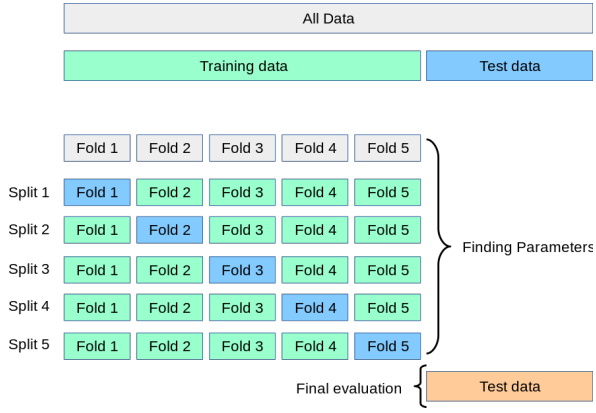
$$\hat{\beta}^{post-lasso} = \underset{\beta}{\arg\min} \sum_{i=1}^{N} \left(Y_i - \beta_0 - \sum_{j=1}^{s} X_{ij}^{sel} \beta_j \right)^2 \tag{11}$$

Caution

We still should not interpret $\hat{\beta}^{post-lasso}$ like we are used to, though it might be tempting. They have just predictive purpose, full stop. No structural meaning!

How to choose the tuning parameter? (1/2)

Cross-validation is the most popular technique:



Source: scikit-learn.org

How to choose the tuning parameter? (2/2)

Cross-validation (CV) estimates the out-of-sample MSE by simulating out-of-sample predictions

Usually we choose the penalty term that minimizes the cross-validated MSE

CV is generic and can be applied for any supervised ML that requires tuning

For Lasso and Post-Lasso Belloni et al. (2012) provide a data-driven alternative based on theoretical arguments (this is implemented in the `hdm` R package)

Simulation notebook: Lasso saves the job of OLS

Application notebook: Lasso

Botanical prediction

Global vs. local methods

OLS and its penalized descendants are global methods that aim to capture the whole functional form of the CEF to produce outcome predictions for a given x

Local methods take a different approach and only use observations in the neighbourhood of x for prediction

Different ways to define the neighbourhood:

- Classic approaches: nearest neighbour or kernel regressions
- Tree-based methods: regression trees, random forest, ...

The latter are a crucial building block for causal ML

Regression trees - concept

Partition the data based on X into M mutually exclusive regions $(R_1, ..., R_M)$

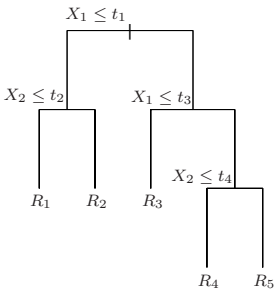
Predict the outcome for $X = x$ as the mean of outcomes falling into the same region/leaf (local constant):

$$\hat{m}^{tree}(x) = \sum_{m=1}^{M} \bar{Y}_m \mathbb{1}[x \in R_m] \tag{12}$$

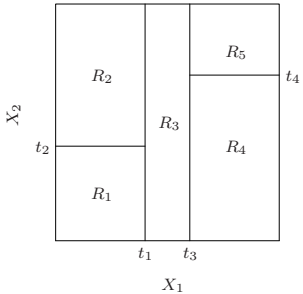
$$\text{where } \bar{Y}_m = \underbrace{\frac{1}{\sum_i \mathbb{1}[X_i \in R_m]}}_{n_m} \sum_i Y_i \mathbb{1}[X_i \in R_m] \tag{13}$$

Regression trees - illustration

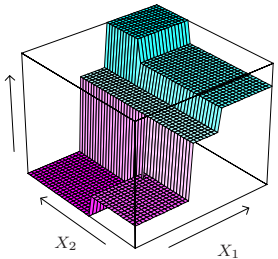
Illustration with two continuous covariates:



Tree



Partitioning



Estimated m(x)

Source: Hastie, Tibshirani & Friedman (2017) (p. 306)

How to find the splits that define the regions? - verbal

Find partitions that minimize the sum of squared error loss:

$$\sum_{i=1}^{N}(Y_i - \hat{m}^{tree}(X_i))^2 \tag{14}$$

Finding the optimal partitioning is often computationally too expensive

$\Rightarrow$ Greedy search of binary splits to minimize in-sample MSE:

- We start in the full sample (parent node) and find the MSE minimizing split and use it to create two daughter nodes
- We repeat this many times using the two daughter nodes as new parent nodes...

$\Rightarrow$ We grow a 🌲

Splitting criterion

Parent node can be split along variable j at split point s into a left leaf $L(j, s) = \{X \mid X_j \leq s\}$ and a right leaf $R(j, s) = \{X \mid X_j > s\}$

We seek j and s that minimize the sum of squared errors (SSE) $\Rightarrow$ min MSE:

$$\min_{j,s} \Bigg[\overbrace{\sum_{i:x_i \in L(j,s)} \overbrace{(y_i - \bar{y}_{L(j,s)})^2}^{\text{SSE in left partition}} + \sum_{i:x_i \in R(j,s)} \overbrace{(y_i - \bar{y}_{R(j,s)})^2}^{\text{SSE in right partition}}}^{\text{SSE for potential split } j,s} \Bigg] \tag{15}$$

Alternative splitting criterion

Note that an alternative but equivalent splitting criterion is to maximize the sum of squared predictions/the variance of predictions:

$$\min \sum_i (Y_i - \hat{m}^{tree}(X_i))^2 = \max \sum_i \hat{m}^{tree}(X_i)^2 = \max Var(\hat{m}^{tree}(X)) \tag{16}$$

This is irrelevant for the prediction case, but will be important for causal ML

The advantage will be that this splitting criterion does not require to observe the outcome, but only an estimate of the leaf mean

Note that we can write, while ignoring the *tree* superscript

$$\begin{aligned}\sum_i (Y_i - \hat{m}(X_i))^2 &= \sum_i (Y_i^2 - 2Y_i\hat{m}(X_i) + \hat{m}(X_i)^2) \\ &= \sum_i (Y_i^2 - 2Y_i\hat{m}(X_i) + \hat{m}(X_i)^2) + \underbrace{2\hat{m}(X_i)^2 - 2\hat{m}(X_i)^2}_{\text{principle}} \\ &= \sum_i (Y_i^2 + 2\hat{m}(X_i)(\hat{m}(X_i) - Y_i) + \hat{m}(X_i)^2 - 2\hat{m}(X_i)^2) \\ &= \sum_i (Y_i^2 + 2\hat{m}(X_i)(\hat{m}(X_i) - Y_i) - \hat{m}(X_i)^2)\end{aligned}$$

The middle term can be decomposed as

$$\sum_i 2\hat{m}(X_i)(\hat{m}(X_i) - Y_i) = 2\left[\sum_{i:X_i \in R_1} \bar{y}_1(\bar{y}_1 - Y_i) + \ldots + \sum_{i:X_i \in R_M} \bar{y}_M(\bar{y}_M - Y_i)\right]$$

The middle term is zero because for all its components we can write

$$\sum_{i:X_i \in R_m} \bar{y}_m(\bar{y}_m - Y_i) = \bar{y}_m \sum_{i:X_i \in R_m} (\bar{y}_m - Y_i) = \bar{y}_m \underbrace{\left[\sum_{i:X_i \in R_m} \bar{y}_m - \sum_{i:X_i \in R_m} Y_i \right]}_{=0} = 0$$

because $\sum_{i:X_i \in R_m} \bar{Y}_m = n_m \bar{Y}_m \overset{(13)}{=} n_m \frac{1}{n_m} \sum_{i:X_i \in R_m} Y_i = \sum_{i:X_i \in R_m} Y_i$

$$\Rightarrow \sum_i (Y_i - \hat{m}(X_i))^2 = \sum_i (Y_i^2 - \hat{m}(X_i)^2)$$

As the Y_i^2 part is irreducible, $\min \sum_i (Y_i^2 - \hat{m}(X_i)^2)$ is equivalent to $\max \sum_i \hat{m}(X_i)^2$

We can further show that this boils down to maximizing the variance of the predictions:

$$\text{Rewrite } \sum_i \hat{m}(X_i)^2 = \sum_{i:x_i \in R_1} \bar{y}_1^2 + \ldots + \sum_{i:x_i \in R_M} \bar{y}_M^2 = n_1 \bar{y}_1^2 + \ldots + n_M \bar{y}_M^2 = \sum_{m=1}^{M} n_m \bar{y}_m^2$$

Denote sample mean as $\bar{y} = \frac{1}{N}\sum_i y_i = \sum_m \frac{n_m}{N}\bar{y}_m$ and expand w/o changing max

$$\begin{aligned}
\frac{1}{N}\sum_m n_m \bar{y}_m^2 - 2\bar{y}\bar{y} + \bar{y}^2 &= \sum_m \frac{n_m}{N}\bar{y}_m^2 - 2\bar{y}\sum_m \frac{n_m}{N}\bar{y}_m + \bar{y}^2 \\
&= \sum_m \frac{n_m}{N}\bar{y}_m^2 - \sum_m \frac{n_m}{N} 2\bar{y}\bar{y}_m + \sum_m \frac{n_m}{N}\bar{y}^2 \\
&= \sum_m \frac{n_m}{N}(\bar{y}_m^2 - 2\bar{y}\bar{y}_m + \bar{y}^2) = \sum_m \frac{n_m}{N}(\bar{y}_m - \bar{y})^2 \\
&= \frac{1}{N}\sum_i \sum_m (\bar{y}_m - \bar{y})^2 \mathbb{1}[x_i \in R_m] = Var(\hat{m}(X_i))
\end{aligned}$$

$\Rightarrow$ $\min \sum_i (y_i - \hat{m}(X_i))^2$, $\max \sum_i \hat{m}(X_i)^2$, $\max Var(\hat{m}(X_i))$ are equivalent

In case of considering only a left and a right leaf we could also proceed differently on the previous slide (Bonus):

$$\sum_{m=1}^{M} n_m \bar{y}_m^2 = n_L \bar{y}_L^2 + n_R \bar{y}_R^2$$

Divide by N such that $\tilde{n}_L = n_L/N$ and $\tilde{n}_R = n_R/N$ and expand w/o changing max:

$$\begin{aligned}
\tilde{n}_L \bar{y}_L^2 + \tilde{n}_R \bar{y}_R^2 - \bar{y}^2 =& \tilde{n}_L \bar{y}_L^2 + \tilde{n}_R \bar{y}_R^2 - (\tilde{n}_L \bar{y}_L + \tilde{n}_R \bar{y}_R)^2 \\
=& \tilde{n}_L \bar{y}_L^2 + \tilde{n}_R \bar{y}_R^2 - \tilde{n}_L^2 \bar{y}_L^2 - 2\tilde{n}_L \bar{y}_L \tilde{n}_R \bar{y}_R - \tilde{n}_R^2 \bar{y}_R^2 \\
=& \tilde{n}_L \bar{y}_L^2 + \tilde{n}_R \bar{y}_R^2 - \tilde{n}_L(1 - \tilde{n}_R)\bar{y}_L^2 - 2\tilde{n}_L \bar{y}_L \tilde{n}_R \bar{y}_R - \tilde{n}_R(1 - \tilde{n}_L)\bar{y}_R^2 \\
=& \tilde{n}_L \bar{y}_L^2 + \tilde{n}_R \bar{y}_R^2 - \tilde{n}_L \bar{y}_L^2 + \tilde{n}_L \tilde{n}_R \bar{y}_L^2 - 2\tilde{n}_L \bar{y}_L \tilde{n}_R \bar{y}_R - \tilde{n}_R \bar{y}_R^2 + \tilde{n}_R \tilde{n}_L \bar{y}_R^2 \\
=& \tilde{n}_L \tilde{n}_R \bar{y}_L^2 - 2\tilde{n}_L \tilde{n}_R \bar{y}_L \bar{y}_R + \tilde{n}_R \tilde{n}_L \bar{y}_R^2 \\
=& \tilde{n}_L \tilde{n}_R (\bar{y}_L^2 - 2\bar{y}_L \bar{y}_R + \bar{y}_R^2) \\
=& \tilde{n}_L \tilde{n}_R (\bar{y}_L - \bar{y}_R)^2
\end{aligned}$$

In case of considering only a left and a right leaf we could also proceed differently on the previous slide (Bonus 2):

$$\begin{aligned}
\sum_{m=1}^{M} n_m \bar{y}_m^2 &= n_L \bar{y}_L^2 + n_R \bar{y}_R^2 \\
&= n_L \left(\frac{1}{n_L} \sum_{i:L} Y \right)^2 + n_R \left(\frac{1}{n_R} \sum_{i:R} Y \right)^2 \\
&= \frac{n_L}{n_L^2} \left(\sum_{i:L} Y \right)^2 + \frac{n_R}{n_R^2} \left(\sum_{i:R} Y \right)^2 \\
&= \frac{1}{n_L} \left(\sum_{i:L} Y \right)^2 + \frac{1}{n_R} \left(\sum_{i:R} Y \right)^2
\end{aligned}$$

This may help to understand equation 9 of Athey, Tibshirani and Wager (2019) discussed later

How to prevent overfitting?

If X is continuous, we could split until every observation has its own leaf

The prediction would thus be based only on one observation predicting itself $\Rightarrow$ perfect in-sample fit but bad approximation of CEF like discussed on slide 8

Again cross-validation can be used to find the optimal depth

To this end we grow deep trees and prune them back, i.e. cut-off some leaves

However, Athey and Imbens (2016) note that regression trees always overfit and propose a solution called honesty

Motivation for honesty

Conditional on finding a split, the estimated means in the leaves are too extreme

of a sample $\mathcal{S} \in \mathbb{S}$ constructs a partition. As a very simple example, suppose the feature space is $\mathbb{X} = \{L, R\}$. In this case there are two possible partitions, $\Pi_N = \{L, R\}$ (no split), or $\Pi_S = \{\{L\}, \{R\}\}$ (full split), and so the space of trees is $\mathbb{P} = \{\Pi_N, \Pi_S\} = \{\{L, R\}, \{\{L\}, \{R, \}\}\}$. Given a sample $\mathcal{S}$, the average outcomes in the two subsamples are $\overline{Y}_L$ and $\overline{Y}_R$. A simple example of an algorithm is one that splits if the difference in average outcomes exceeds a threshold c:

$$\pi(\mathcal{S}) = \begin{cases} \{\{L, R\}\} & \text{if } \overline{Y}_L - \overline{Y}_R \leq c, \\ \{\{L\}, \{R\}\} & \text{if } \overline{Y}_L - \overline{Y}_R > c. \end{cases}$$

The potential bias in leaf estimates from adaptive estimation can be seen in this simple example. While $\overline{Y}_L - \overline{Y}_R$ is in general an unbiased estimator for the difference in the population conditional means $\mu(L) - \mu(R)$, if we condition on finding that $\overline{Y}_L - \overline{Y}_R \geq c$ in a particular sample, we expect that $\overline{Y}_L - \overline{Y}_R$ is larger than the population analog.

Source: Athey and Imbens (2016)

Honest estimation

Athey and Imbens (2016) propose an ”honest” procedure:

1. Randomly split your sample in two halves
2. Use the first half to learn the tree structure
3. Use the second half to estimate the means in the leaves

This allows to use standard inference for the leaf means in the second half of the sample

Inference is then valid for this particular sample split

Random forest - idea

Regression trees are instable $\Rightarrow$ high variance $\Rightarrow$ high MSE

Small changes in the data can lead to completely different trees, especially with correlated predictors

Breiman (2001) proposes the Random Forest to reduce variability

Random Forest grows many trees on random subsamples using a random subset of predictors

Predictions are then formed as average over all tree predictions $\Rightarrow$ ensemble learning

Prediction uses outcomes from a data adaptive neighbourhood/kernel with closer outcomes receiving higher weights

Random forest - illustration

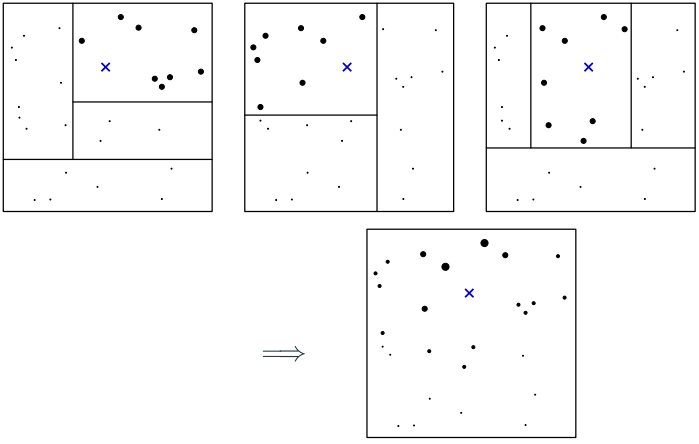


FIG. 1. *Illustration of the random forest weighting function. Each tree starts by giving equal (positive) weight to the training examples in the same leaf as our test point x of interest, and zero weight to all the other training examples. Then the forest averages all these tree-based weightings, and effectively measures how often each training example falls into the same leaf as x.*

Source: Athey, Tibshirani & Wager (2019)

Much more

Today we had a very selective look at the possibilities to approximate CEFs

See, e.g. Hastie, Tibshirani & Friedman (2017) for a comprehensive and extensive treatment

Tree-based methods like Random Forest or XGBoost are still very successful and outperform Deep Learning on ”tabular data” like the approximation of CEFs (see Grinsztajn et al., 2022)

The ”classic” relatively fast and easy to tune methods suffice for our purposes

$\Rightarrow$ No neural/deep nets required in this course

A computational expensive, but more flexible alternative is to use an ensemble, stacked, or ”super” learner that combines predictions of different methods in a data-driven way (see Naimi & Balzer, 2018 for a nice introduction)

Simulation notebook: Tree-based methods

Application notebook: Tree-based methods

Why causal ML is needed

Inference issues for average effect estimation

This variable selection thing seems useful for the selection of control variables, right?

Yes! BUT not naively. There are subtleties to be aware of and to be addressed:

- Post-model-selection estimators might lead to invalid statistical inference (Leeb & Pötscher, 2005, 2008)
- Single-equation approaches that use either an outcome or a treatment model to select control variables are biased (Belloni, Chernozhukov & Hansen, 2014a, 2014b)

$\Rightarrow$ We need estimators that constructively address these issues

No ground truth for effect prediction

Wouldn't it be great if we could predict causal effects instead of outcomes

Yes! BUT the causal effect of an individual is unobserved (fundamental problem of causal inference)

$\Rightarrow$ We can not use the standard machinery to predict effects instead of outcomes because there is no ground truth

We will consider different ways to circumvent this problem

Simulation notebook: Why naive model selection fails