ML for causal inference

Han Zhang

Outline

Review: three types of heterogeneous treatment effect

- Individual level treatment effect: $\rho_i = Y_i^1 - Y_i^0$
- Average treatment effects
  - ATE: $E(\rho_i)$
  - ATT: $E(\rho_i|D = 1)$
- Heterogeneous treatment effects
  - by covariate: CATE (the most popular approach)
  - by propensity score

Problems

- Use theory to guide you to add some interaction terms and obtain CATE
- This approach may miss important treatment effect heterogeneity
  - There may be heterogeneity for another variable, but you did not add its interaction into regression
  - Or, there are higher-order interactions
    - like it requires combinations of three covariates to have a causal effect
  - Or if the relationship cannot be captured by a linear model (nonlinear relationship)
    - If the relationship is U-shaped, then interaction term may not significant

Using ML for HTE

- Use some algorithms to predict counterfactual for each data point
- Then we can estimates individual-level treatment effect
  - So you can obtain effect by covariate, by propensity score, or by subgroups, or any other way you like
- In previous lectures, we know that regression/matching are implicitly doing the same thing: counterfactual predictions
  - But we know they are limited by the simple functional form and may has high estimator bias

Regression as Imputation

- If ignorability is true, the regression estimator of ATE is implicitly making counterfactual imputation using linear regression.
- Hence the similar form of ignorability and MCAR assumption

Unit	Y_i^0	Y_i^1	D_i	$X_{[1]i}$	$X_{[2]i}$
1	?	2	1	1	7
2	5	?	0	8	2
3	?	3	1	9	3
4	?	10	1	3	1
5	?	2	1	5	2
6	0	?	0	7	0

- Run a regression as $Y = \beta_0 + \beta_1 D_i + \beta_2 X_{[1]i} + \beta_3 X_{[2]i}$, and impute counterfactual outcome using the linear regression:

Unit	Y_i^0	Y_i^1	D_i	$X_{[1]i}$	$X_{[2]i}$
1	$\hat{\beta}_0 + \hat{\beta}_1 \cdot 0 + \hat{\beta}_2 \cdot 1 + \hat{\beta}_3 \cdot 7$	2	1	1	7
2	5	$\hat{\beta}_0 + \hat{\beta}_1 \cdot 1 + \hat{\beta}_2 \cdot 8 + \hat{\beta}_3 \cdot 2$	0	8	2
3	$\hat{\beta}_0 + \hat{\beta}_1 \cdot 0 + \hat{\beta}_2 \cdot 9 + \hat{\beta}_3 \cdot 3$	3	1	9	3
4	$\hat{\beta}_0 + \hat{\beta}_1 \cdot 0 + \hat{\beta}_2 \cdot 3 + \hat{\beta}_3 \cdot 1$	10	1	3	1
5	$\hat{\beta}_0 + \hat{\beta}_1 \cdot 0 + \hat{\beta}_2 \cdot 5 + \hat{\beta}_3 \cdot 2$	2	1	5	2

Matching estimator using Propensity Score

Unit	Y_i^0	Y_i^1	D_i	$X_{[1]i}$	$X_{[2]i}$	$p(D_i = 1 \mid X_i)$
1	?	2	1	1	7	0.33
2	5	?	0	8	2	0.14
3	?	3	1	10	3	0.73
4	?	10	1	3	1	0.35
5	?	2	1	5	2	0.78
6	0	?	0	7	0	0.70

(2)

Unit	Y_i^0	Y_i^1	D_i	$X_{[1]i}$	$X_{[2]i}$	$p(D_i = 1 \mid X_i)$
1	5	2	1	1	7	0.33
2	5	2	0	8	2	0.14
3	0	3	1	10	3	0.73
4	5	10	1	3	1	0.35
5	0	2	1	5	2	0.78
6	0	3	0	7	0	0.70

(3)

- Estimated ATE is $7/6$

Using ML for HTE

- ML algorithms can typically predict better than regresion/matching
  - Matching actually relies on a simplest ML algorithm: k-nearest-neighbor (KNN)
- Next, we talk about how to make a better prediction algorithm

Y-hat vs β-hat

- $Y = g(X)$ is a prediction function
- Two perspectives: y-hat vs. β-hat
- y-hat perspective: I care more about whether the final predictions are unbiased
- β-hat:
  - Make parametric assumptions about $f(X)$,
  - e.g., $g(X) = \alpha + \beta X$
  - Then the problem becomes whether my estimates of parameters $(\hat{\alpha}, \hat{\beta})$ are unbiased

Prediction

- So we have two kinds of unbiasedness:
  - unbiasedness for final outcomes Y
  - unbiasedness for parameters
    - when talking about some estimators (e.g., OLS, MLE) are unbiased, we are talking about this one: unbiasedness for parameters
- if my assumptions are correct, then unbiased estimates of parameters also mean unbiased estimates of Y
- But what if they are wrong? We know the linear assumptions are very likely to be wrong in real worlds

Quantify a good estimator

- Now we forgot about linear assumptions; just imagine we can have all sorts of possible functions $g(X)$
- Now we compare the mean squared error (MSE) between Y and its empirical prediction $\hat{g}(X)$, $E[(Y - \hat{g}(X))^2]$
  - Among all possible $g(X)$, $E(Y|X)$ is the best predictor of Y because it minimizes mean squared error $E[(Y - g(X)^2]$
  - error is $\epsilon = Y - E(Y|X)$

Review

Conditional Expectation as the Best Predictor.

$$
\begin{aligned}
E\Big[\big(Y - g(X)\big)^2\Big] &= E\Big[\big(\epsilon + E(Y|X) - g(X)\big)^2\Big] && (4)\\
&= E\Big[\epsilon^2 + 2\epsilon\big(E(Y|X) - g(X)\big) + \big(E(Y|X) - g(X)\big)^2\Big] && (5)\\
&= E\big[\epsilon^2\big] + 2E\Big[\epsilon\big(E(Y|X) - g(X)\big)\Big] + E\Big[\big(E(Y|X) - g(X)\big)^2\Big] && (6)\\
&= E\big[\epsilon^2\big] + E\Big[\big(E(Y|X) - g(X)\big)^2\Big] && (7)\\
&\geq E\big[\epsilon^2\big] && (8)\\
&= E\Big[\big(Y - E(Y|X)\big)^2\Big] && (9)
\end{aligned}
$$

□

Bias-variance decomposition

From Equation 7

Bias Variance Decomposition.

$$E\Big[\big(Y - g(X)\big)^2\Big] = E\big[\epsilon^2\big] + E\Big[\big(E(Y|X) - g(X)\big)^2\Big] \tag{10}$$

$$= \big(E(\epsilon^2) + V(\epsilon)\big) + E\Big[\big(E(Y|X) - g(X)\big)^2 \tag{11}$$

$$= V(\epsilon) + E(\text{bias}^2) \tag{12}$$

□

- MSE between truth ans prediction can be decomposed into:
- bias squared: this bias is the difference between best possible prediction, and whatever function you are using (e.g. linear regression or others)
- variance of error:
  - error is the difference between best possible prediction and the truth $\epsilon = Y - E(Y|X)$

Bias-variance decomposition for population

$$E\left[\left(Y - g(X)\right)^2\right] = V(\epsilon) + E(\text{bias}^2)$$

- How to reduce bias:
  - once you have determined to use certain prediction function, such as linear regression, then there is a fundamental difference between the function you choose, and the best possible one
  - Simple models such as linear regression often result in large bias
- How to reduce variance of error:
  - You have to find more predictive X in order to reduce this variance of error
  - This relates to your data only; the difference between best predictor and the truth
  - unrelated with your model

Bias Variance Decomposition

- Now we move to more complex situations, from $g(X)$ to its estimates, $\hat{g}(X)$)

$$E[(Y-\hat{g}(X))^2] = V\big[Y-E(Y|X)\big] + \big[\hat{g}(X)-E(Y|X)\big]^2 + V(\hat{g}(x))$$

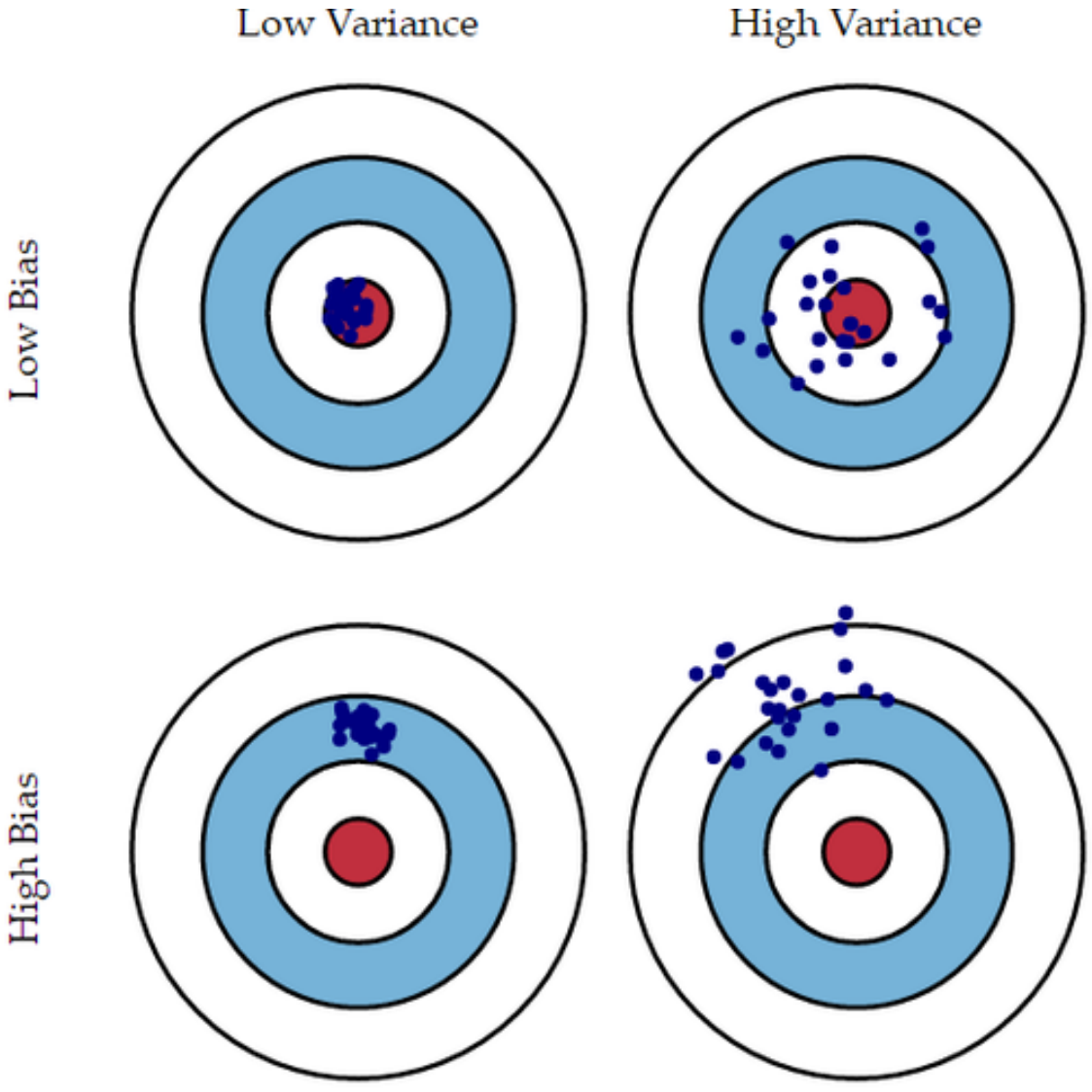
$=$ variance of error $+$ (estimator bias)$^2+$
estimator variance

- Variance of error: this only relates to your data;
  - They are irreducible as long as you have selected X; it will be large if X has nothing to do with Y.
- Estimator bias: relating to your model
  - OLS is often bad at approaching $E(Y|X)$
- Estimator variance: new, relating to your model
  - It roughly indicates how varied your predictions can be
  - Simple models typically have low prediction variance

Bias Variance Trade-off

- To reduce error: find more predictive X
- The other two quantities relate to your model (estimator):
  - Simple models (like OLS) have large estimator bias, but small estimator variance
  - Complex models have small estimator bias, but large estimator variance

Bias vs Variance (illustration)

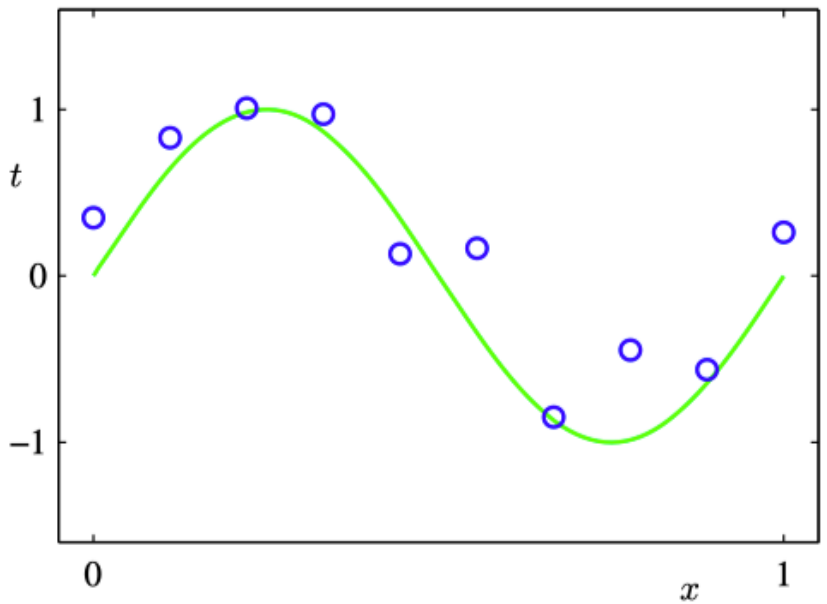


Bias Variance Trade-off (example)

- We have a linear regression with only one variable X, but we add higher order terms

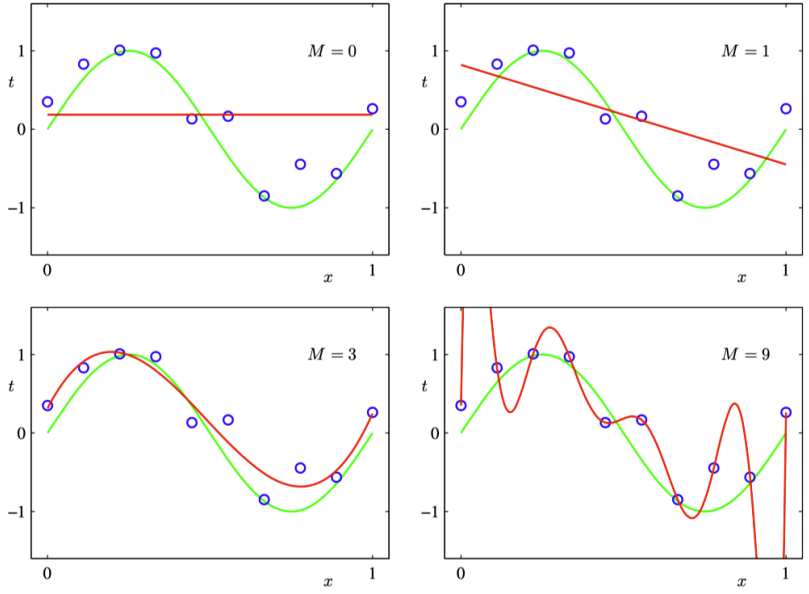
$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_M X^M$$

- True data generating process is a sin function (green)
- Randomly sample some data points from this curve (blue)
- Then we try different OLS models to fit the curve by adding more and more high-order terms



Bias Variance Trade-off (cont'd)

- $M = 1$, fits the data very bad (high predictor bias)
- $M = 9$, fits the data so well (small predictor bias), but it is highly sensitive to small changes in observations
  - The prediction on new data can be very bad
  - This is known as over-fitting
- $M = 3$, it achieves a good balance between predictor bias and variance



Bias Variance Trade-Off

- Simple model predicts the data very bad (high predictor bias)
- Complex model predicts the data too well (low predictor bias), but it has high estimation variance and is does not generalize well
- Ideal predictive models should balance the predictor bias and variances

Motivating example: strong interactions

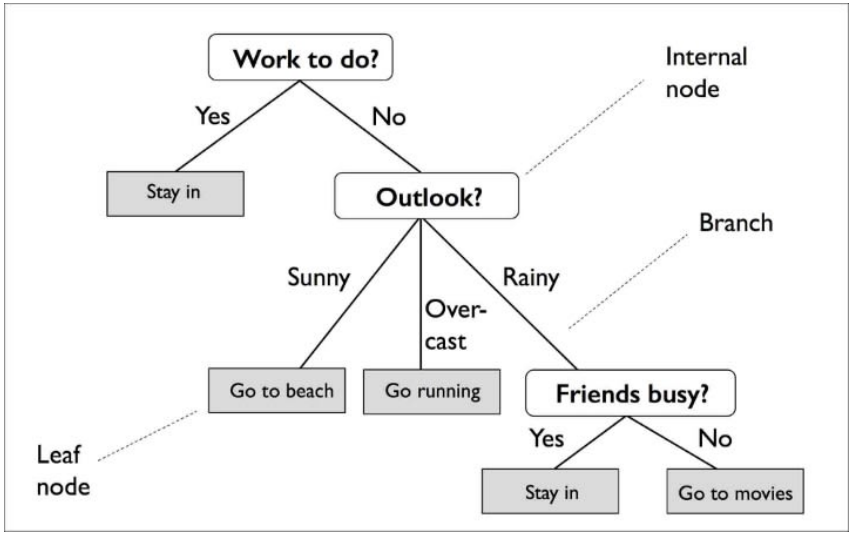
- Let us assume that we have a dataset with 1000 observations and 20 continuous predictors
- Say, each pair of the 20 predictors can have an interaction effect
- That means $20 + \binom{20}{2} = 190$ parameters
- What about we assume triple interactions between each triplet of predictors?
- $20 + \binom{20}{2} + \binom{20}{3} = 1350$ parameters
- OLS estimator cannot find a unique solution

Decision Tree

- Decision tree is a simple method to handle complex data with lots of interactions
  - classification tree: binary/categorical outcomes
  - regression tree: continuous outcomes
- A different name: CART:
  - Classification And Regression Tree

Decision Tree Example

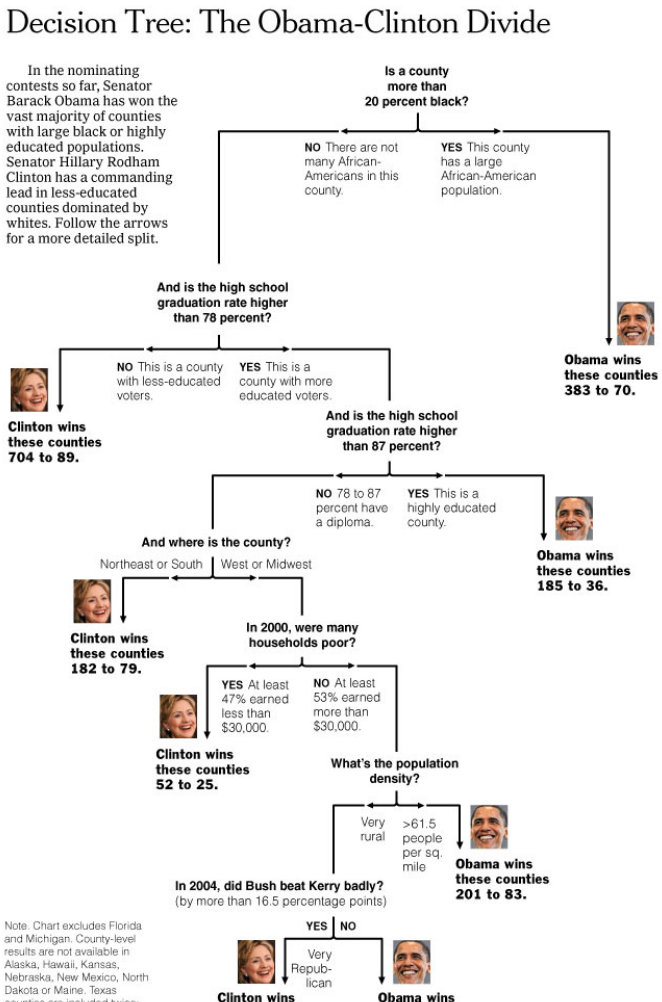
- Intuitively, decision tree visualizes one's sequential decisions process (Y), based on some predictors



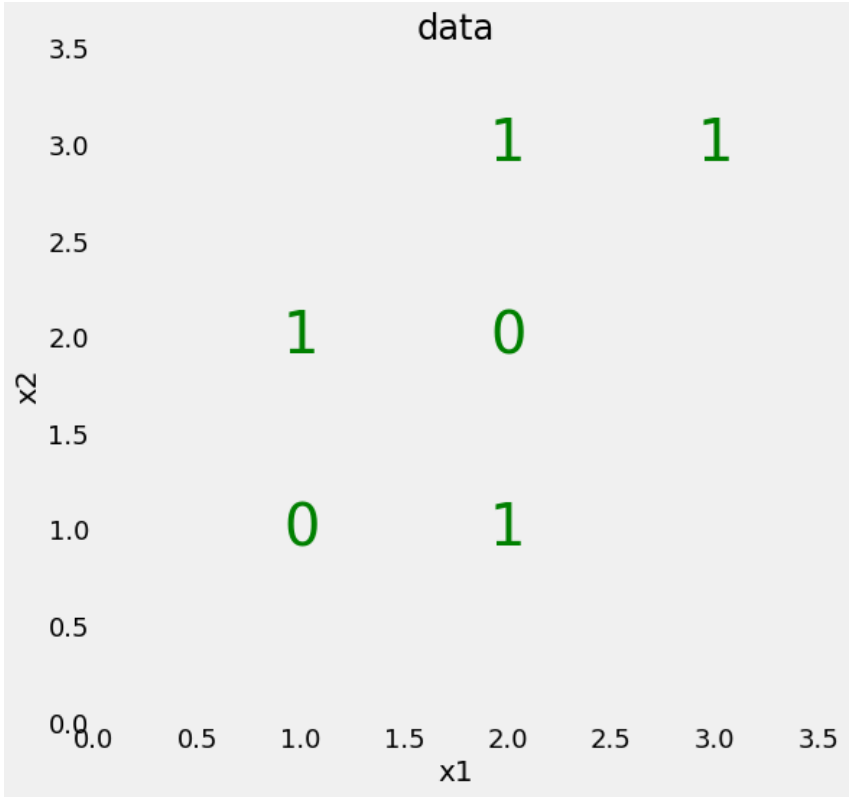
- Decisions Y are located at leaves
- The rightmost branch has a triple interaction

Decision Tree Example

https://archive.nytimes.com/www.nytimes.com/imagepages/2008/04/16/us/20080416_OBAMA_GRAPHIC.html?emc=polb1&nl=pol



Growing a Tree



- The above data cannot easily be separated by drawing a straight line (i.e., simplest linear regression)
- Let us draw a tree by ourselves to distinguish $Y = 0$ vs. $Y = 1$, based on X_1 and X_2
  - We want a binary tree: split into two branches

Some principles

- If you want to teach computers to draw a tree, here are some principles people usually follow:
- There are usually multiple ways to grow a tree. How do we pick the order we draw branches?
  - We should select a variable that best split data into two groups
  - In other words, it predicts the outcomes the best
- Machine is silly; we force it to be binary tree: each time split into two branches.
  - This means that for categorical and continuous X, we force them to make binary splits
- What if we there are multiple outcomes on a same leaf?
  - For continuous outcomes, the prediction is the mean
  - For categorical outcomes, the prediction is the mode
- One predictor can be used multiple times

Desision Tree Algorithms

- Decision Tree Algorithms help you to draw a tree from more complex data
- What are the steps we should take
- Let us first work with continuous outcome Y: regression tree and J continuous predictors X_1 to X_J
- There are two questions to consider:
  - Which X_j to choose first?
  - We will split X_j into $X_j < s$ and $X_j \geq s$. How do we choose s?
- And the intuitive answer is that:
  - You choose choose X_j and s that best separates Y (thus predicts Y the best)

Decision Tree Algorithms: formal math

- If we write this intuition down mathematically:
- We have p predictors: $X_1, \cdots, X_p$
- For each predictor X_j , calculate its minimum MSE:
  - Consider all it possible cutoffs s. A particular cutoff s will split the data into two regions:

$$R_1(j,s) = \{X|X_j < s\} \text{ and } R_2(j,s) = \{X|X_j \geq s\} \tag{13}$$

  - We should select a s that minimizes the MSE

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{14}$$

  - $\hat{y}_{R_1}$ is the mean response for the training observations in region $R_1(j,s)$
- Finally, select X_j and its s that yields the smallest MSE

Some details

- The above procedure tells you how to make the first split
- We repeat this procedure to grow sub-trees for each branch the procedure is the same, expect that you only use observations that belong to this branch to calculate MSE
  - This is known as *greedy recursive binary splitting*
  - It is possible that a particular choice of X_j may not be the current best, but may turn out to be the best choice in the future (locally best vs. globally best)
  - We do not consider the above possibility; just *greedily* choose the best partition and do not look back
- Decision Tree is a non-parametric model
  - Intuitively, whenever we grow a new branch, we are adding more interactions, using regression analogy.
  - But we do not write down parameters explicitly
  - MLE is not applicable

Classification Tree

- Very similar to regression trees.
- With one exception: we use cross-entropy loss instead of MSE
  - Also called log-loss or entropy loss
- For binary classification:

$$-\sum_{i=1}^{N} y_i \cdot \log P(\hat{y}_i = 1) + (1 - y_i) \cdot \log\left(1 - P(\hat{y}_i = 1)\right)$$

Pros and Cons of Decision Trees

- Pros:
  - easy to understand with visualization
  - easy to recognize which feature is more important (top of a tree)
  - handles complex interactive data with ease
  - handles categorical variables easily: no need for dummies
- Cons:
  - If your data are not that complex, tree easily overfit
  - It's not a parametric model: cannot summarize with several parameters (like linear regression do)
  - Confidence interval?
    - There are no explicit parameters, so surely no confidence intervals for parameters
    - The first analytical results is by Wager, Hastie and Efron (2014); JMLR

From One Tree to Many Trees

- Bagging tree (or ensemble of trees): averaging the predictions of many trees
  1. From the original training data, draw a sample with replacement of equal size
  2. Fit a tree for each sample
  3. Repeate 1 and 2 for some times
- Take the mean of estimates of each tree to produce a single estimate for each test data point

Random Forest

- Random Forests further extend the idea of bagging
- The key innovation of random forests:
- For each sample from the original training data, randomly select m variables (not using all p variables), and grow a tree;
  - A common choice: $m = \sqrt{p}$
- In other words, we just force $p - m$ predictors to be non-relevant each time
- Why? avoid over fitting

Boosting trees

- Bagging tree and Random Forest create many trees and average them together
  - Each of the tree is independent of the others
- A different idea is to create a sequence of trees that gradually improve over each other

Boosting trees

- Assume you first fit a decision tree $G_1(X)$
- Bagging trees and random forests: fit another decision $G_2(X)$, totally independent of G_1
  - Then final prediction $Y = \frac{G_1(X)+G_2(X)}{2}$
- Boosting trees: find $G_2(X)$ based on prediction error of the first tree
  1. Learn a second tree $G_2(X)$ to predict $Y - G_1(X)$
  2. Then final prediction $Y = G_1(X) + G_2(X)$
  3. Repeat Step 1 and 2: learn a new tree $G_3(X) = Y - G_1(X) - G_2(X)$, and so on.
- Essentially, boosting trees find data points that previous algorithms are most likely to be wrong, and improve the algorithm on these points.

Boosting trees vs Random Forests

- You man hear may different variants of trees
  - AdaBoost is the first and Gradient Boosting Tree is the most successful
- Gradient Boosting Tree (GBT) and Random Forests (RF) are typically the two best prediction methods you can get
  - GBT typically works well then the dimension is not that high
  - RF works well when the dimension is very high

Decision Tree: Bias vs Variance

- The tree grown using the above procedure can be quite complex
- We can always make a very complex tree by:
  - Try your best to make every single leaf contains only one Y
- Bias-variance trade-off:
  - Complex trees fit the data nearly perfect (low predictor MSE)
  - But has high predictor variance
    - Each time you have a new observation, the tree may look entire different
- We need to regularize to make tree simpler
  - regularize = avoid overfitting
  - explicitly make a very complex model less complex

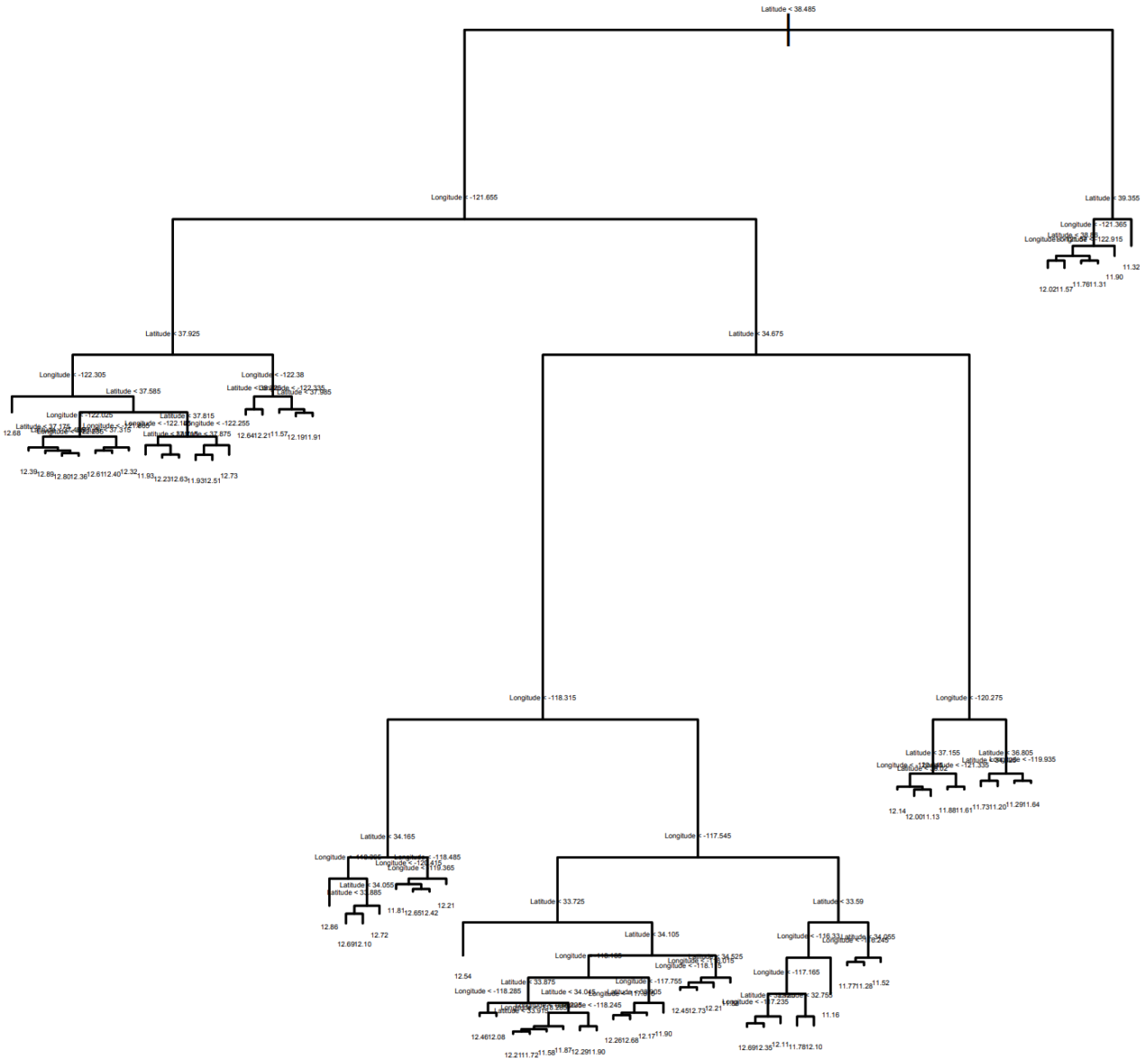
Regularization methods

- There are model-specific and model-agnostic ways to perform regularization
- Model-specific: tied to a specific model
  - For tree, it's called pruning
  - For regression, it's LASSO
  - For deep learning, it's max pooling
- Model-agnostic: train/test split

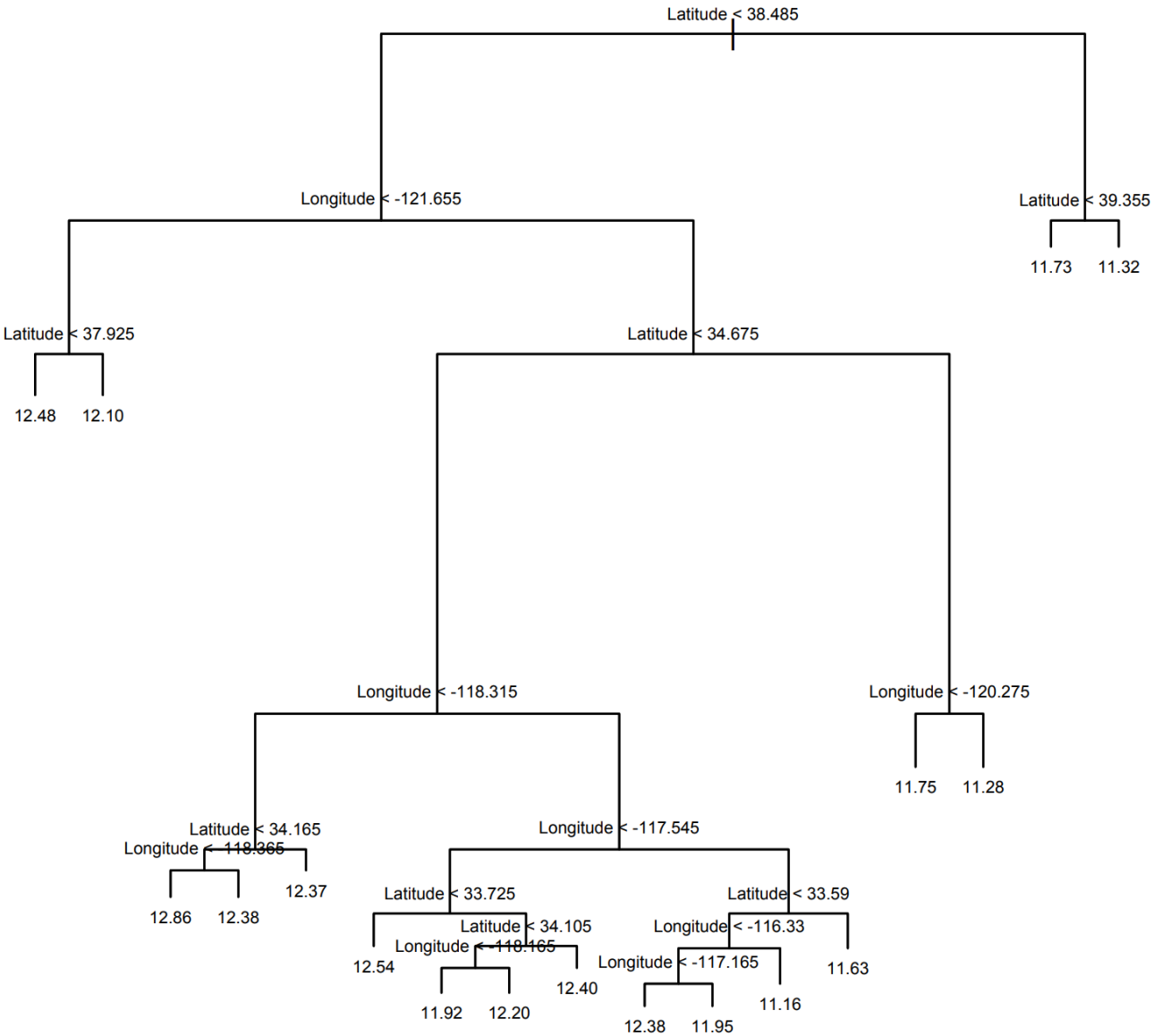
Decision tree pruning

- In decision trees, people often call regularization as pruning
- Intuitively, we are pruning leaves that are too small
  - So that in each leaf, there may be multiple observations and thus prediction errors for the training data
  - But it improves generalization performance
- Many different ways:
  - For the leaf node to have at least k observations
  - Force the tree to have no more than k leaves
  - Force the tree to have no more than k - level deep

Decision Tree: un-pruned



Decision Tree: pruned



Tuning parameters

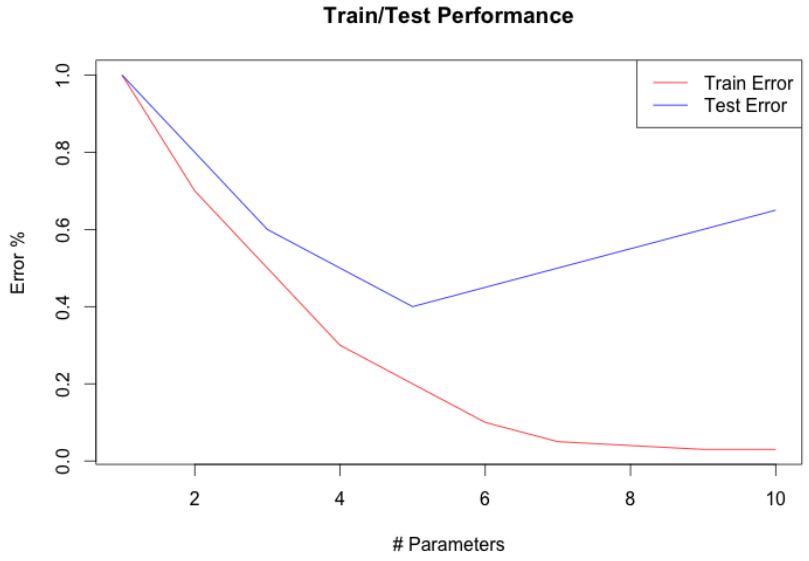
- Even pruning is not enough
- For instance, should we let the tree to be 5-level deep, or 6-level?
- How can we distinguish between setting the min number of observation for leaf node to be 10 versus 15?
- These parameters, whose values need to be chosen, are called tuning parameters

Training and Test data

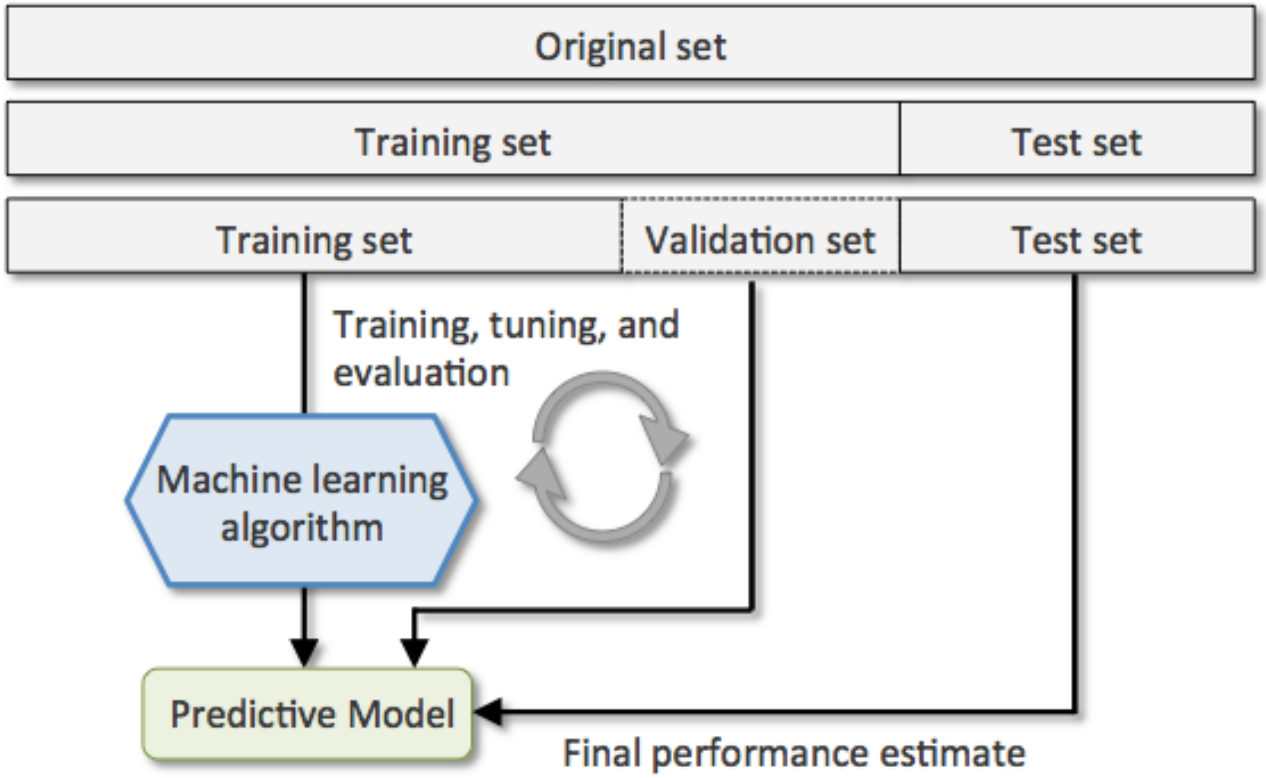
- Split data into two sets: training and test data (say, 80% vs. 20%)
- We only use training data to fit the model
- And we test our predictions on the test data
- Our best model should minimize the MSE on test data (also called test error)

Train/test split

- Test MSE is usually larger than training MSE
- When test error begins to increase, overfitting occurs
  - the predictor variances begin to increase
- Error/performance metrics based on test data gives more faithful evaluation of how the algorithm will perform in real-world, unseen new data



Train/validation/test split

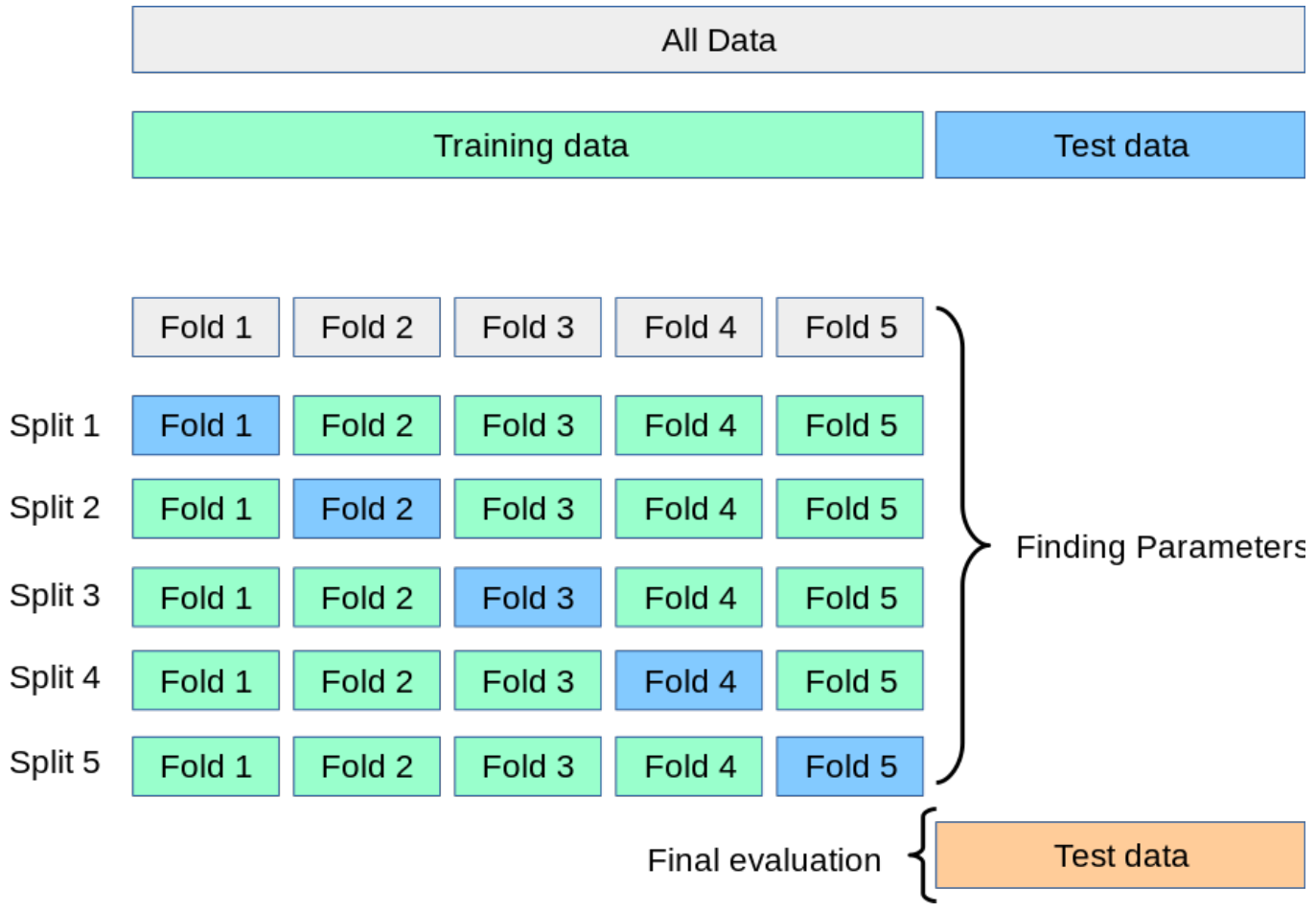


Cross-validation

- Imagine you have chosen a particular three-way split:
  - 70% as training
  - 10% as validation
  - 20% as test
- Sometimes the concern is that you the particular 70% may be different from the entire sample
- A more complex and popular approach is to use K-fold cross validation

Cross-validation

- K-fold cross validation (below example shows $K = 5$)



Select Tuning Parameters: cross-validation

- K-fold Cross-Validation:
  1. Divide your data into K subsets (i.e., folds) of roughly the same size (K is often 5 or 10)
  2. Select one subset k as the test data, and use the rest $K-1$ subsets as the training data
  3. Calculate the prediction error and save it
  4. repeat and take the average for k prediction errors
- For each possible value of your tuning parameters (e.g., how many observations a leaf node must have), we calculate its average prediction error over K folds
- Then select the choice that made the prediction error the smallest

Causal forests

- Causal forests: a series of articles such as Athey and Imbens, 2016, PNAS; Wager and Athey, 2018, JASA; Athey, Tibshirani, and Wager, 2019, Ann. Stat.
- Setup: we already know treatment D and outcome Y; we know many ways to estimates ATT
- By which factor do causal effect vary the most?
  - random forest finds a split to best predict Y
  - Causal tree finds a split (variable) to maximize the differences in treatment effects
    - On each leaf, observations are a mix of treated and control units
    - And the estimated causal effect for each leaf is the differences in means of treated and control units

Causal forest

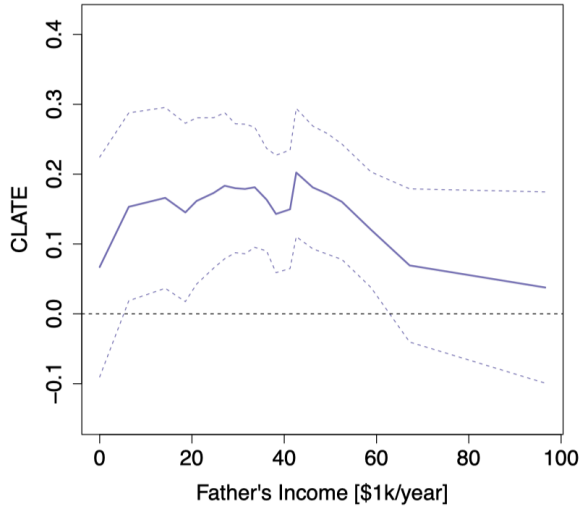
- To alleviate the concern of overfitting
- Athey et al. extends the three-way split idea to be what they called honest and subsampling
- Each time, 50% is hold out (denote this as C)
- The rest 50% is further split into two:
  - Use 25% to grow a tree;
  - Use the rest 25% to calculate treatment effects
- Then for each observation in the C holdout set, just assign its causal effect to the branch whose covariates would place it into
- The above procedure can be repeated many times; from causal tree to causal forests

Causal forest

- Athey, Tibshirani, and Wager, 2019, Ann. Stat.
- Questions: child penalty? more children decrease mother's tendency to work?
- Treatment is whether the mother had 3 or more children at census time
- Outcome: whether the mother did not work in the year preceding the census
- HTE question: does treatment effect varies by other covariates?
  - e.g., by fathers' income?

Causal forest

- Regression approach: just interact father's income with treatment
- Causal forest: predict individual treatment effect τ_i; then plot τ_i against father's income
- Found an inverse-U shape HTE; not easy to obtain if you just interact father's income with treatment status



Causal forest

- Jonathan M. V. Davis and Sara B. Heller, *Using Causal Forests to Predict Treatment Heterogeneity: An Application to Summer Jobs*, American Economic Review **107** (2017), no. 5, 546–550
- Jonathan M.V. Davis and Sara B. Heller, *Rethinking the Benefits of Youth Employment Programs: The Heterogeneous Effects of Summer Jobs*, The Review of Economics and Statistics **102** (2020), no. 4, 664–677
- Outcome:
  - violent-crime arrest
  - post-program employments
  - school persistence

Interaction based models

- Literature suggest that people who would benefit will be disconnected youth
  - Also most policy were designed to provide job search training for these group
- Hence, they suspect that treatment effect may vary by:
  - gender
  - whether someone had arrest history
  - had been enrolling in schools
- Run a treatment-covariate interaction model
- Cannot find consistant patterns; mostly these HTE are not statistically significant

Causal forest based estimations

- They identified some heterogeneity, but not in the pre-specified variables they believed
- Younger, Hispanic, Female, Less criminally involved have larger effect
- These are not the group predicted by the literature because they are not disconnected youth

TABLE 5.—SUMMARY STATISTICS BY QUARTILE OF PREDICTED EMPLOYMENT IMPACT

	Quartile of Predicted Employment Impact			
Variable	Q1	Q2	Q3	Q4
Prediction, any postprogram employment	−0.03	0.00	0.02	0.05
Take-up rate	0.39	0.41	0.43	0.46
In 2012 cohort	0.20	0.22	0.25	0.38
Age at program start	18.66	18.48	18.00	16.89
Hispanic	0.01	0.02	0.06	0.16
Male	0.89	0.85	0.84	0.77
Any baseline arrest	0.59	0.54	0.51	0.31
Graduated preprogram	0.35	0.30	0.24	0.10
Engaged in CPS in June	0.45	0.48	0.64	0.85
Days attended in prior school year (if any)	109.69	119.51	122.50	138.50
GPA	1.97	2.11	2.06	2.17
Worked in prior year	0.19	0.22	0.20	0.09
Census tract unemployment rate	17.32	14.72	12.89	12.26

Causal forests for observation data

- The default causal forest works with experiment data
- If you want to use that for observational data
  - Has to add ignorability assumption
  - And then weight observations by their treatment propensity
  - Called propensity trees
- There is a package called `grf` to achieve the above

Summary

- We have discussed one promising approach of using ML (tree and forests) to estimate causal effect
- There are many other directions of using ML for causal inference
- Ideas are similar: predict counterfactual