

Introduction to Trees & Random Forests

Jacob M. Schauer
Northwestern University

March 2020
University of Texas

Flipped Classroom

Before class:

- ▶ Students are assigned reading
 - ▶ Chapter 8 in *Introduction to Statistical Learning With Applications in R*
- ▶ Students watch online videos.

During class:

- ▶ Short lecture
- ▶ Lab
 - ▶ Guided example
 - ▶ Exercises (to be submitted for a grade)

Class Materials

Files at

https://github.com/j3schaue/intro_to_random_forests.git

Prior Knowledge

Regression:

- ▶ Linear and regularized models
- ▶ Generalized additive models (GAM)

Classification:

- ▶ Logistic and regularized models
- ▶ Discriminant analyses

Tuning & cross validation

CART & Extensions

- ▶ CART = Classification and Regression Trees
- ▶ Iterative algorithm that recursively partitions the data
- ▶ While trees are not always particularly great on their own, extensions that are based on trees usually are.
 - ▶ Random forests
 - ▶ Boosted models
- ▶ CART extensions are becoming popular
 - ▶ Predictive accuracy (He & Hahn, 2020)
 - ▶ Causal inference (Hill, 2007; Yeager et al., 2019)

Review: Regression

$$Y = f(\mathbf{X}) + \epsilon$$

- ▶ We want to find \hat{f} that gives good predictions of $Y|\mathbf{X}$.
- ▶ We usually do this by minimizing an empirical risk function:

$$\sum (Y_i - \hat{f}(\mathbf{x}_i))^2$$

- ▶ Example: Linear regression

$$\hat{f} \equiv \arg \min_{\beta} \sum (Y_i - \beta_0 - \beta_1 X_{i1} - \dots \beta_p X_{ip})^2$$

Review: Classification

- ▶ Classes $1, \dots, M$ with $P[Y_i \in \text{class } m] = p_{mi}$

$$g(p_{mi}) = f(\mathbf{X})$$

- ▶ Optimization problems vary (maximum likelihood, empirical risk minimization, maximum margins, etc.)
- ▶ Example: logistic regression with two classes

$$\log \frac{p_i}{1 - p_i} = \beta_0 + \beta_1 X_{i1} + \dots \beta_p X_{ip}$$

Issues to Remember

- ▶ Training vs. test error: Worry if your model is **too good** on the data it's trained on. . .
- ▶ Cross validation: Fold data and use it to tune models and estimate test error.
- ▶ Bias-variance tradeoff
- ▶ Curse of dimensionality

Insight on “good” predictive models

- ▶ Take observations with similar \mathbf{X} and assume that they should have similar Y .
- ▶ Many techniques (linear/logistic regression) make you specify the structure of \mathbf{X} and \hat{f} .
- ▶ Why not just try to automatically find observations that have similar \mathbf{X} and Y ?

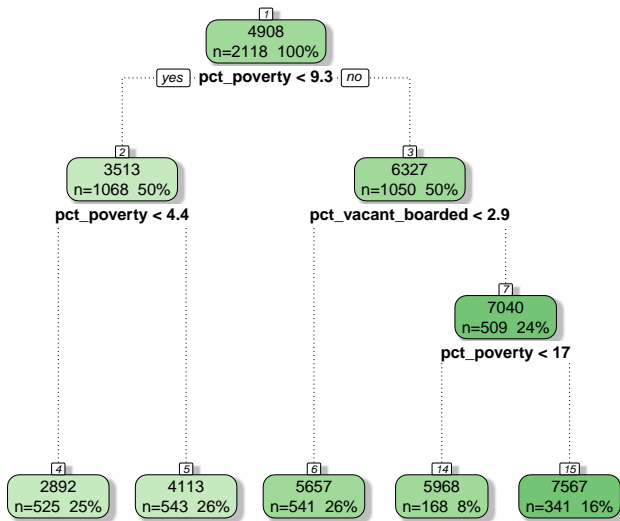
Example: Tree for Nonviolent Crime in Communities

Predict a community's nonviolent crime rate (per 100,000 people) based on the poverty rate and percent of housing that is vacant in that community.

Previously, we've used

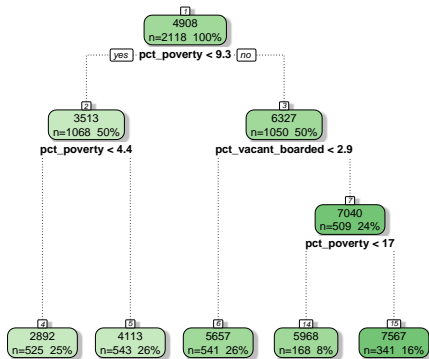
- ▶ Linear regression
- ▶ Regularized regression
- ▶ GAMs

Trees



Tree Predicting Nonviolent Crime in Communities

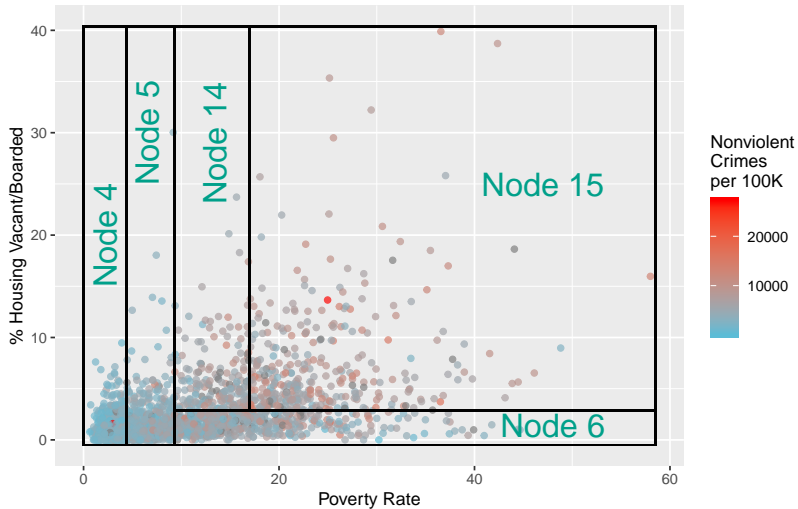
Trees



Tree Predicting Nonviolent Crime in Communities

- ▶ Nodes/Splitting rules
- ▶ Splitting rules are recursive
- ▶ Node/Tree depth
- ▶ Terminal nodes (used to make prediction)
- ▶ $\hat{f}(X)$ = terminal node mean for X

Trees



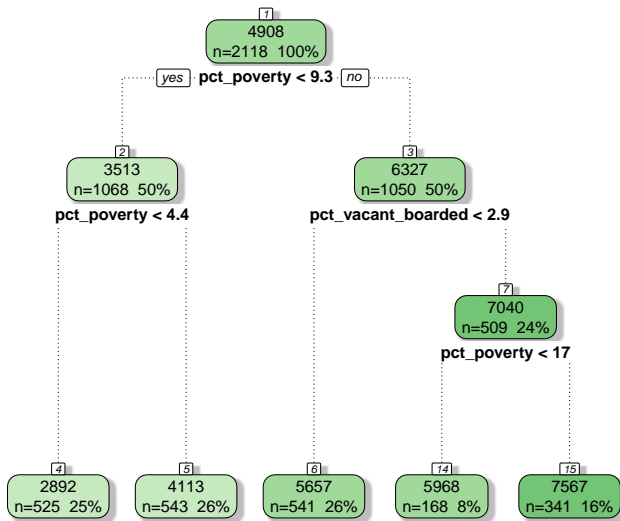
Regression Trees

Splitting rules are determined by a *greedy algorithm*.

1. Start with a tree with one terminal node (the entire data set).
2. For each value of each predictor:
 - 2a. Split the data on that predictor/value
 - 2b. Compute sum of squared errors across all resulting terminal nodes $SSE = \sum_{R_j} \sum (Y_i - \hat{Y}_{R_j})^2$
 - 2c. Choose predictor/value that minimizes SSE
3. Repeat 2 within each resulting terminal node.

Note that at each split, the SSE in terminal nodes decreases.

Recursive Partitioning

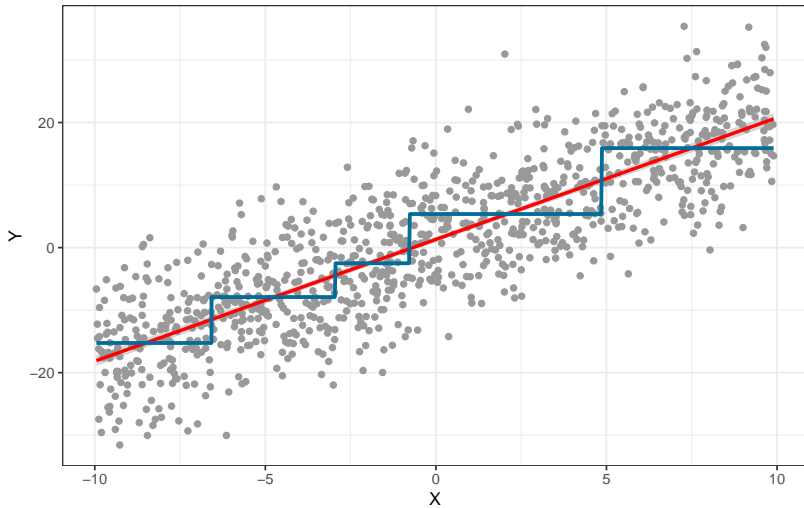


Tree Predicting Nonviolent Crime in Communities

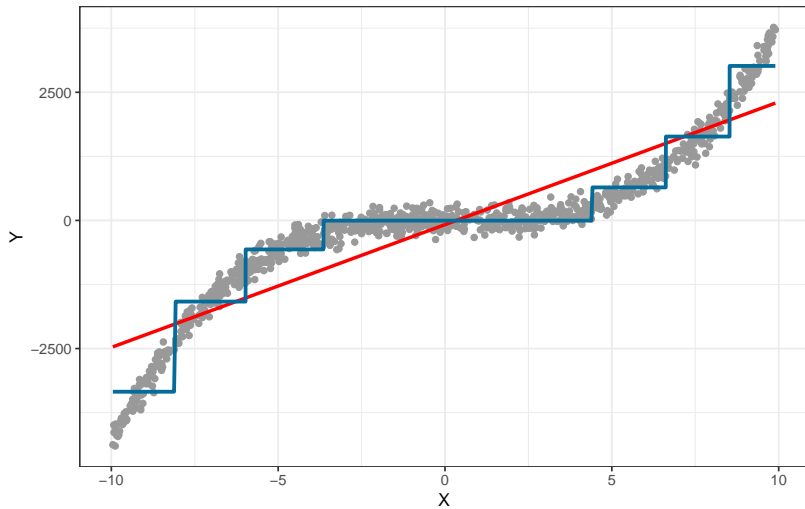
Flexibility

- ▶ Nonlinearity does not need to be specified in advance.
 - ▶ Split points are similar to step functions with splines.
- ▶ Interactions are automatically modelled.
 - ▶ Tree splits on predictor X_1 and then later on X_2 , then the prediction for a given X_1 will depend on the value of X_2 (an interaction).

Tree vs. Linear Model



Flexibility



Classification Trees

Splitting rules are determined by a *greedy algorithm*.

1. Start with a tree with one terminal node (the entire data set).
2. For each value of each predictor:
 - 2a. Split the data on that predictor/value
 - 2b. Compute some metric of impurity (i.e., Gini Index, cross entropy) in the terminal nodes
 - 2c. Choose predictor/value that minimizes impurity index
3. Repeat 2 within each resulting terminal node.

Note that at each split, the purity of terminal nodes increases

Impurity indices

Suppose our classification problem involves K classes, and our tree has m regions.

- ▶ Let \hat{p}_{mk} be the % of observations in region m in class k .
- ▶ Gini index: $\sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk})$
- ▶ Cross entropy: $-\sum_{k=1}^K \hat{p}_{mk} \log \hat{p}_{mk}$

Trees Notes

- ▶ Tree depth can be set *a priori* or could be adaptive (e.g., stop when training MSE is small enough).
- ▶ Trees are static given the data set; if you run a tree the same way on the same dataset, you'll always get the same fit.
- ▶ Deeper trees tend to have lower bias but more variance; shallower trees tend to have high bias but low variance.
- ▶ Deeper trees can be “pruned” to have less variance, but even pruned trees are seldom competitive with more advanced models (even linear models).

Bias-Variance Tradeoff for Trees

Fact 1: Trees will have very high variance.

Fact 2: A model that is the average of models can reduce variance.

Proposed solution: Take the average of a bunch of trees.

Averaging Trees

- ▶ Since trees are not stochastic, if we fit a bunch of trees on a dataset, they'll all be the same. . .
- ▶ Idea 1: Bootstrapping
 - ▶ Bootstrapping = creating B new datasets by sampling rows from your existing dataset **with replacement**.
 - ▶ On average, each bootstrap sample will contain about $1/3$ of the rows of the training data (but they will be repeated)
 - ▶ Each of these datasets will be a little bit different, which means the trees that are fit on them will be a little bit different.
 - ▶ This is gets a handle on how variable the trees are given different datasets you could train them on.

Bootstrap

Bootstrap Samples	1	2	3	...	B
Observation 1	2	0	4		1
Observation 2	1	3	0		0
\vdots	\vdots	\vdots	\vdots	\ddots	\vdots
Observation n	0	1	1		0
	Tree 1 \hat{f}_1	Tree 2 \hat{f}_2	Tree 3 \hat{f}_3		Tree B \hat{f}_B

Averaging Trees

- ▶ Idea 2: Considering only a random subset for splitting.
 - ▶ Trees consider **all** p possible predictors for each split.
 - ▶ Instead, what if we randomly selected only a few predictors ($m_{try} < p$) to consider at each split?
 - ▶ Then, even trees grown this way on the same dataset would be different.

Random Forests

Random forests = Bootstrapping + Randomly selecting predictors

Bootstrap Samples	1	2	3	...	B
Observation 1	2	0	4		1
Observation 2	1	3	0		0
\vdots	\vdots	\vdots	\vdots	\ddots	\vdots
Observation n	0	1	1		0
	Tree 1 \hat{f}_1	Tree 2 \hat{f}_2	Tree 3 \hat{f}_3		Tree B \hat{f}_B

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^B \frac{\hat{f}_i(\mathbf{x})}{B}$$

Conveniences of Random Forests

- ▶ Does not matter how many variables you have (though it may get slow).
- ▶ Can automatically incorporate nonlinearity (though not all smooth functions).
- ▶ Can automatically incorporate interactions (because trees can)
- ▶ Potential on-the-fly tuning with out-of-bag (OOB) error
- ▶ Open up the “black box”
 - ▶ Variable importance
 - ▶ Partial dependence

OOB Error

- ▶ On average, each tree makes use of about $1/3$ of the training data.
- ▶ $2/3$ of the data are a “test” set for a given tree.
- ▶ Predict Y_i of the i th observation with an average of the trees **not** trained on that observation.
- ▶ OOB error is a useful approximation to test error and can be used to tune random forests (e.g., if CV is too hard)

Bootstrap Samples	1	2	3	...	B
Observation 1	2	0	4		1
Observation 2	1	3	0		0
\vdots	\vdots	\vdots	\vdots	\ddots	\vdots
Observation n	0	1	1		0
	Tree 1	Tree 2	Tree 3		Tree B
	\hat{f}_1	\hat{f}_2	\hat{f}_3		\hat{f}_B

Variable Importance

If a variable is “important” then we will split on it frequently, and those splits will greatly reduce SSE or impurity.

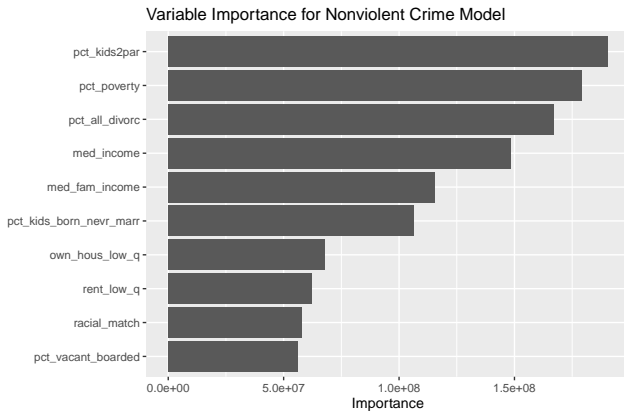
Regression

- ▶ Total amount the SSE decreased when a split occurs on variable X_j (summed across all splits on all B trees).

Classification

- ▶ Total amount the the impurity metric decreased when a split occurs on variable X_j (summed across all splits on all B trees).

Variable Importance



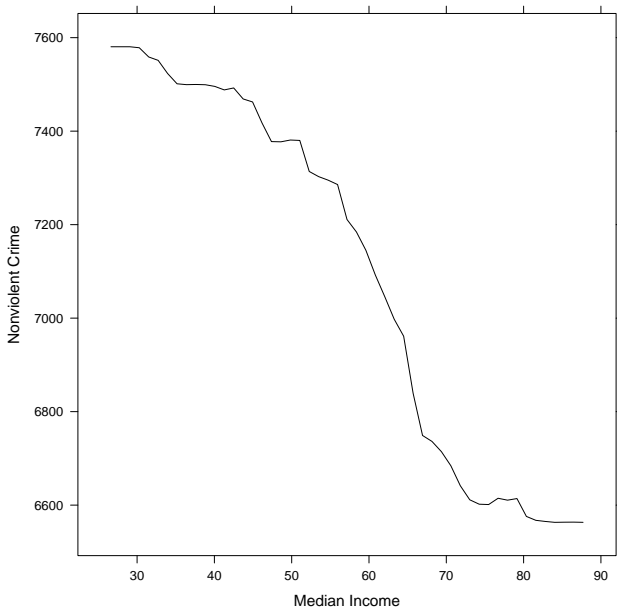
Partial Dependence

We may be interested in the marginal relationship between one predictor X_1 and Y in the random forest: $\hat{f}(X_1)$

We can estimate this from the training data by averaging over the predictions for all other values of X_2, \dots, X_p in the data:

$$\hat{f}(x_1) = \sum_{i=1}^{n_{x_1}} \hat{f}(x_1, X_{i2}, \dots, X_{ip}) / n_{x_1}$$

Partial Dependence



Multiple Partial Dependence & Treatment Effect Variation

Suppose $\mathbf{X} = [T, X_1, \dots, X_p]^T$ includes treatment indicator
 $T \in \{0, 1\}$

Partial dependence can explore the relationship between treatment effect and some variable X_1

$$\begin{aligned}\hat{f}(T = 1, X_1) &= \sum_{i=1}^{n_{T=1, X_1}} \hat{f}(1, X_1, X_{i2}, \dots, X_{ip}) / n_{T=1, X_1} \\ \hat{f}(T = 0, X_1) &= \sum_{i=1}^{n_{T=0, X_1}} \hat{f}(0, X_1, X_{i2}, \dots, X_{ip}) / n_{T=0, X_1} \\ &\hat{f}(T = 1, X_1) - \hat{f}(T = 0, X_1)\end{aligned}$$

Tuning Random Forests

1. Number of predictors to consider for each split m_{try}
2. Number of trees
3. Tree depth

How do we find the right tuning parameter values?

Lab Time

Please access the lab on random forests now.