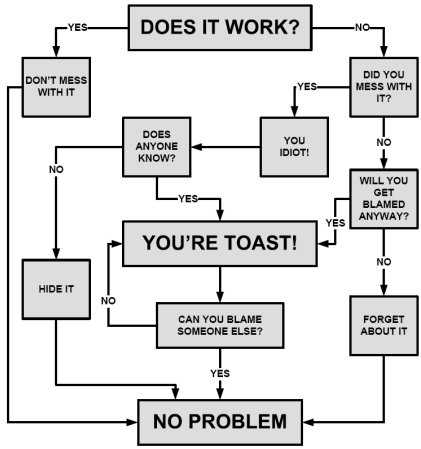
Data, Environment and Society: Lecture 18: Regression trees, ctd

Instructor: Duncan Callaway
GSI: Salma Elmallah

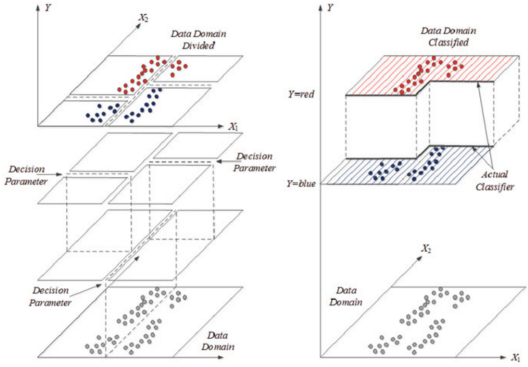
October 31, 2019



https://thenexttobestblogever.wordpress.com/2009/11/07/problem-solving-flowchart-2/

Objectives

- Quick review of Lecture 16 on regression trees
  - Terminology
  - "T_0"
- How to choose the right tree via cross validation
- Classification trees
  - Same as regression, just different loss functions



(medium.com)

Terminology we'll cover...

- Terminal node
- Internal node
- Branches
- Leaves
- Binary splits
- Recursive binary splitting $\leftrightarrow$ Top-down greedy
- Cost complexity pruning

Basic idea for regression trees

All we are doing is "splitting" the observations into regions in the feature space, and averaging the response variable within each region.

Basic idea for regression trees

All we are doing is "splitting" the observations into regions in the feature space, and averaging the response variable within each region.

Doing predictions with the model just involves

- locating a set of **features** in a region,
- then setting the response variable equal to the **average** from the training **response** variable in that region.

Basic idea for regression trees

All we are doing is "splitting" the observations into regions in the feature space, and averaging the response variable within each region.

Doing predictions with the model just involves

- locating a set of **features** in a region,
- then setting the response variable equal to the **average** from the training **response** variable in that region.

Big decision in regression trees: *What are the regions we should use?*

Regression trees – basic approach

1. Divide the **feature** space into non-overlapping regions
   - This distinguishes the method from KNN regression
2. Within each region, the prediction is just the average of the **response variable** from training data.
   - This is similar to KNN regression

Regression trees – basic approach

1. Divide the **feature** space into non-overlapping regions
   - This distinguishes the method from KNN regression
2. Within each region, the prediction is just the average of the **response variable** from training data.
   - This is similar to KNN regression

Two Basic Questions:

1. Where should I put the internal nodes (i.e. the "splits")?
2. How many regions should there be?

The answers are, as it turns out, really simple.

Where should the splits be?

Then we partition any region by choosing j and s as follows:

$$\{j, s\} = \arg \min_{j \in J, s \in X_j} \sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

where $\hat{y}_{R_k}$ is the mean of all response variables in region k.

It would be tedious to identify j and s by hand, but it's actually very quick computationally.

Question: How many j-s pairs for p features and n observations?

Where should the splits be?

Then we partition any region by choosing j and s as follows:

$$\{j, s\} = \arg \min_{j \in J, s \in X_j} \sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

where $\hat{y}_{R_k}$ is the mean of all response variables in region k.

It would be tedious to identify j and s by hand, but it's actually very quick computationally.

Question: How many j-s pairs for p features and n observations?

- No more than $p(n-1)$, since we can only choose $(n-1)$ boundaries between observations.
- There may be fewer, if separate observations share the same values for some of their features.

Repeating the splits

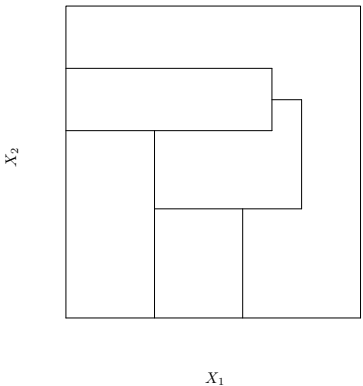
On each step, we're choosing the single best possible split from among the R regions, resulting in $R + 1$ regions to take to the next step.

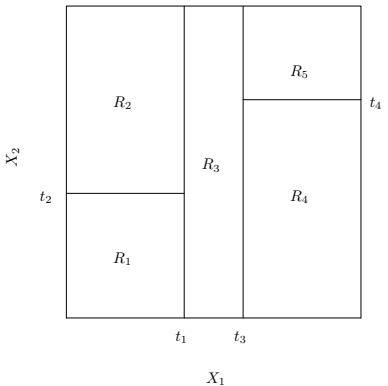
Repeat this process until you reach a stopping criterion – typically a maximum number of observations in each region. (For example all regions have no more than 5 observations.)

Call the resulting tree T_0.

We call this approach "greedy" because when we do the first partition we're not thinking ahead to future partitions to evaluate it.

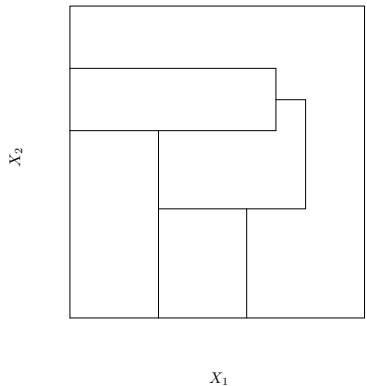
One of these doesn't belong...

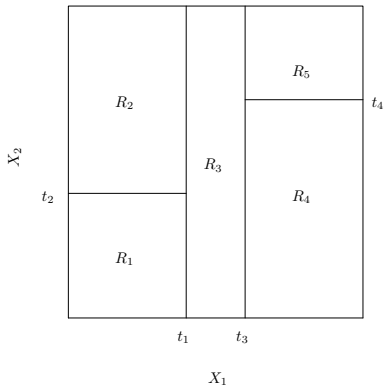




Q: Which picture results from successively splitting the regions into values greater or less than feature values?

One of these doesn't belong...

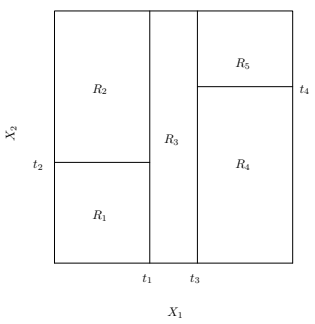


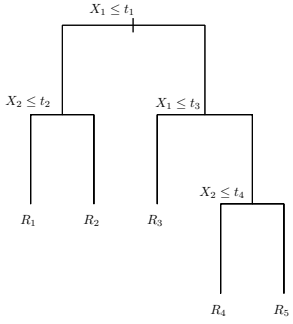


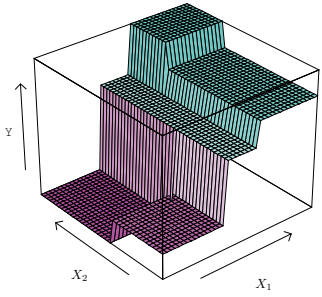
Q: Which picture results from successively splitting the regions into values greater or less than feature values?

A: The right one. The left one is not possible with simple splitting.

A five region example... with two dimensional feature space







What do we call it?

The process of splitting regions over and over is called...

“recursive binary splitting”

You can also call it a **“top-down greedy”** approach.

Because it’s “greedy” we can’t be sure that the splits we’re getting are the best possible splits.

Why binary?

In other words, why not multiway splits?

Why binary?

In other words, why not multiway splits?

In general multiway splits fragment the data too quickly, leaving insufficient data at the next level down

Since we do the binary splitting recursively, we get the same flexibility as a multiway split, since a region can be split a second time later.

Terminology so far...

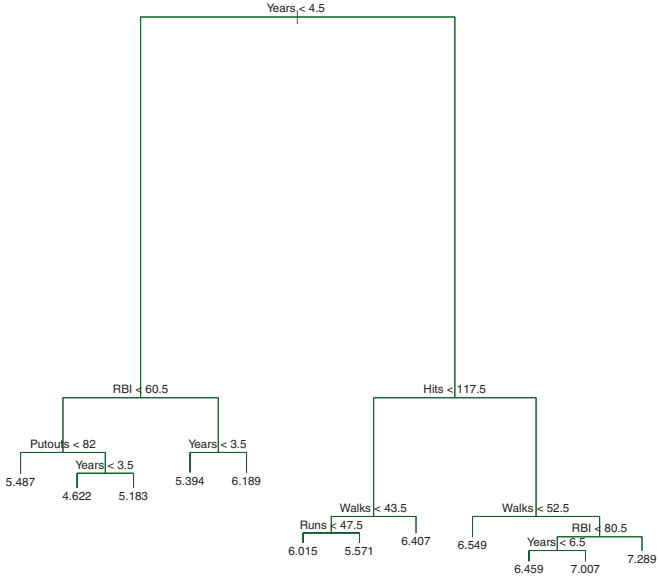
- Terminal node
- Internal node
- Branches
- Leaves

Terminology so far...

- Terminal node
- Internal node
- Branches
- Leaves
- Binary splits
- Recursive binary splitting $\leftrightarrow$ Top-down greedy

Example T_0

Remember, T_0 is the biggest tree we build. We get there by recursively splitting until we meet a threshold (often a maximum number of observations per terminal node).



When will we test?

All the steps above involve model *building*. We have yet to evaluate different models against one another. First let's build the candidate models, then we can evaluate.

Step 1: "cost complexity pruning"

We'll test models that are **subtrees** of T_0. (trees that are the same as T_0 except they are missing some internal nodes and branches).

We identify subtrees using **cost-complexity pruning** a.k.a. weakest link pruning:

- To get the first subtree, evaluate model performance for all subtrees with one leaf removed from T_0. Choose the best one, call it T_1.
  - R^2 works for measuring performance
  - ...but not for categorical variables, stay tuned!
- Then evaluate performance for all models with one leaf removed from T_1. Choose the best, call it T_2. And so on.

Step 1: "cost complexity pruning"

We'll test models that are **subtrees** of T_0. (trees that are the same as T_0 except they are missing some internal nodes and branches).

We identify subtrees using **cost-complexity pruning** a.k.a. weakest link pruning:

- To get the first subtree, evaluate model performance for all subtrees with one leaf removed from T_0. Choose the best one, call it T_1.
  - R^2 works for measuring performance
  - ...but not for categorical variables, stay tuned!
- Then evaluate performance for all models with one leaf removed from T_1. Choose the best, call it T_2. And so on.

(Smart researchers have shown that this "greedy" approach is an optimal *pruning* strategy. But recursive binary splitting is not always optimal for growth.)

Step 2: Tune up your α

Take your set of subtrees, T_0 through T_{N-2}. Call $|T|$ the number of terminal nodes in the tree.

For a given α, *one* of the T_i will minimize :

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

- $\alpha = 0$ will choose T_0, the biggest tree.
- As α grows you'll choose successively smaller trees.

Quick quiz

For a given α, *one* of the T_i will minimize :

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Fill in the blank: As α increases, bias goes __ and variance goes _____.

Quick quiz

For a given α, *one* of the T_i will minimize :

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

Fill in the blank: As α increases, bias goes **up** and variance goes **down**. Bigger α means fewer leaves, which means more bias but less variance.

Though it seems unnecessary to define α (why not just evaluate all subtrees?), we'll see it's useful for cross validation.

The (cross validation) process

1. Split your data into K folds.
2. Repeat this process for each fold: Withhold the fold and for remaining training data:
   a. Grow a large tree via recursive binary splitting. "Large" means each leaf has some pre-specified maximum number of observations (e.g. 5)
   b. Then "prune" the tree via cost complexity pruning to get a sequence of subtrees.
   c. Choose the tree in the sequence that minimizes $\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$ for each of a range of values of α.
   d. Record the test MSE (i.e. MSE on withheld fold) for each value of α.
3. Average the test MSE across all folds *for each value of* α,
4. Choose the α that gives the lowest cross validated error,
5. Build your final model with the chosen α with *all the data.*

Why use α?

Why didn't we just evaluate cross validated error for each tree size?

That is, is α just overly complicating things?

Why use α?

Why didn't we just evaluate cross validated error for each tree size?

That is, is α just overly complicating things?

Ans: Sometimes it might. But it may be that across different folds we'd choose different subtrees. α provides a better representation of the bias-variance tradeoff across folds.

But: out of convenience the book *displays* results in terms of tree size rather than α. Argh!

Terminology so far...

- Terminal node
- Internal node
- Branches
- Leaves
- Binary splits
- Recursive binary splitting $\leftrightarrow$ Top-down greedy

Terminology so far...

- Terminal node
- Internal node
- Branches
- Leaves
- Binary splits
- Recursive binary splitting $\leftrightarrow$ Top-down greedy
- Cost complexity pruning
- Cross validation to choose best α

Questions

Do regression trees utilize linear regression?

Questions

Do regression trees utilize linear regression? Nope.

What do you think their advantages are (vs LASSO or nonlinear models...)?

Questions

Do regression trees utilize linear regression? Nope.

What do you think their advantages are (vs LASSO or nonlinear models...)?

- Easy to explain and non-experts can understand the results.
- They're more like human decision-making. Doctors like them.
- They easily handle qualitative feature – no need for dummies.

Questions

Do regression trees utilize linear regression? Nope.

What do you think their advantages are (vs LASSO or nonlinear models...)?

- Easy to explain and non-experts can understand the results.
- They're more like human decision-making. Doctors like them.
- They easily handle qualitative feature – no need for dummies.

How about some disadvantages?

Questions

Do regression trees utilize linear regression? Nope.

What do you think their advantages are (vs LASSO or nonlinear models...)?

- Easy to explain and non-experts can understand the results.
- They're more like human decision-making. Doctors like them.
- They easily handle qualitative feature – no need for dummies.

How about some disadvantages?

- As described, they don't usually provide the same predictive power that the other tools we've studied can.
- They can be pretty sensitive to small changes in the data.
- Recursive binary splitting may generate a suboptimal tree (like forward / backward model selection). Things later in the chapter address this.