Ch 8.1: Decision Trees

Lecture 16 - CMSE 381

Michigan State University
::
Dept of Computational Mathematics, Science & Engineering

Wed, March 20, 2024

Announcements

Last time:

- Cubic Splines

This lecture:

- 8.1 Decision Trees

Announcements:

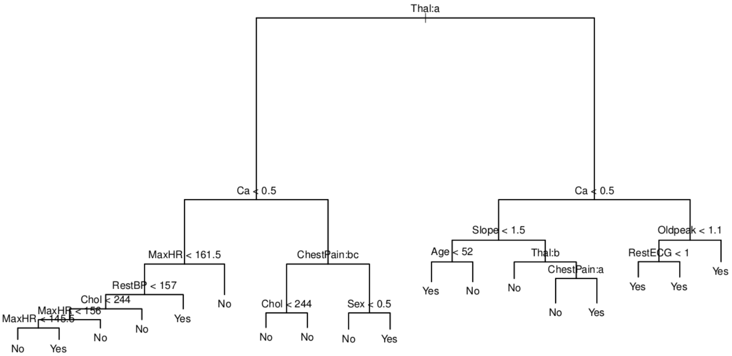
- Mid-term exam 2: next week

Section 1

Decision Trees

Big idea

- Previously, we've always had some form of function for prediction: $f(X_1, \cdots, X_p) = \hat{Y}$
- Now, we make a sequence of decisions to make a prediction, either regression or classification version



Subset of Hitters data

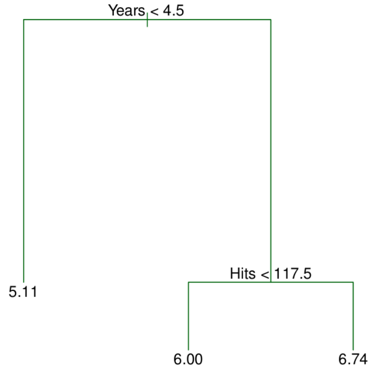
	Hits	Years	Salary	LogSalary
1	81	14	475.0	6.163315
2	130	3	480.0	6.173786
3	141	11	500.0	6.214608
4	87	2	91.5	4.516339
5	169	11	750.0	6.620073
...	...	...	...	...
317	127	5	700.0	6.551080
318	136	12	875.0	6.774224
319	126	6	385.0	5.953243
320	144	8	960.0	6.866933
321	170	11	1000.0	6.907755

- Remove observations missing salary values
- log transform salary for something closer to bell shape
- Goal: predict log salary (can reverse by returning $\exp(x)$ if model returns x)

A simpler decision tree example

	Hits	Years	LogSalary
1	81	14	6.163315
2	130	3	6.173786
3	141	11	6.214608
4	87	2	4.516339
5	169	11	6.620073
...	...	...	...
317	127	5	6.551080
318	136	12	6.774224
319	126	6	5.953243
320	144	8	6.866933
321	170	11	6.907755

- Top split assigns observations with *Years* < 4.5 to left branch
- Return mean response for players with that.
- Predictions:
  - mean log salary is 5.107, so returns $\exp(5.107) = \$165.174$ thousand dollars
  - $5.999 \Rightarrow \$402,834$
  - $6.740 \Rightarrow \$845,346$



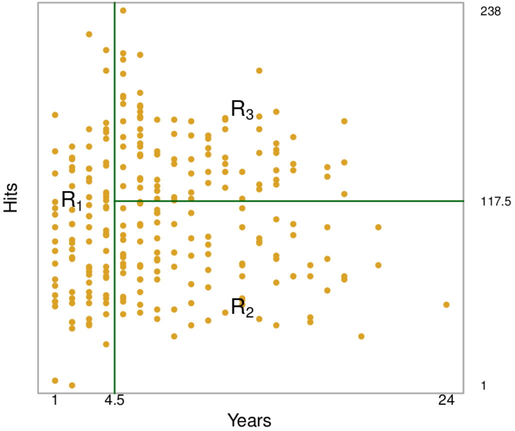
Interpretation of example

- Years most important factor for determining salary
- Players with less experience earn lower salaries than more experienced
- For the less experienced players, number of hits plays little role in salary
- For more experienced players, number of hits affects it
- Likely an oversimplification of real relationship, but easier to interpret and has nice graphical representation

Regions defined by the tree

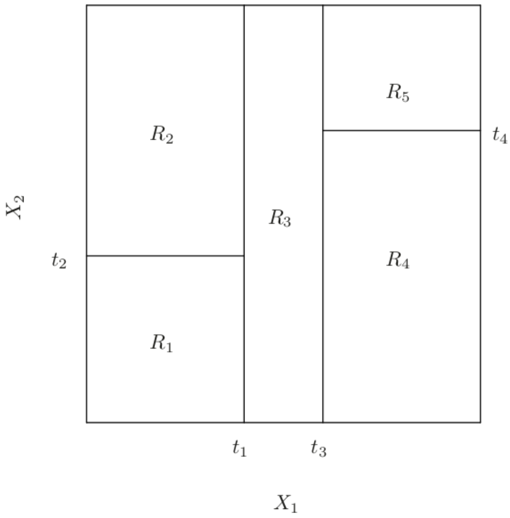
- Point out tree is "upside down"
- Leaves of the tree:
  - $R_1 = \{X \mid Years < 4.5\}$
  - $R_2 = \{X \mid Years >= 4.5, Hits < 117.5\}$
  - $R_3 = \{X \mid Years >= 4.5, Hits >= 117.5\}$
- Other splits are called Internal Nodes
- Segments that connect nodes called Branches or Edges

Viewing Regions Defined by Tree



Viewing Regions Defined by Tree

1. We divide the predictor space – that is, the set of possible values for $X_1, X_2, \cdots, X_p$ — into J distinct and non-overlapping regions, $R_1, R_2, \cdots, R_J$.
2. For every observation that falls into the region R_j, we make the same prediction $=$ the mean of the response values for the training observations in R_j.



How to build the tree?

Step 1, grow the tree iteratively: in each step, we decide which region R_j to add.
Step 2, prune the tree: cut the unnecessary branches

Step 1: How do we decide on R_js?

Training error: For any fixed partition, $R_1, \cdots, R_J$, the training error is defined as

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

$\hat{y}_{R_j}$ = mean response for training observations in jth box

Goal:

Find optimal boxes $R_1, \cdots, R_J$ that minimize

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

- Can't actually check every possible partition
- Instead, go for top-down greedy approach called *Recursive binary splitting*
- Begins at top of tree with all data points
- Uses best split at every step

Recursive Binary Splitting

In the first iteration,

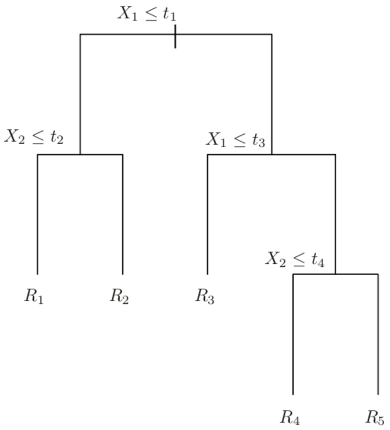
- Pick X_j and s, so that splitting into $\{X \mid X_j < s\}$ and $\{X \mid X_j \geq s\}$ results in largest possible reduction in RSS

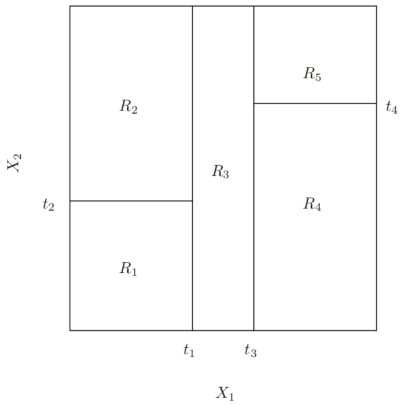
$$R_1(j,s) = \{X \mid X_j < s\}$$
$$R_2(j,s) = \{X \mid X_j \geq s\}$$

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

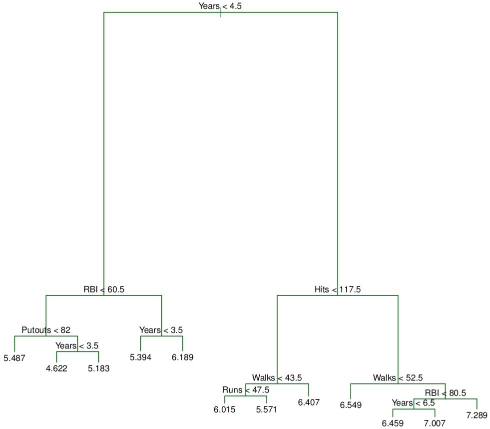
Repeat the process

- Do this repeatedly
- Each time can split one of the previously identified regions
- Keep going until some stopping criterion is reached. E.g. until each region has at most 5 observations





Step 2: Pruning



- Big trees leave you open to potential overfitting
- Could just stop building earlier, but that's short sighted
- Instead, grow a big tree and prune it back
- Find subtree with best test error rate
- Too many subtrees to test them all

Weakest Link Pruning

Also called Cost complexity pruning

For every α, there is a subtree T that minimizes:

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

- $|T| =$ number of terminal nodes of T
- R_m is rectangle for mth terminal node
- $\hat{y}_{R_m}$ is mean of training observations in R_m

- $\alpha = 0$ gets entire tree
- Increasing α penalizes size of tree
- Branches pruned from tree in nested and predictable fashion
- Easy to get trees for all vales of α
- Pick α via CV

Algorithm version

Algorithm 8.1 *Building a Regression Tree*

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

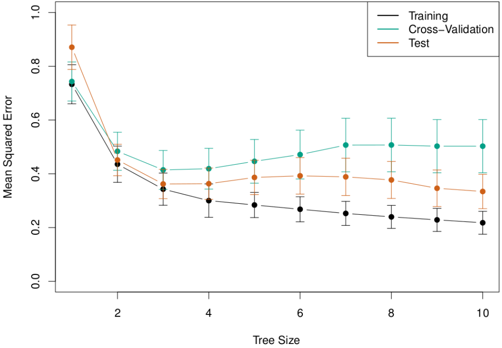
   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.

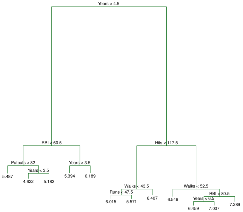
   (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.

4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Messing with α





Result of pruning is the three leaf tree on the right

A small exercise

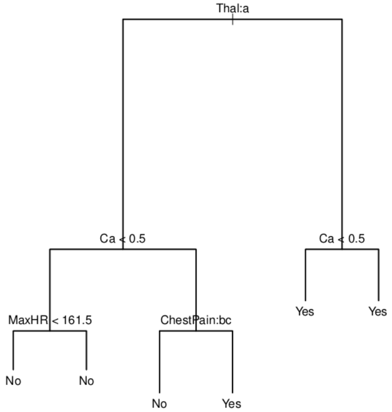
Build a decision tree for the following training data with $\alpha = 10$.

	X1	X2	Y
1	0	1	6
2	1	0	1
3	1	1	-1

Section 2

Classification Decision Tree

Basic idea



- Example from heart data
- Can't use RSS, need error rate
- Could decide splits by classification error rate
- Gives too much emphasis on large classes, so use something else
- Use two other options
- $\hat{p}_{mk}$ = proportion of training observations in R_m from the kth class
- $E = 1 - \max_k(\hat{p}_{mk})$ *Fraction of training observations not in the most common class*

Gini index

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

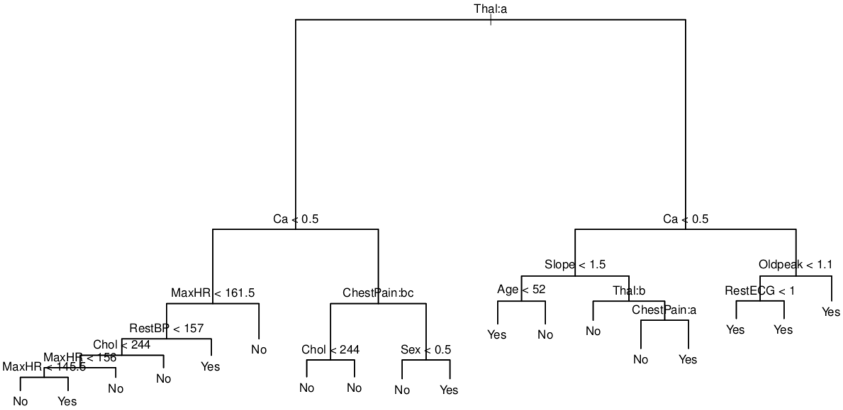
- Measure of total variance across K classes
- small value if all $\hat{p}_{mk}$'s close to zero or 1
- Small value means node contains mostly observations from one class

Entropy

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

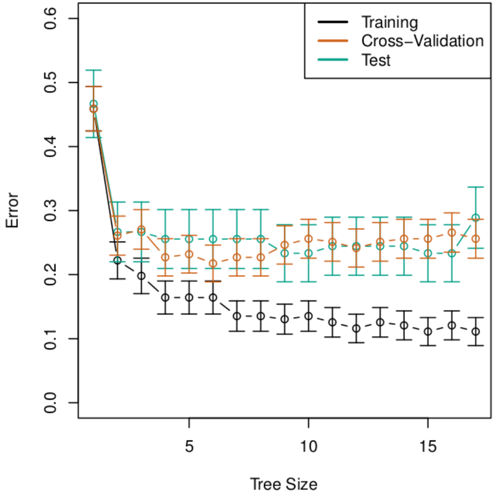
- Positive because $0 \leq \hat{p}_{mk} \leq 1$
- Near zero if $\hat{p}_{mk}$ all near 0 or near 1
- Small value if mth node has majority one class

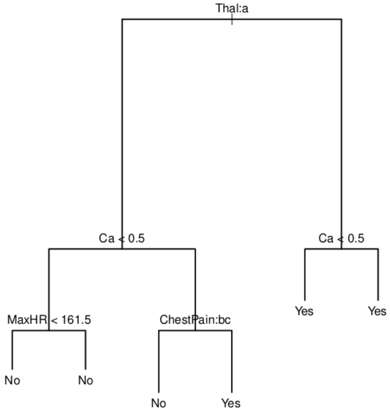
Example



Thal:a
Ca < 0.5
Ca < 0.5
MaxHR < 161.5
ChestPain:bc
Slope < 1.5
Oldpeak < 1.1
Age < 52
Thal:b
RestECG < 1
ChestPain:a
RestBP < 157
Chol < 244
MaxHR < 156
MaxHR < 145.5
Chol < 244
Sex < 0.5
No
Yes
No
No
Yes
No
No
No
No
Yes
Yes
No
No
No
Yes
Yes
Yes
Yes

Pruning the example



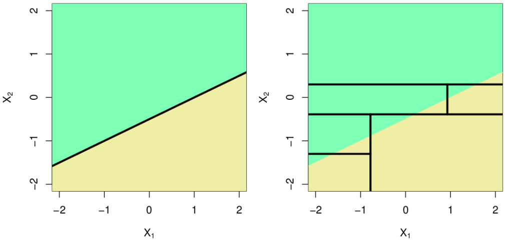


Another small exercise

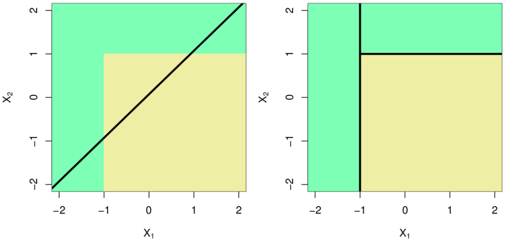
Build a decision tree for the following training data with $\alpha = 0.1$.

	X1	X2	Y
1	-1	2	0
2	1	0	1
3	2	-1	1

Linear models vs trees



Obviously linear does better here



Not going to beat this case though

Pros/Cons

Pros:

- Trees are very easy to explain to people. Often easier to explain than linear regression!
- Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- Trees can easily handle qualitative predictors without the need to create dummy variables.

Cons:

- Not as accurate as other methods of classification and regression
- Not robust: small change in data can cause large change in estimated tree
- Fix..... aggregate many decision trees

Summarize

- Split into regions by greedily decreasing RSS
- Prune tree by using cost complexity
- Not robust - Next time, figure out how to aggregate trees