40.016: The Analytics Edge
Week 8 Lecture 2

FORECASTING THE SUPREME COURT'S DECISIONS WITH CARTS (PART 2)

Term 5, 2022

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Outline

1. Brief recap on CARTs
2. Pruning
3. Trees versus linear models
4. Advantages and Disadvantages of CARTs

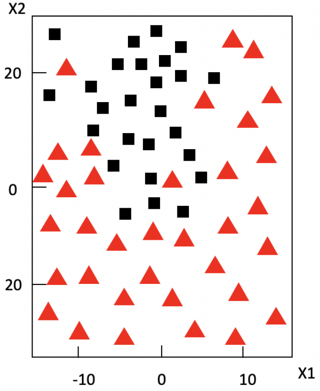
Outline

1. Brief recap on CARTs
2. Pruning
3. Trees versus linear models
4. Advantages and Disadvantages of CARTs

CARTs

- Decision Trees can be applied to both regression and classification problems
- The term Classification And Regression Tree (CART) is used to refer to procedures that learn a Classification or Regression Tree
- CART always creates a binary tree, meaning each non-terminal node has two child nodes
- One big appeal CART has is that the decision process is very akin to how we humans make decisions. Therefore it is easy to understand and accept the results

Is the output red?



Structure of a Decision Tree

Learning algorithm (for regression)

Step 1. Partition the predictor space into J distinct and non-overlapping regions $(R_1, R_2, \dots, R_J)$. This process uses recursive binary splitting and minimizes the RSS.

- Exhaustive search to find: which predictor? which cut point?
- Number of searches when splittig one region into two: $n \times p$
  (n: # training observations in this region, p: # predictors)
- In recursive binary splitting, for predictor k and cut s, set

$$R_1(k,s) = \{X | X^{(k)} < s\} \text{ and } R_2(k,s) = \{X | X^{(k)} \geq s\}$$

Step 2. For every observation that falls into the region R_j, we make the same prediction c_j (mean of the response values for the training observations in R_j)

Example

X	(1,0)	(3,1)	(2,2)
Y	1	10	4

Case 1: choose X_1, sort observations of X by the order of X_1: $(1, 0), (2, 2), (3, 1)$

s	1	2	3
R_1	N.A.	(1,0)	(1,0) (2,2)
R_2	(1,0) (2,2) (3,1)	(2,2) (3,1)	(3,1)
c_1	N.A.	1	2.5
c_2	5	7	10
RSS	$(1-5)^2 + (4-5)^2 + (10-5)^2 = 42$	$(4-7)^2 + (10-7)^2 = 18$	$(1-2.5)^2 + (4-2.5)^2 = 4.5$

Case 2: choose X_2, sort observations of X by the order of X_2: $(1, 0), (3, 1), (2, 2)$

s	0	1	2
R_1	N.A.	(1,0)	(1,0) (3,1)
R_2	(1,0) (3,1) (2,2)	(3,1) (2,2)	(2,2)
c_1	N.A.	1	5.5
c_2	5	7	4
RSS	$(1-5)^2 + (10-5)^2 + (4-5)^2 = 42$	$(10-7)^2 + (4-7)^2 = 18$	$(1-5.5)^2 + (10-5.5)^2 = 40.5$

Among 6 choices, lowest RSS $= 4.5$. The chosen predictor is X_1 and the chosen cut is $s = 3$.

Learning algorithm (for classification)

For Classification Trees, we use the same procedure, but:

- We use a measure of impurity (instead of RSS) in the partitioning process, such as: Classification error rate, Gini index, Entropy.
- The prediction c_j is the most commonly occurring class.

Example

ID	1	2	3	4
Annual income $\geq$ 100k	Y	Y	N	N
Own houses	Y	N	N	Y
Result	Y	Y	N	Y

Case 1: choose "Annual income $\geq$ 100k":

R_1(Yes): $\{1, 2\}$, $p_{1,1} = 1$, $p_{1,2} = 0$, $G_1 = 0$

R_2(No): $\{3, 4\}$, $p_{2,1} = 0.5$, $p_{2,2} = 0.5$, $G_2 = 0.5$

$G = \frac{2}{4}G_1 + \frac{2}{4}G_2 = 0.25$

Case 2: choose "Own houses":

R_1(Yes): $\{1, 4\}$, $p_{1,1} = 1$, $p_{1,2} = 0$, $G_1 = 0$

R_2(No): $\{2, 3\}$, $p_{2,1} = 0.5$, $p_{2,2} = 0.5$, $G_2 = 0.5$

$G = \frac{2}{4}G_1 + \frac{2}{4}G_2 = 0.25$

Lowest $G = 0.25$.

Back to R!

To learn CARTs, we will use the function `rpart`, implemented in the package ... `rpart`:

`model <- rpart(formula, data, method, ...)` – fit CART
`print(model)` or `summary(model)` – print details of CART
`prp(model,type)` or `plot(model);text(model)` – visualize CART
`predict(model,newdata,method)` – prediction via CART

Outline

1 Brief recap on CARTs

2 Pruning

3 Trees versus linear models

4 Advantages and Disadvantages of CARTs

Overfitting of CART

- The CART's learning algorithm is likely to build complex trees that overfit the training data.
  - When we stop recursive binary splitting?
    – One possible way is to set minimum number of observations in the leaves.
  - The number of leaves is at most equal to the number of observations.
- A smaller tree (with fewer regions $R_1, \cdots, R_J$) may lead to lower variance at the cost of a little bias.

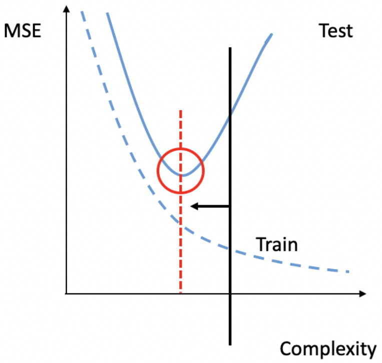
Recall Bias-Variance tradeoff

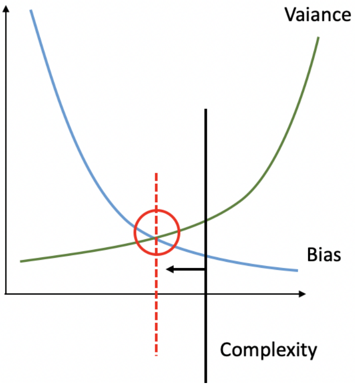
Fit $y = f(x) + \varepsilon$, $\mathbb{E}[\varepsilon] = 0$, $\mathrm{Var}[\varepsilon] = \sigma^2$. Fit a model $\hat{y} = \hat{f}(x)$ by solving $\min(y - \hat{f}(x))^2$. For any fixed x_0, we have

$$\begin{aligned} \mathbb{E}[(y_0 - \hat{f}(x_0))^2] &= \left(\mathbb{E}[\hat{f}(x_0)] - f(x_0)\right)^2 + \mathrm{Var}[\hat{f}(x_0)] + \sigma^2 \\ &= \text{Bias}^2 + \text{Variance} + \text{Random Error} \end{aligned}$$

- Bias: the difference between the average prediction of our model and the correct value which we are trying to predict
  – the fitting performance of model
- Variance: captures how much your performance changes if you train on a different training set
  – the effect of data perturbation

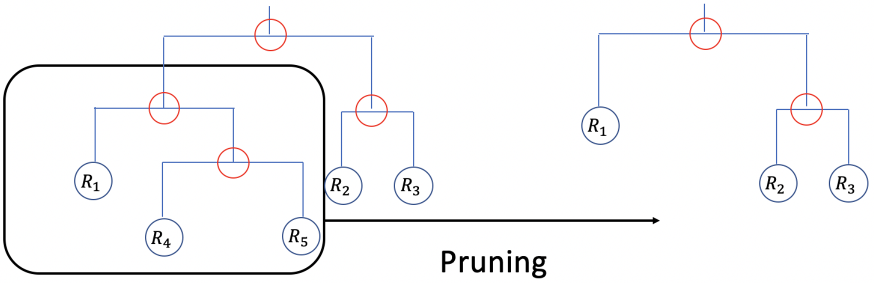
Bias-Variance tradeoff





Complexity of CARTs

Complexity of a CART $\sim$ "Depth" or "Width" of the tree $\sim$ Number of leaves



Pruning

- Early stopping or pre-pruning:

  Try and stop the tree-building process early, before it produces leaves with very small number of observations.

  - At each stage of splitting the tree, we check the cross-validation error.
  - If the error does not decrease significantly enough, then we stop.
  - Early stopping may underfit by stopping too early
    – The current split may be of little benefit, but having made it, subsequent splits may significantly reduce the error.

- Pruning or post-pruning:
  – usually lower risk of underfitting, higher test accuracy, more branches, longer computational time

Pruning

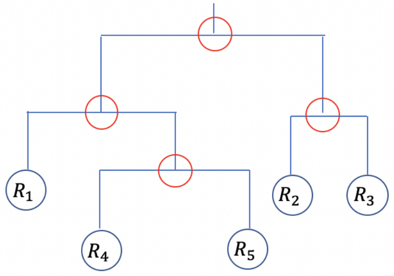
Idea:

- Grow a large tree T_0 and then
- Prune it back to obtain a subtree

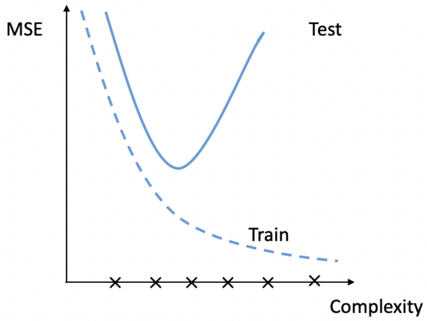
Some considerations:

- To determine how to prune the tree, we can use the cross-validation error
- We cannot calculate the cross-validation error for all trees, because there are too many subtrees $\rightarrow$ it would take too long!

Discrete value of “complexity”



9 nodes, 5 leaves



Discrete value of “complexity”

Cost complexity pruning (or weakest link pruning)

We begin by regression trees. For each value of a nonnegative tuning parameter α, there corresponds a subtree $T \subset T_0$ s.t. the value

$$\underbrace{\sum_{m=1}^{|T|} \sum_{i:X_i \in R_m} (y_i - c_m)^2}_{\text{RSS: measure accuracy}} + \alpha \underbrace{|T|}_{\text{Complexity}}$$

is as small as possible. Here $|T|$ means the number of leaves in T.

- If $\alpha = 0$, $T = T_0$
- If α increases, $|T|$ decreases (we pay a price for building a tree with many leaves)
- The above expression is a reminiscent of the LASSO, which controls the complexity of a linear model (Week 5, Lecture 2)

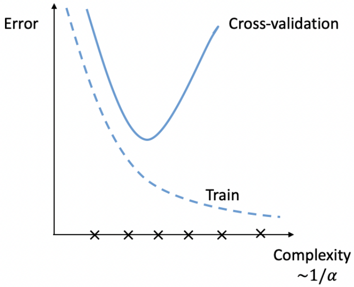
(Full) algorithm for building a regression tree

Step 1 Use recursive binary splitting to grow a large tree on the training data

Step 2 Use cost complexity pruning to obtain a sequence of subtrees as a function of α

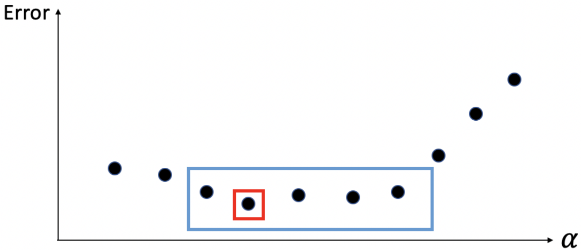
Step 3 Use k-fold cross-validation to choose the best value of α

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



How to choose the best value of α?

Method 1: choose the value of α with the smallest k-fold cross-validation error



Method 2: "One-standard error" rule
– choose the largest α whose cross-validation error is still within a SE of the minimum possible cross-validation error

How do we prune a Classification Tree?

We follow the same procedure, keeping in mind that the model error is calculated with a **measure of impurity**. This leads to the following expression

$$\underbrace{\sum_{m=1}^{|T|} E_m}_{\text{Error: measure accuracy}} + \alpha \underbrace{|T|}_{Complexity}$$

which we still want to minimize.

Back to R!

To learn CARTs, we will use the function `rpart`, implemented in the package ... `rpart`:

`model <- rpart(formula, data, method, ...)` – fit CART

`print(model)` or `summary(model)` – print details of CART

`prp(model,type)` or `plot(model);text(model)` – visualize CART

`predict(model,newdata,method)` – prediction via CART

`printcp(model)` or `plotcp(model)` – the table or plot of α

`cp <- model$cptable[which.min(model$cptable[,"xerror"]),"CP"]`
– optimal α (by Method 1)

`model2 <- prune(model,cp)` – prune CART with given α

Outline

1. Brief recap on CARTs
2. Pruning
3. Trees versus linear models
4. Advantages and Disadvantages of CARTs

Trees versus linear models

Let’s compare a linear regression model

$$\hat{f}(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

to a regression tree

$$\hat{f}(X) = \sum_{j=1}^{J} c_j 1_{R_j}(X).$$

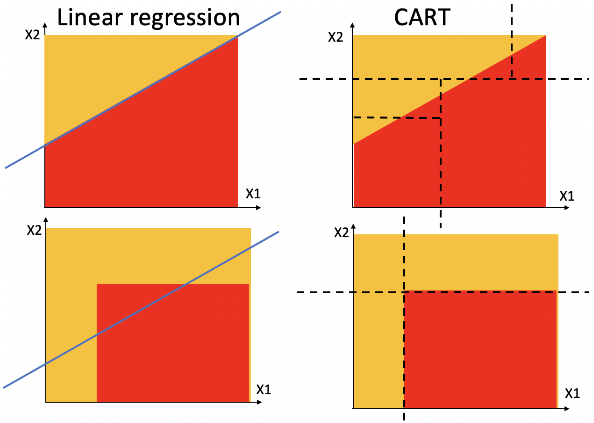
Recall the indicator function $1_{R_j}(\cdot)$ is defined as

$$1_{R_j}(X) = \begin{cases} 1 & X \in R_j \\ 0 & \text{otherwise} \end{cases}.$$

Trees versus linear models

Which model is better? The answer depends on the problem at hand.

Example:



Outline

1 Brief recap on CARTs

2 Pruning

3 Trees versus linear models

4 Advantages and Disadvantages of CARTs

Advantages and Disadvantages of CARTs

Pros:

- Interpretability
- Can be displayed graphically
- Can handle qualitative predictors (that take no continuous values)
- No assumptions on the relationship between input and output variables

Cons:

- They are not very accurate
- Not robust

References

- Martin et al. (2004) Competing approaches to predicting supreme court decision making. *Perspectives on Politics*, 2 (4), 761767.
- James et al. (2014) *An Introduction to Statistical Learning with Applications in R*, Springer, 2014. Chapter 8.1.