NBA 4920/6921 Lecture 16

Tree-based Methods: Regression

Murat Unal

Johnson Graduate School of Management
Cornell University

10/26/2021

Agenda

Regression Trees

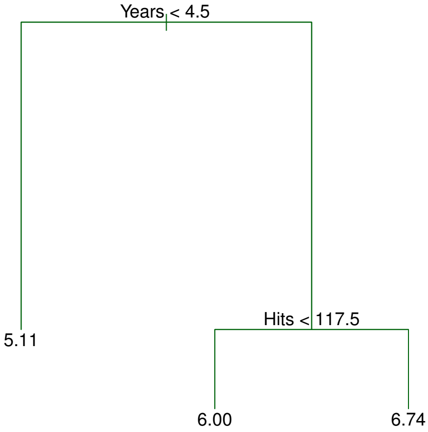
Application in R

Tree-based methods

Work by splitting the feature space into regions and then making predictions based on the most common occurrences within a given region

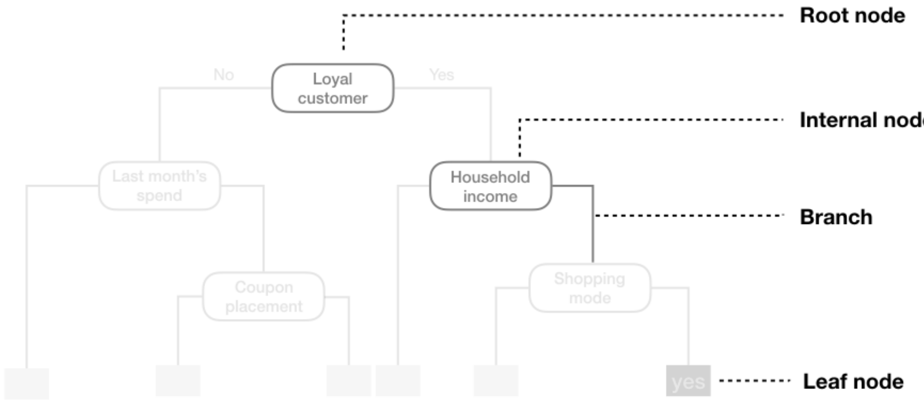
- Simple and useful for interpretation
- Can use for both regression and classification problems
- Have a nonlinear structure
- Less predictive power compared to other methods
- Combining many trees leads to ensemble methods

Decision tree



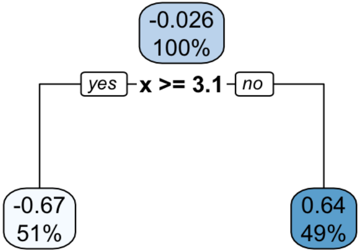
- Decision trees are drawn upside down
- The bottom of the tree are the leaves or terminal nodes
- The number in each leaf is the mean of the response for the observations that fall there.
- The points that split the feature space are the internal nodes
- `Years`< 4.5 and `Hits`< 117.5

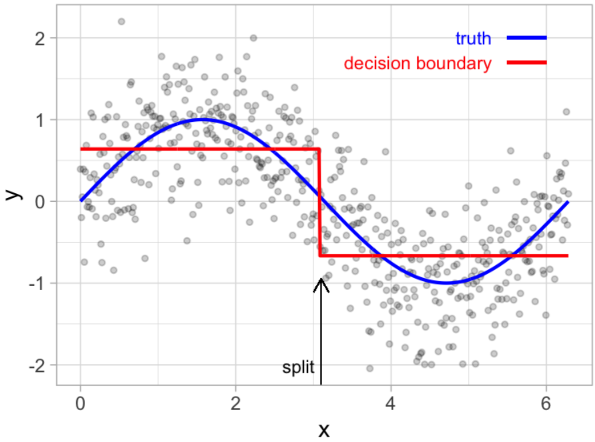
Decision tree



Source: HOML

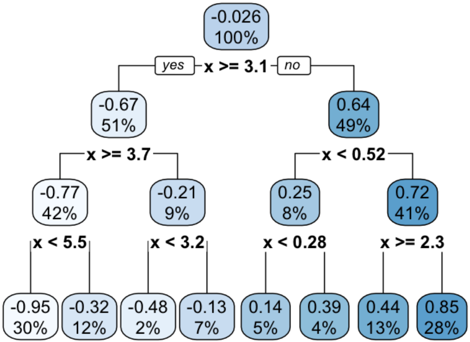
Decision tree with a single split

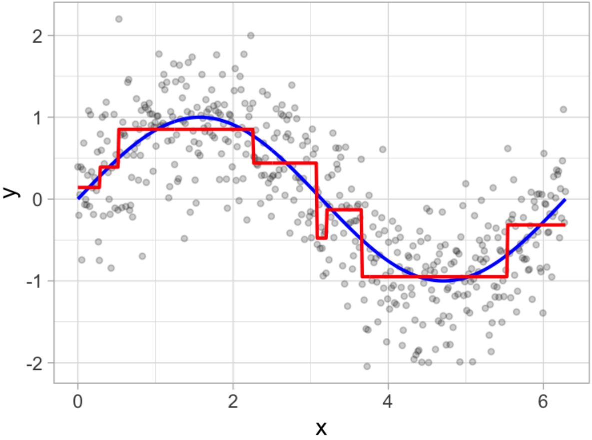




Source: HOML

Decision tree with depth=3

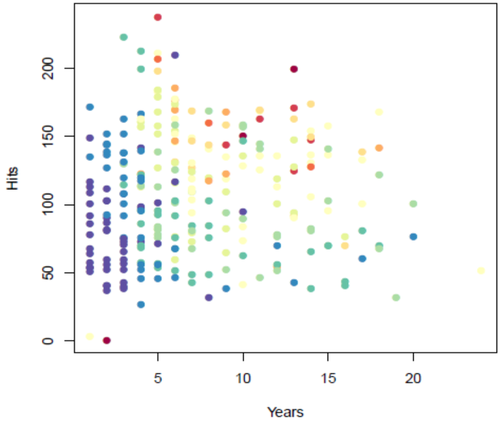




Source: HOML

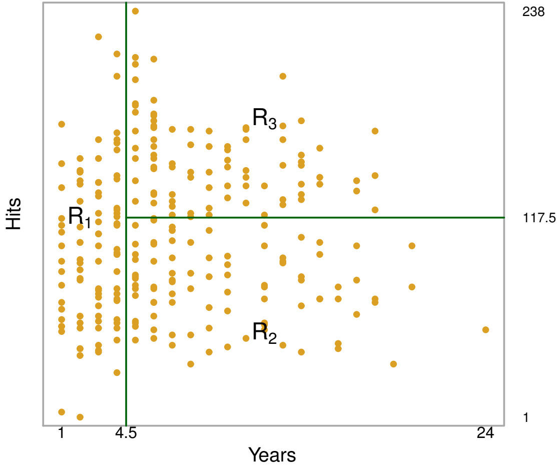
Stratifying baseball salary data

Salary is coded from low (blue, green) to high (yellow, red)



Source: ISL

Decision tree for baseball salary data



Source: ISL

- The tree stratifies players into three regions
- R1$= \{X|\texttt{Years} < 4.5\}$
- R2 $= \{X|\texttt{Years} >= 4.5, \texttt{Hits} < 117.5\}$
- R3 $= \{X|\texttt{Years} >= 4.5, \texttt{Hits} >= 117.5\}$

Decision tree for baseball salary data

Source: ISL

- We could represent this tree as a linear function, where each of the leaves corresponds to a product of dummy variables

- $x_1 = \mathbf{1}_{\texttt{Years}<4.5}$
- $x_2 = \mathbf{1}_{\texttt{Years}>=4.5} \times \mathbf{1}_{\texttt{Hits}<117.5}$
- $x_2 = \mathbf{1}_{\texttt{Years}>=4.5} \times \mathbf{1}_{\texttt{Hits}>=117.5}$

Interpreting the tree

- `Years` is the most important factor in determining salary.
- Players with less experience earn lower salaries than more experienced players.
- Among players who have been in the major leagues for five or more years, the number of `Hits` made in the previous year does affect salary, and players who made more `Hits` last year tend to have higher salaries.

Growing the tree

- Growing a tree consists of two main steps:
  1. Stratifying the feature space into J regions
  2. Making predictions $\hat{y}_{R_j}$ using the mean outcome of a given region R_j:

$$\hat{y}_{R_j} = \frac{1}{n_j} \sum_{i \in R_j} y$$

- For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

Growing the tree

- The regions are chosen by minimizing the RSS across all J regions

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

, where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

Growing the tree

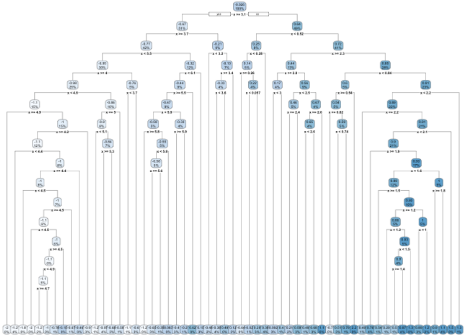
- **Problem:** It is computationally infeasible to consider every possible partition of the feature space into J regions.
- **Solution:** Take a top-down, greedy approach that is known as recursive binary splitting.
  - recursive: start with the best split, then find the next best split
  - binary: each split creates two branches
  - greedy: the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

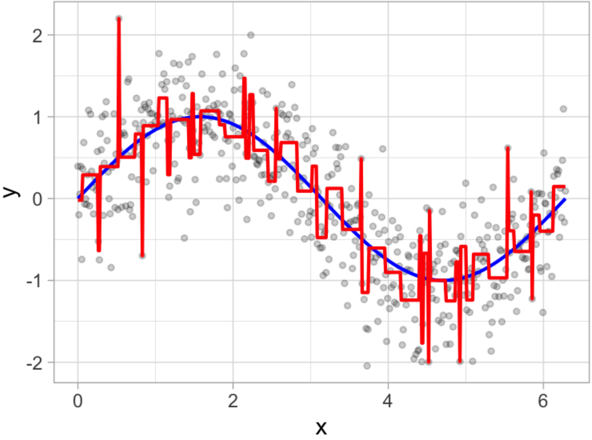
Growing the tree

- We first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $X_j < s$ and $X_j >= s$ leads to the greatest possible reduction in RSS.
- Once we make the split, we then continue splitting, conditional on the regions from the previous splits
- So if our first split creates R_1 and R_2, then our next split searches the predictor space only in R_1 or R_2
- We define a stopping criteria, e.g., at most 5 observations in each leaf, and the tree continues to grow until it hits this criteria

How deep?

If we grow an overly complex tree, we tend to overfit to our training data resulting in poor generalization performance.





Source: HOML

Pruning

- The tree building process can result in too many splits
- This will increase flexibility, hence lead to overfitting and reduce interpretability
- A smaller tree with fewer splits (that is, fewer regions) might lead to lower variance and better interpretation at the cost of a little bias

Pruning

- The solution lies in regularization and pruning the tree
- Grow a very large tree T_0, and then prune it back in order to obtain a subtree
- This is called cost complexity pruning

Pruning

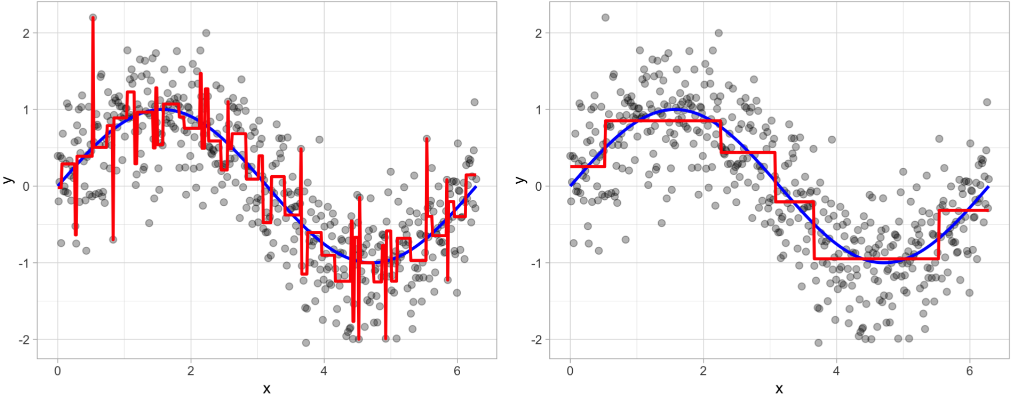
- Just like the Lasso, cost complexity pruning forces the tree to pay a price (penalty) α to become more complex
- We define complexity here as the number of regions or terminal nodes of the tree $|T|$

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

- For any value of α, we get a subtree $T \subset T_0$
- For $\alpha = 0$ we have T_0, as we increase α we start pruning the tree
- We choose $\hat{\alpha}$ via cross validation
- We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$

Pruning

Pruning involves growing an overly complex tree (left) and then using a cost complexity parameter to identify the optimal subtree (right).



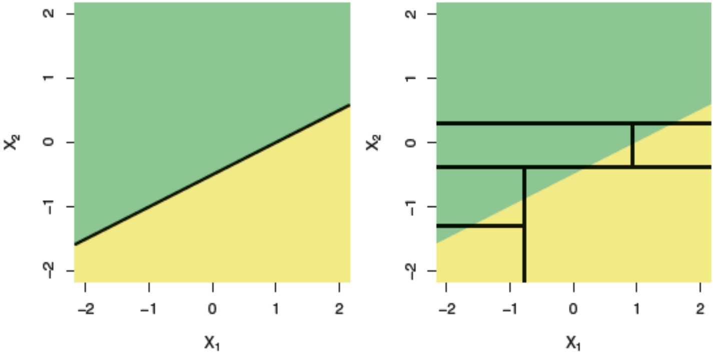
Source: HOML

Recap: tree building algorithm

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. For each $k = 1 \cdots K$:
   3.1 Repeat Steps 1 and 2 on the $\frac{K-1}{K}$th fraction of the training data, excluding the kth fold.
   3.2 Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α. Average the results, and pick α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

Tree-based models vs. linear models

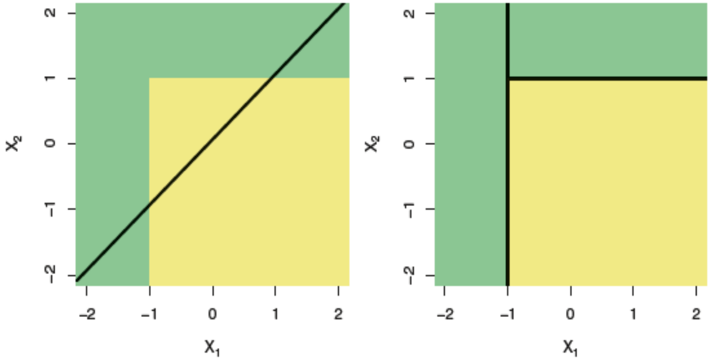
- When the true relationship is linear, trees do not perform well.



Source: ISL

Tree-based models vs. linear models

- Trees perform great when the true model is non-linear.



Source: ISL

Pros and cons of trees

Pros

- Easy to interpret
- Graphical representation
- Can handle categorical variables without the need to creating dummies
- Can capture complex relationships and nonlinearities in the data

Cons

- Predictive power is inferior to other methods
- Performs bad if the true model is linear
- Relatively non-robust, i.e. small changes in the data can cause big changes in the final tree

Later we'll see forests, which have been introduced to remedy for some of the weaknesses of trees.

References

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani (2017)
An Introduction to Statistical Learning
Springer.
https://www.statlearning.com/

Ed Rubin (2020)
Economics 524 (424): Prediction and Machine-Learning in Econometrics
Univ, of Oregon.

Bradley Boehmke, Brandon Greenwell (2020)
Hands-On Machine Learning with R
Taylor & Francis Group
https://bradleyboehmke.github.io/HOML/