RECSM Summer School: Machine Learning for Social Sciences

Session 2.1: Introduction to Classification and Regression Trees

Reto Wüest

Department of Political Science and International Relations
University of Geneva

The Basics of Decision Trees

The Basics of Decision Trees

- Tree-based methods stratify or segment the predictor space into a number of simple regions.
- To make a prediction for a test observation, we use the mean or mode of the training observations in the region to which it belongs.
- These methods are called decision-tree methods because the splitting rules used to segment the predictor space can be summarized in a tree.
- Decision trees can be applied to both regression and classification problems.

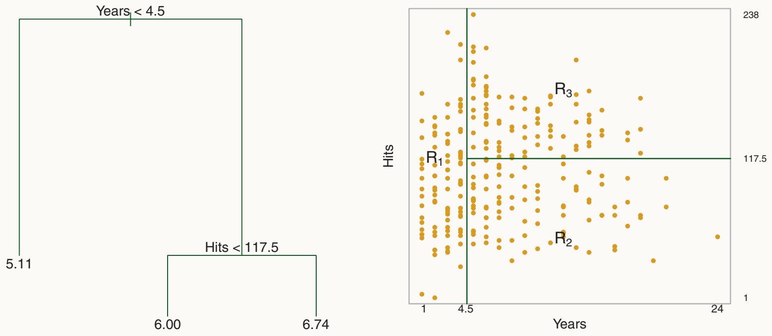
The Basics of Decision Trees

Regression Trees

Regression Trees – Example

The goal is to predict a baseball player's (log) salary based on the number of years played in the major leagues and the number of hits in the previous year.

Regression Tree Fit to Baseball Salary Data



(Source: James et al. 2013, 304f.)

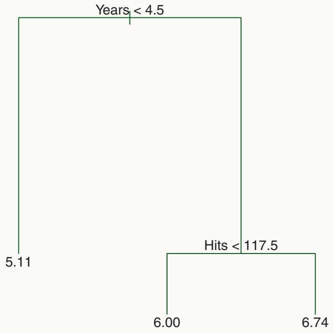
Terminology for Trees

- Regions R_1, R_2, and R_3 above are the terminal nodes or leaves of the tree.
- Points along the tree where the predictor space is split are the internal nodes (indicated above by the text `Years < 4.5` and `Hits < 117.5`).
- Segments of the tree that connect the nodes are called branches.

Interpretation of Trees

- Experience is the most important factor determining salary: players with less experience earn lower salaries than players with more experience.
- Among less experienced players, the number of hits matters little for a player's salary.
- Among more experienced players, those with a higher number of hits tend to have higher salaries.

Regression Tree Fit to Baseball Salary Data



Building a Regression Tree

Roughly speaking, there are two steps:

1. Divide the predictor space (i.e., the set of possible values for predictors $X_1, X_2, \ldots, X_p$) into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.
2. Make the same prediction for every test observation that falls into region R_j: the prediction is the mean of the response values of the training observations in R_j.

Building a Regression Tree

Step 1 (more detailed):

- How do we construct the regions $R_1, \ldots, R_J$?
- We divide the predictor space into high-dimensional rectangles (boxes), regions $\{R_j\}_{j=1}^{J}$, such that they minimize the RSS

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \tag{2.1.1}$$

where $\hat{y}_{R_j}$ is the mean response of the training observations in the jth box.

Building a Regression Tree

Step 1 (more detailed):

- It is computationally not feasible to consider every possible partition of the predictor space into J boxes.
- Therefore, we take a top-down, greedy approach that is known as recursive binary splitting:
    - Top-down: we begin at the top of the tree (where all observations belong to a single region) and successively split the predictor space;
    - Greedy: at each step of the tree-building process we make the split that is best at that step (i.e., we do not look ahead and pick a split that will lead to a better tree in some future step).

Building a Regression Tree

Step 1 (more detailed):

- How do we perform recursive binary splitting?
- We first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X \mid X_j < s\}$ and $\{X \mid X_j \geq s\}$ leads to the greatest possible reduction in RSS. (We now have two regions.)
- Next, we again select the predictor and the cutpoint that minimize the RSS, but this time we split one of the two previously identified regions. (We now have three regions.)

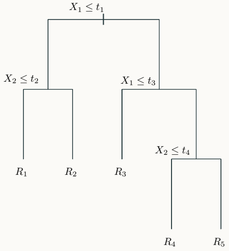
Building a Regression Tree

Step 1 (more detailed):

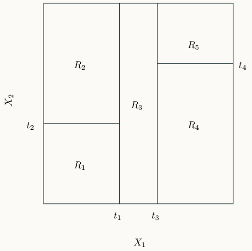
- Next, we split one of the three regions further, so as to minimize the RSS. (We now have four regions.)
- We continue this process until a stopping criterion is reached.
- Once the regions $R_1, \ldots, R_J$ have been created, we predict the response for a test observation using the mean of the training observations in the region to which the test observation belongs.

Building a Regression Tree – Example

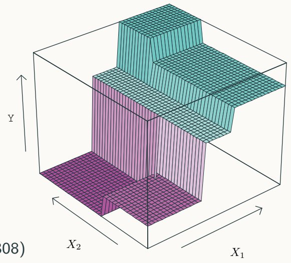
Decision Tree



Predictor Space



Prediction Surface



(Source: James et al. 2013, 308)

The Basics of Decision Trees

Tree Pruning

Tree Pruning

- The above process may produce good predictions on the training set, but it is likely to overfit the data, leading to poor test set performance.
- The reason is that the resulting tree might be too complex. A less complex tree (fewer splits) might lead to lower variance at the cost of a little bias.
- A less complex tree can be achieved by tree pruning: grow a very large tree T_0 and then prune it back in order to obtain a subtree.

Tree Pruning

- How do we find the best subtree?
- Our goal is to select a subtree that leads to the lowest test error rate.
- For each subtree, we could estimate its test error using cross-validation (CV).
- However, this approach is not feasible as there is a very large number of possible subtrees.
- Cost complexity pruning allows us to select only a small set of subtrees for consideration.

Cost Complexity Pruning

- Let $\alpha \geq 0$ be a tuning parameter. For each value of α, there is a subtree $T \subset T_0$ that minimizes

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|, \tag{2.1.2}$$

  where $|T|$ is the number of terminal nodes of subtree T.
- The tuning parameter α controls the trade-off between the subtree's complexity and its fit to the training data.
- The price we need to pay for having a tree with many terminal nodes increases with α. Hence, (2.1.2) will be minimized for a smaller subtree. (Note the similarity to the lasso!)

Cost Complexity Pruning

- We can select the optimal value of α using CV (or, in a data-rich situation, the validation set approach).
- Finally, we return to the full data set and obtain the subtree corresponding to the optimal value of α.

Cost Complexity Pruning

Algorithm: Fitting and Pruning a Regression Tree

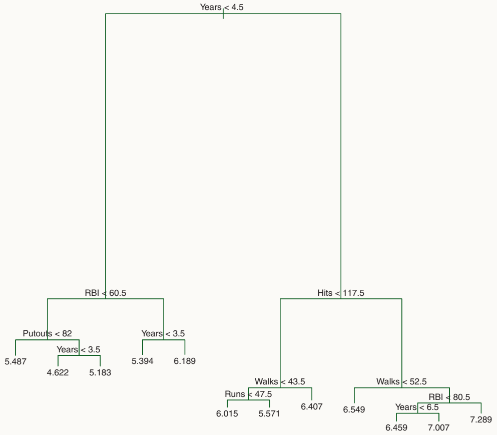
1. Use recursive binary splitting to grow a large tree T_0 on the training data.
2. Apply cost complexity pruning to T_0 in order to obtain a sequence of best subtrees, as a function of α.
3. Use K-fold CV to choose the optimal α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:
   (a) Repeat Steps 1 and 2 on all but the kth fold of the training data.
   (b) Evaluate the 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 choose α to minimize the average error.
4. Return the subtree from Step 2 that corresponds to the chosen value of α.

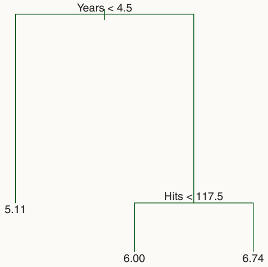
Cost Complexity Pruning – Example

Fitting and Pruning a Regression Tree on the Baseball Salary Data

Unpruned Tree



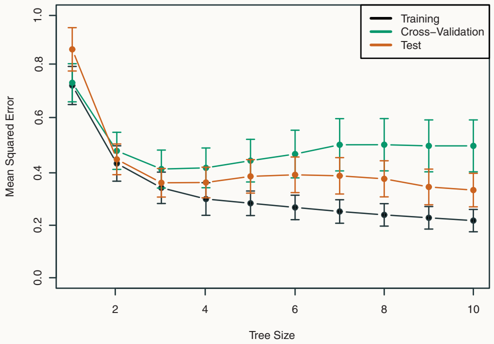
Pruned Tree (for optimal α)



(Source: James et al. 2013, 304 & 310)

Cost Complexity Pruning – Example

Fitting and Pruning a Regression Tree on the Baseball Salary Data



(Green curve shows the CV error associated with α and, therefore, the number of terminal nodes; orange curve shows the test error; black curve shows the training error curve. Source: James et al. 2013, 311)

The CV error is a reasonable approximation of the test error. The CV error takes on its minimum for a three-node tree (see previous slide).

The Basics of Decision Trees

Classification Trees

Classification Trees

- Classification trees are very similar to regression trees, except that they are used to predict a qualitative rather than a quantitative response.
- For a regression tree, the predicted response for an observation is given by the mean response of the training observations that belong to the same terminal node.
- For a classification tree, the predicted response for an observation is the most commonly occurring class among the training observations that belong to the same terminal node.

Building a Classification Tree

- Just as in the regression setting, we use recursive binary splitting to grow a classification tree.
- However, in the classification setting, RSS cannot be used as a criterion for making binary splits. Alternatively, we could use the classification error rate.
- We would assign each observation in terminal node m to the most commonly occurring class, so the classification error rate is the fraction of training observations in that terminal node that do not belong to the most common class

$$E = 1 - \arg\max_k(\hat{p}_{mk}), \tag{2.1.3}$$

where $\hat{p}_{mk}$ represents the proportion of training observations in the mth terminal node that are from the kth class.

Building a Classification Tree

- However, it turns out that classification error is not sufficiently sensitive for tree-growing.
- Therefore, two other measures are preferable: the Gini index and entropy.
- The Gini index is a measure of total variance across the K classes:

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}). \tag{2.1.4}$$

  It takes on a small value if all of the $\hat{p}_{mk}$'s are close to 0 or 1. Therefore, a small value indicates that a node contains predominantly observations from a single class (node purity).

Building a Classification Tree

- An alternative to the Gini index is the entropy, given by

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}. \tag{2.1.5}$$

  (Note that since $0 \leq \hat{p}_{mk} \leq 1$, it is $0 \leq -\hat{p}_{mk} \log \hat{p}_{mk}$.)
- The entropy will take on a value near 0 if the $\hat{p}_{mk}$'s are all near 0 or 1. Therefore, like the Gini index, the entropy will take on a small value if the mth node is pure.

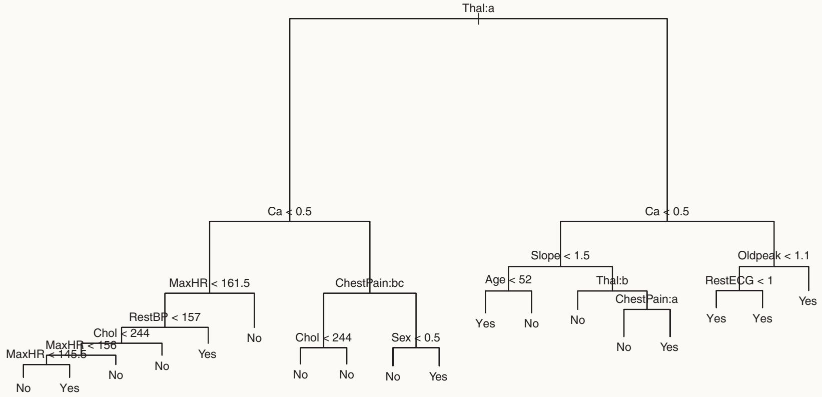
Building a Classification Tree

- **Building a classification tree**: either the Gini index or the entropy is used to evaluate the quality of a particular split, since these measures are more sensitive than the classification error rate.
- **Pruning the tree**: any of the three measures might be used, but the classification error rate is preferable if prediction accuracy of the final tree is the goal.

Building a Classification Tree – Example

Fitting and Pruning a Classification Tree on Heart Disease Data

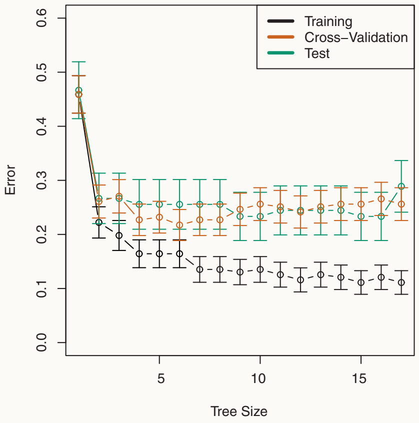
Data for 303 patients with chest pain. Output variables takes a value of Yes if a patient has a heart disease and a value of No if the patient has no heart disease. There are 13 input variables.

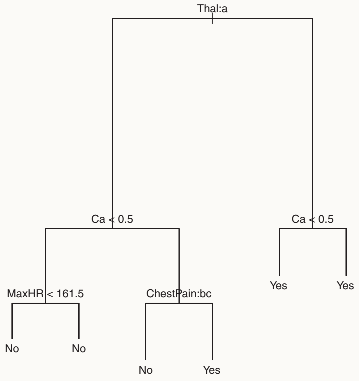


(Source: James et al. 2013, 313)

Building a Classification Tree – Example

Fitting and Pruning a Classification Tree on the Heart Disease Data





(Source: James et al. 2013, 313)

Building a Classification Tree – Example

- Note that in the above example, some of the splits yielded two terminal nodes that have the same predicted value.
- Why are these splits performed at all?
- Such splits lead to increased node purity (they do not reduce the classification error, but they improve the Gini index and the entropy, which are more sensitive to node purity).
- Node purity is important because it tells us something about how certain we are when making a prediction.