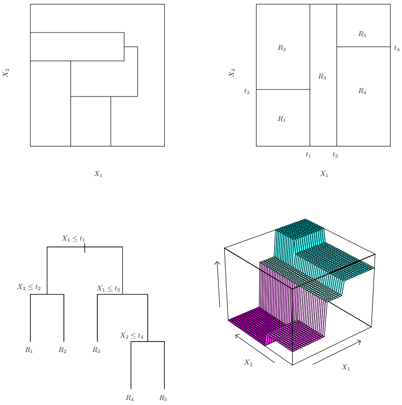
Lecture 19: Decision trees

Reading: Section 8.1

STATS 202: Data mining and analysis

November 6, 2019

Decision trees, 10,000 foot view

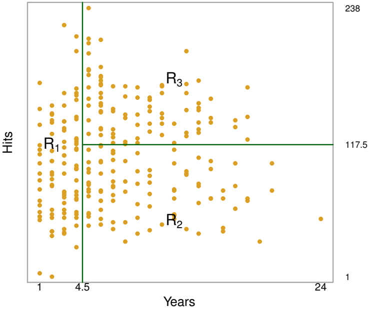


1. Find a partition of the space of predictors.
2. Predict a constant in each set of the partition.
3. The partition is defined by splitting the range of one predictor at a time.

   $\rightarrow$ Can be represented as a decision tree.

   $\rightarrow$ Not all partitions are possible.

Example: Predicting a baseball player's salary



The prediction for a point in region R_i is the average of the training points in R_i.

How is a decision tree built?

- Start with a single region R_1 (entire input space), and iterate:

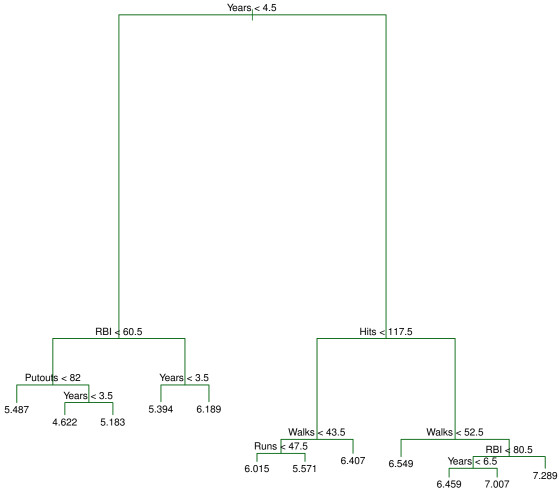
    1. Select a region R_k, a predictor X_j, and a splitting point s, such that splitting R_k with the criterion $X_j < s$ produces the largest decrease in RSS:

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

    2. Redefine the regions with this additional split.

- Terminate when there are 5 observations or fewer in each region (or use a different stopping criterion.)
- This grows the tree from the root towards the leaves (Top-down greedy approach).

A decision tree for baseball salaries



How do we control overfitting?

- **Idea 1:** Find the optimal subtree by cross validation.

  $\rightarrow$ There are too many possibilities, so we would still over fit.

- **Idea 2:** Stop growing the tree when the RSS doesn't drop by more than a threshold with any new cut.

  $\rightarrow$ In our greedy algorithm, it is possible to find good cuts after bad ones.

How do we control overfitting?

Solution: Prune a large tree from the leaves to the root.

- **Weakest link pruning:**
  - Starting with with the initial full tree T_0, replace a subtree with a leaf node to obtain a new tree T_1. Select subtree to prune by minimizing:
    $$\frac{RSS(T_1) - RSS(T_0)}{|T_0| - |T_1|}.$$
  - Iterate this pruning to obtain a sequence $T_0, T_1, T_2, \ldots, T_m$ where T_m is the tree with a single leaf node.
  - Select the optimal tree T_i by cross validation.

How do we control overfitting?

… or an equivalent procedure

- **Cost complexity pruning:**
  - Minimize the following objective over all prunings T of T_0:

$$\text{minimize} \sum_{R_m \in T} \sum_{x_i \in R_m} (y_i - \bar{y}_{R_m})^2 + \alpha |T|.$$

  - When $\alpha = \infty$, we select the null tree (=tree with one leaf node.)
  - When $\alpha = 0$, we select the full tree.
  - Fun fact: The solution for each α is among $T_1, T_2, \ldots, T_m$ from weakest link pruning.
  - Choose the optimal α (the optimal T_i) by cross validation.

Cross validation, the wrong way

1. Construct a sequence of trees $T_0, \ldots, T_m$ for a range of values of α.
2. Split the training points into 10 folds.
3. For $k = 1, \ldots, 10$,
   - For each tree T_i, use every fold except the kth to estimate the averages in each region.
   - For each tree T_i, calculate the RSS in the test fold.
4. For each tree T_i, average the 10 test errors, and select the value of α that minimizes the error.

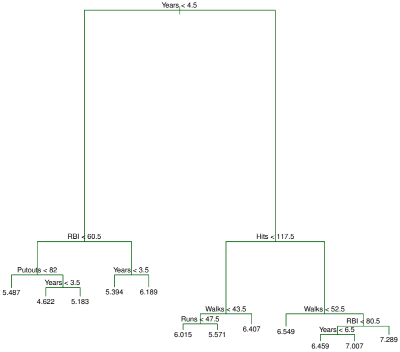
WRONG WAY TO DO CROSS VALIDATION!

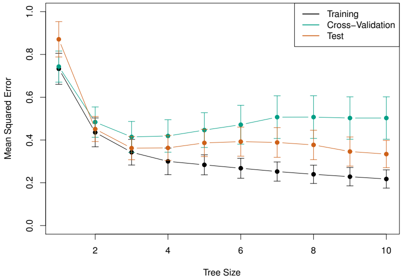
Cross validation, the right way

1. Split the training points into 10 folds.
2. For $k = 1, \ldots, 10$, using every fold except the kth:
   - Construct a sequence of trees $T_1, \ldots, T_m$ for a range of values of α, and find the prediction for each region in each one.
   - For each tree T_i, calculate the RSS on the test set.
3. Select the parameter α that minimizes the average test error.

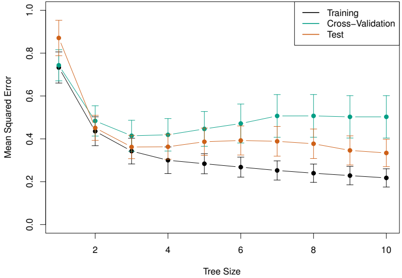
Note: We are doing all fitting, **including the construction of the trees**, using only the training data.

Example. Predicting baseball salaries





Example. Predicting baseball salaries



Classification trees

- They work much like regression trees.
- We predict the response by **majority vote**, i.e. pick the most common class in every region.
- Instead of trying to minimize the RSS:

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

we minimize a classification loss function.

Classification losses

- The 0-1 loss or misclassification rate:

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} \mathbf{1}(y_i \neq \hat{y}_{R_m})$$

- The Gini index:

$$\sum_{m=1}^{|T|} q_m \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

  where $\hat{p}_{m,k}$ is the proportion of class k within R_m, and q_m is the proportion of samples in R_m.

- The cross-entropy:

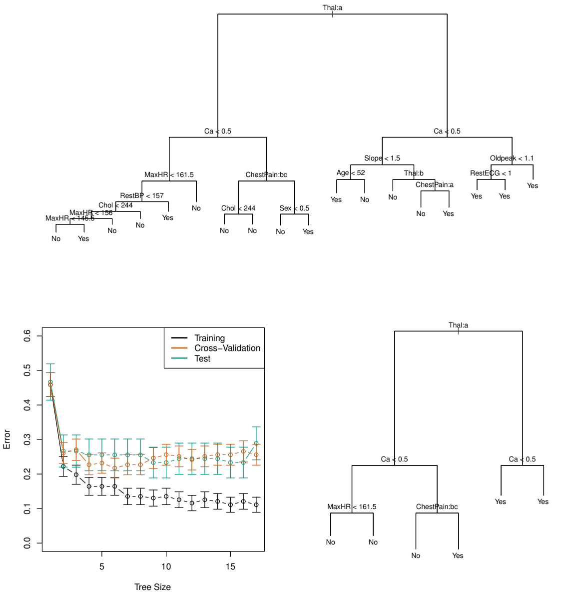
$$-\sum_{m=1}^{|T|} q_m \sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk}).$$

Classification losses

- The Gini index and cross-entropy are better measures of the purity of a region, i.e. they are low when the region is mostly one category.
- **Motivation for the Gini index:**

  If instead of predicting the most likely class, we predict a random sample from the distribution $(\hat{p}_{1,m}, \hat{p}_{2,m}, \ldots, \hat{p}_{K,m})$, the Gini index is the expected misclassification rate.
- It is typical to use the Gini index or cross-entropy for growing the tree, while using the misclassification rate when pruning the tree.

Example. Heart dataset.



Some advantages of decision trees

- Very easy to interpret!
- Closer to human decision-making.
- Easy to visualize graphically.
- They easily handle qualitative predictors and missing data.