Classification and regression trees

Patrick Breheny

December 9

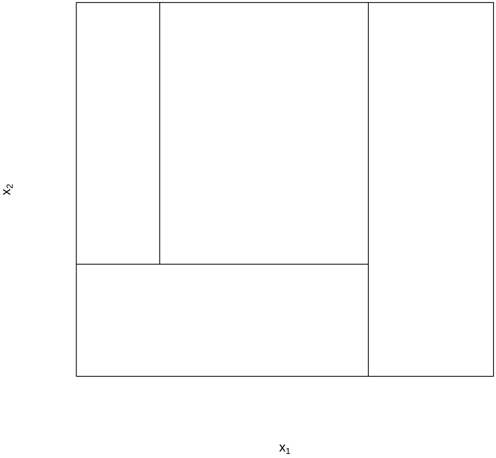
Introduction

- We've seen that local methods and splines both operate by partitioning the sample space of the regression variable(s), and then fitting separate/piecewise models in each partition
- This partitioning occurs in a prespecified way – the data does not guide the partitioning
- Another possibility is to use the data to actively seek partitions which improve the fit as much as possible
- This is the main idea behind *tree-based methods*, which recursively partition the sample space into smaller and smaller rectangles

Recursive partitioning

- To see how this works, consider a linear regression problem with a continuous response y and two predictors x_1 and x_2
- We begin by splitting the space into two regions on the basis of a rule of the form $x_j \leq s$, and modeling the response using the mean of y in the two regions
- The optimal split (in terms of reducing the residual sum of squares) is found over all variables j and all possible split points s
- The process is then repeated in a recursive fashion for each of the two sub-regions

Partitioning illustration



The regression model

- This process continues until some stopping rule is applied
- For example, letting $\{R_m\}$ denote the collection of rectangular partitions, we might continue partitioning until $|R_m| = 10$
- The end result is a piecewise constant model over the partition $\{R_m\}$ of the form

$$f(\mathbf{x}) = \sum_m c_m I(\mathbf{x} \in R_m)$$

where c_m is the constant term for the mth region (*i.e.*, the mean of y_i for those observations $\mathbf{x}_i \in R_m$)

Trees

- The model can be neatly expressed in the form of a tree
- The regions $\{R_m\}$ are then referred to as the *terminal nodes* of the tree
- The non-terminal nodes are referred to as *interior nodes*
- The splits are variously referred to as "splits", "edges", or "branches"
- We'll see some pictures of these trees later in lecture

Categorical data

- The same idea can be used – and is in fact more straightforward to implement – when the predictors are categorical
- The same idea can also be used when the outcome is categorical
- In that case, we fit a simple model in each region R_m, predicting the outcome based on the observed sample proportions in R_m
- Trees for continuous outcomes are called *regression trees*, while trees for categorical outcomes are called *classification trees*

The bias-variance tradeoff

- How large should we grow our tree?
- A large tree will fragment the data into smaller and smaller samples and result in overfitting
- On the other hand, a small tree might not capture the important relationships among the variables

Pruning

- The most common approach is to grow a large tree, and then *prune* the tree to a size that seems to balance fitting vs. overfitting
- Denote the large tree T_0, and define a subtree $T \subset T_0$ as a tree that can be obtained by collapsing any number of its internal nodes
- We then define the *cost-complexity criterion*:

$$C_\alpha(T) = L(T) + \alpha|T|,$$

where $L(T)$ is the loss associated with tree T, $|T|$ is the number of terminal nodes (parameters) in tree T, and α is a tuning parameter that controls the tradeoff between the two

Choosing α

- We can build a sequence of subtrees using *weakest link pruning*: starting with T_0, at each step we collapse the internal node that produces the smallest increase in the loss function
- It can be shown that this sequence contains the subtree that minimizes $C_\alpha(T)$ for all α
- Estimation of α can then be achieved via cross-validation

Loss functions

- For continuous outcomes, squared error loss is by far the most common
- For categorical outcomes, the natural loss function to use is the deviance:

$$L(T) = -\sum_m \sum_{x_i \in R_m} \sum_k \hat{p}_{mk} \log \hat{p}_{mk}$$

- However, misclassification (0-1) loss is also used, as is the *Gini index*:

$$L(T) = \sum_m \sum_{x_i \in R_m} \sum_k \hat{p}_{mk}(1 - \hat{p}_{mk})$$

Fitting tree-based models in R

- There are many R packages that implement classification and regression trees and variations and extensions thereof
- The one that ships with base R is called `rpart`
- Other packages that have been developed since `rpart` have made the effort to preserve the syntax, and so all of them fit models using some variation of:

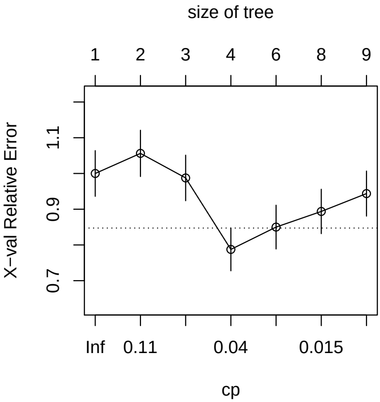
  ```
  fit0 <- rpart(y~.,data=mydata)
  ```

  which fits a tree using the supplied data frame, treating one variable as the outcome and the rest as predictors
- If `y` is numeric, a regression tree is fit; if `y` is a character or factor, a classification tree is fit

Cost-complexity pruning

Cross-validation is performed automatically; its results are available in `fit0$cptable` and can be plotted via

```
plotcp(fit0)
```



Cost-complexity pruning (cont'd)

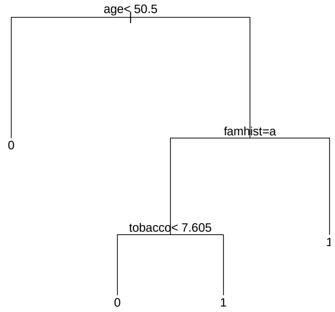
We can prune the tree via:

```
cptable <- as.data.frame(fit0$cptable)
alpha <- cptable$CP[which.min(cptable$xerror)]
fit <- prune(fit0,cp=alpha)
```

Plotting the tree

We can then plot the tree with

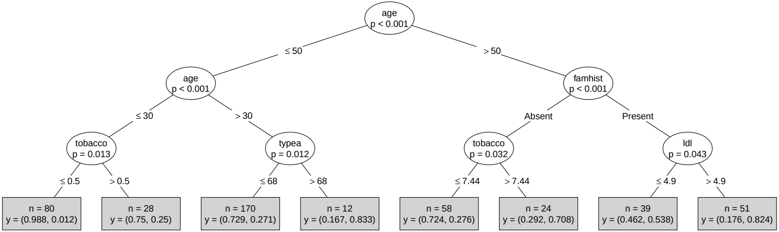
```
plot(fit)
text(fit)
```



Another approach

The party package provide a different approach, with much more attractive plotting methods:

```
fit <- ctree(chd~.,data=heart)
plot(fit,type="simple")
```



Regression trees: Pros

- Recursive partitioning models have two major advantages: interpretability and modeling of interactions
- The model tree obtained in the end is one of the easiest-understood ways to convey a model to a non-statistician
- Furthermore, they are among very few methods that have been developed that are capable of automatically modeling interactions without becoming computationally and statistically infeasible
- Other pros: handle missing data without difficulty

Regression trees: Cons

- They also have two big disadvantages: instability and difficulty capturing simple relationships
- Trees tend to have high variance, in the sense that a very small change in the data can produce a very different series of splits
- One reason for this is that any change at an upper level of the tree is propagated down the tree and affects all other splits
- Furthermore, regression trees require a large number of parameters (splits) to capture simple models such as linear and additive relationships
- Other cons: lack of inferential methods, lack of smoothness