Classification trees

Patrick Breheny

November 10

Introduction

- Our discussion of tree-based methods in the previous section assumed that the outcome was continuous
- Tree-based methods for categorical outcomes are referred to as *classification trees*
- The main idea is the same: we recursively partition the sample space, fitting a very simple model in each partition
- In the case of classification, this our model is to simply use the observed proportions

Classification tree model

- The classification model is simply to estimate $\Pr(G = k|x)$ by

$$\sum_m \hat{\pi}_{mk} I(\mathbf{x} \in R_m),$$

  where $\hat{\pi}_{mk}$ is the proportion of observations in partition m that belong to class k
- Regression trees are based on the least squares criterion: we used this criterion both to find optimal splits and to measure node impurity for the purpose of pruning
- This criterion is not appropriate for classification
- The primary change we need to make in the algorithm in order to obtain classification trees is to establish a new criterion to determine goodness of fit

Node impurity measures

There are three commonly used measures of node impurity $Q_m(T)$ for classification trees:

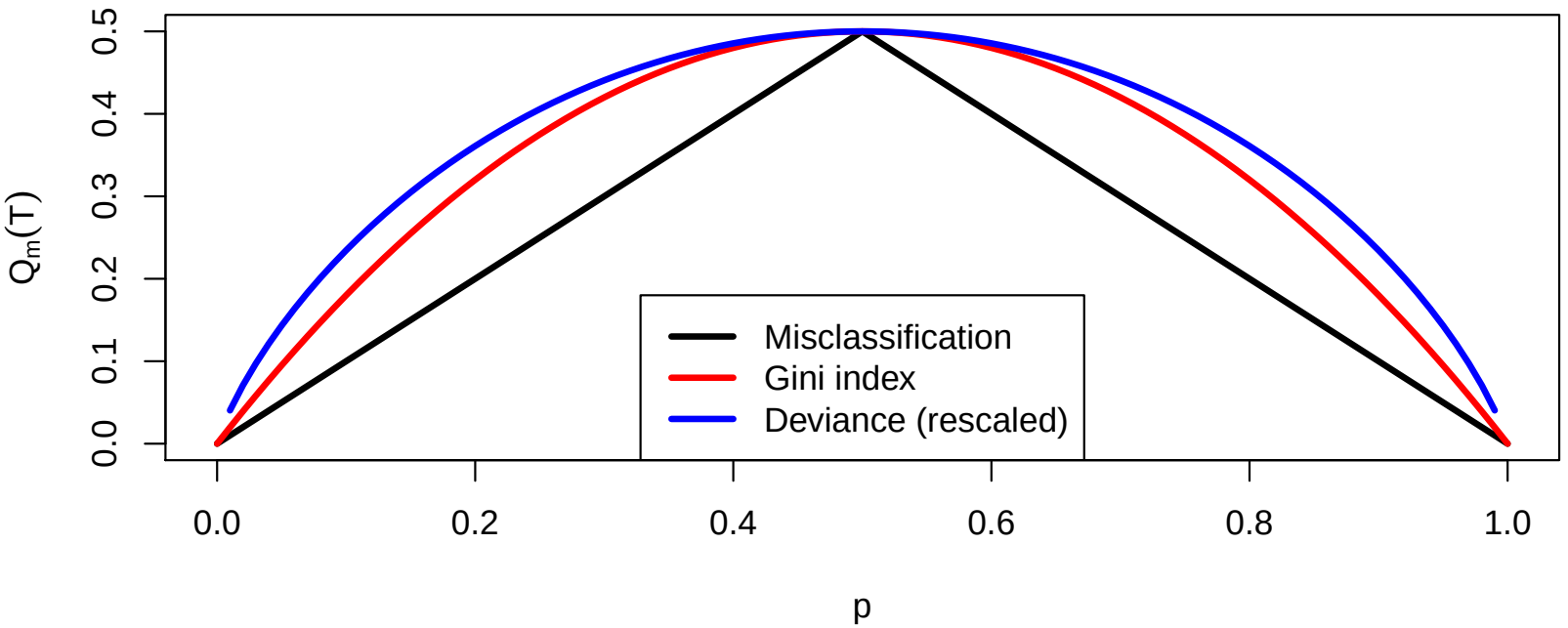
Misclassification error: $\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq \arg\max_k \hat{\pi}_{mk})$

Gini index: $\sum_k \hat{\pi}_{mk}(1 - \hat{\pi}_{mk})$

Deviance: $-\sum_k \hat{\pi}_{mk} \log(\hat{\pi}_{mk})$

Node impurity measures (cont'd)

For $K = 2$:



Note that all three are similar, but that the Gini index and deviance are differentiable, and thus easier to optimize numerically

Growing and pruning your classification tree

- As with regression trees, the classification tree algorithm determines, for each variable j, the split point s_j that minimizes

$$N_1 Q_1(s_j) + N_2 Q_2(s_j),$$

  where N_i is the number of observations in each new node and $Q_i(s_j)$ is the impurity measure of each new node
- Again, we search over all (j, s) pairs and choose the one that provides the lowest total impurity $\sum_m N_m Q_m(T)$
- Cost-complexity pruning works the same way, although typically misclassification loss is used for pruning regardless of the method used to grow the tree, as it is the easiest to implement via cross-validation

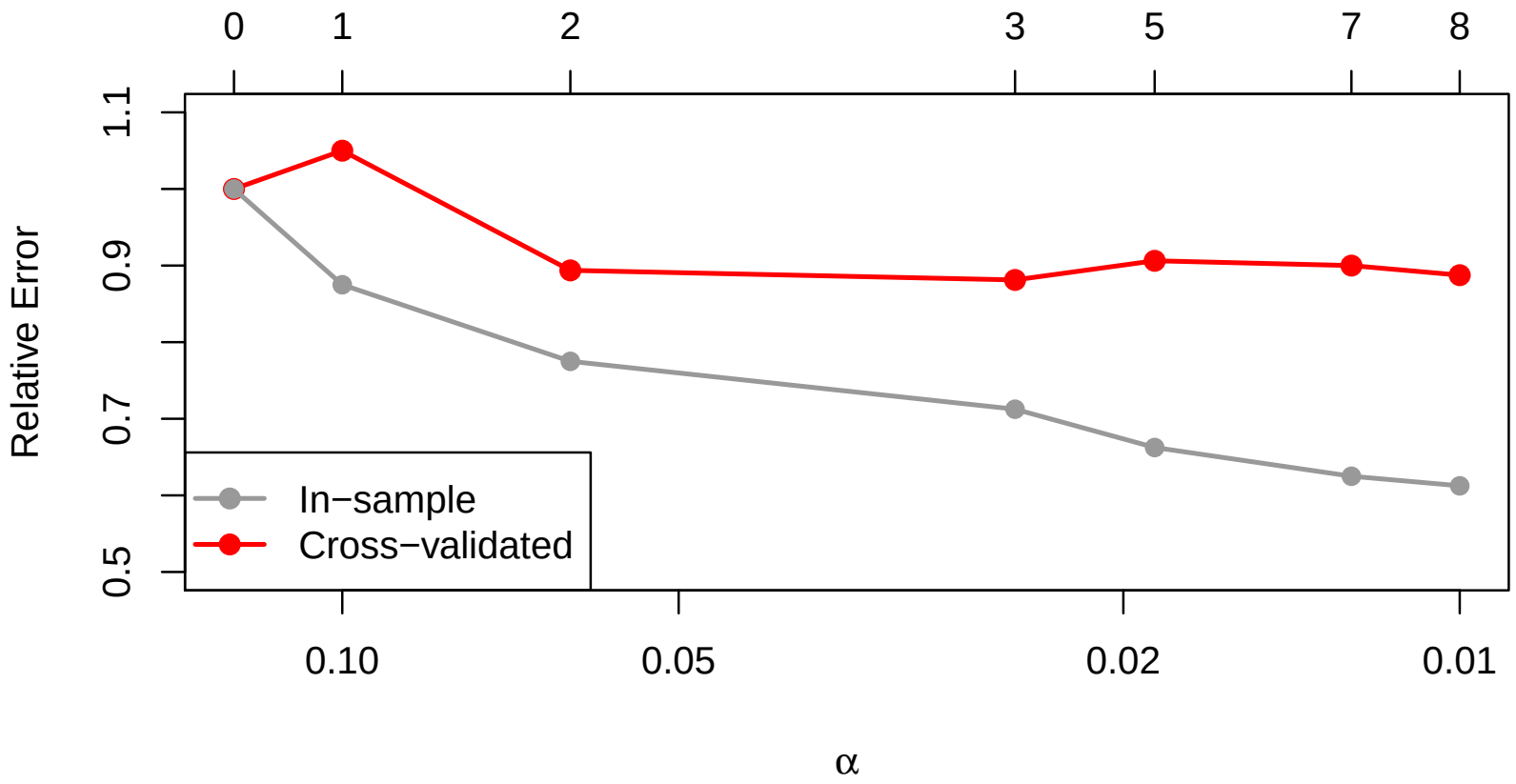
Example

- As an example of classification trees, we will consider the coronary heart disease data set that we have examined a few times already
- To obtain classification trees, one can either specify `method='class'` or convert the outcome to a factor, in which case `method='class'` is inferred:

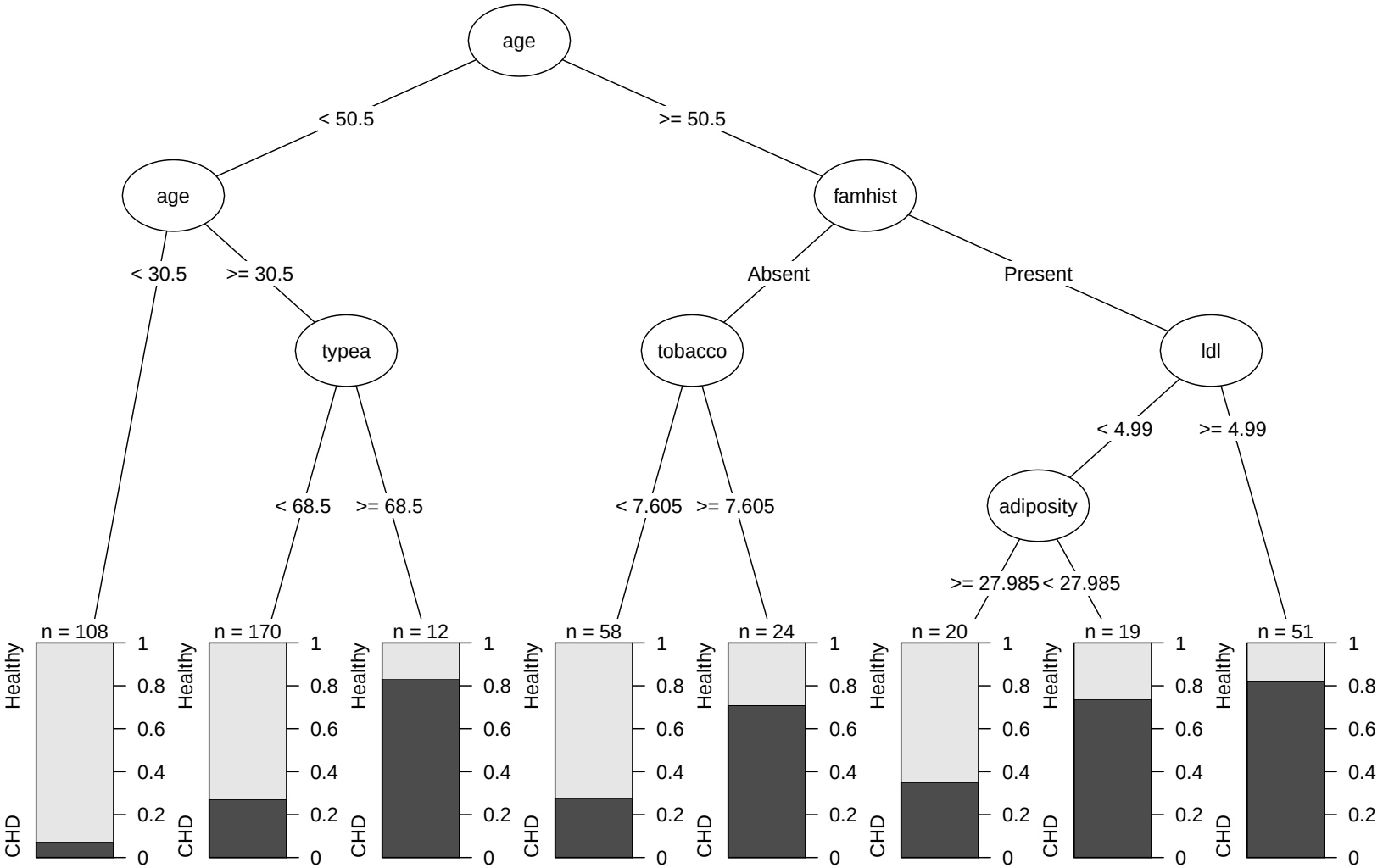
```
heart$chd <- factor(heart$chd,labels=c("Healthy","CHD"))
fit0 <- rpart(chd~.,data=heart)
```

- This method of inferring the type of tree by the class of the outcome is also used by `party`, which lacks the option to directly specify the model
- By default, `rpart` uses the Gini index to grow classification trees, although there is an option to use deviance instead; `party` is again based on hypothesis testing

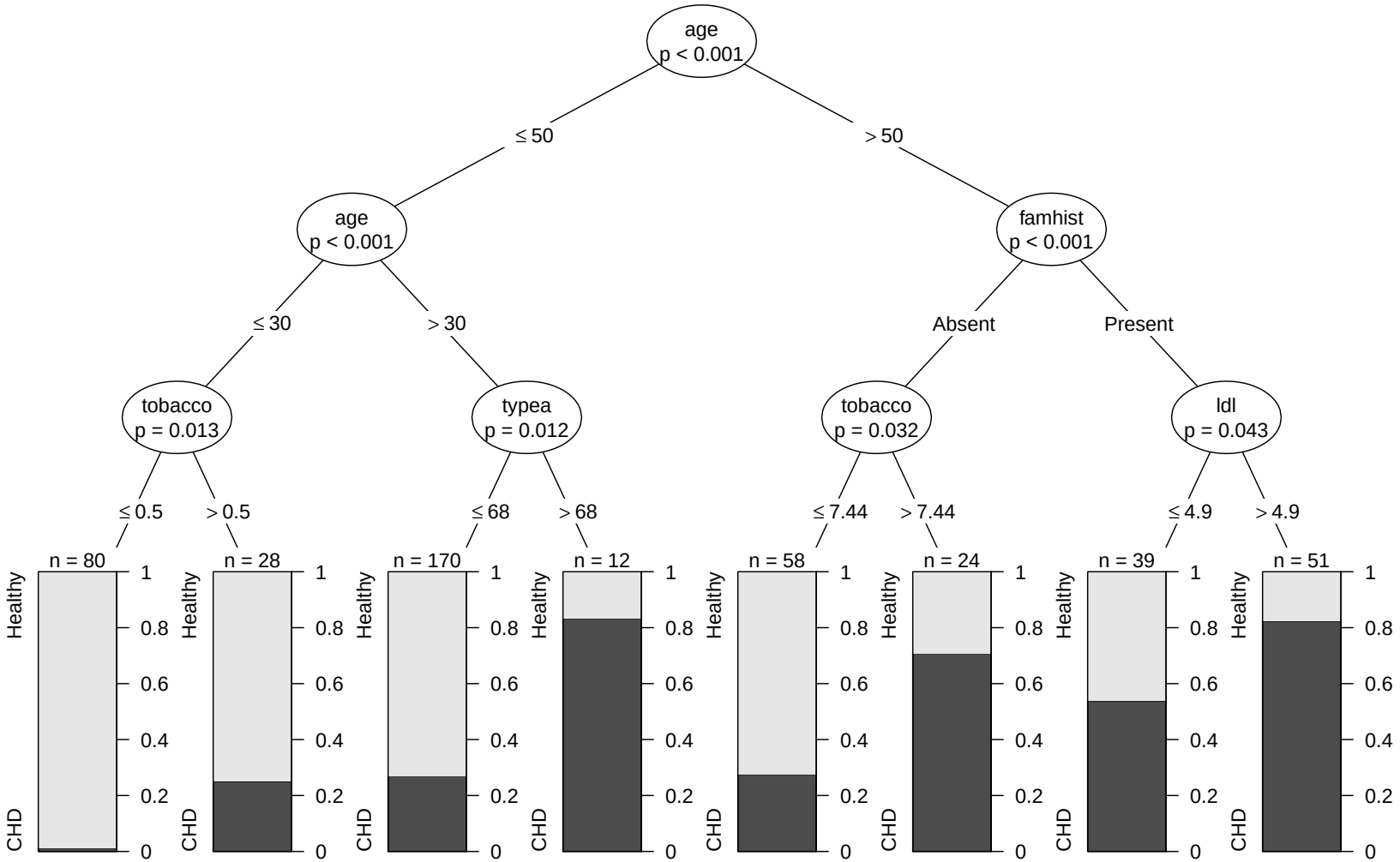
Cost-complexity pruning



Results: rpart



Results: party



Assumption-free

- Perhaps the primary advantage of tree-based methods is that they are virtually assumption-free
- Consequently, they are very simple to fit and interpret, since no time or effort has to go into making, checking, or explaining assumptions
- Furthermore, they are capable of discovering associations, such as higher-order interactions, that would otherwise go utterly unsuspected

Easily extended

- Another advantage is that, since very simple models are being fit to each node, trees are fairly easy to extend to other problems
- For instance, the `rpart` and `party` packages have implemented survival trees, poisson regression trees, and ordinal regression trees (some are only available in one package or the other, however)
- Indeed, `rpart` lets you pass your method to the function although admittedly, this is somewhat easier said than done

Missing data

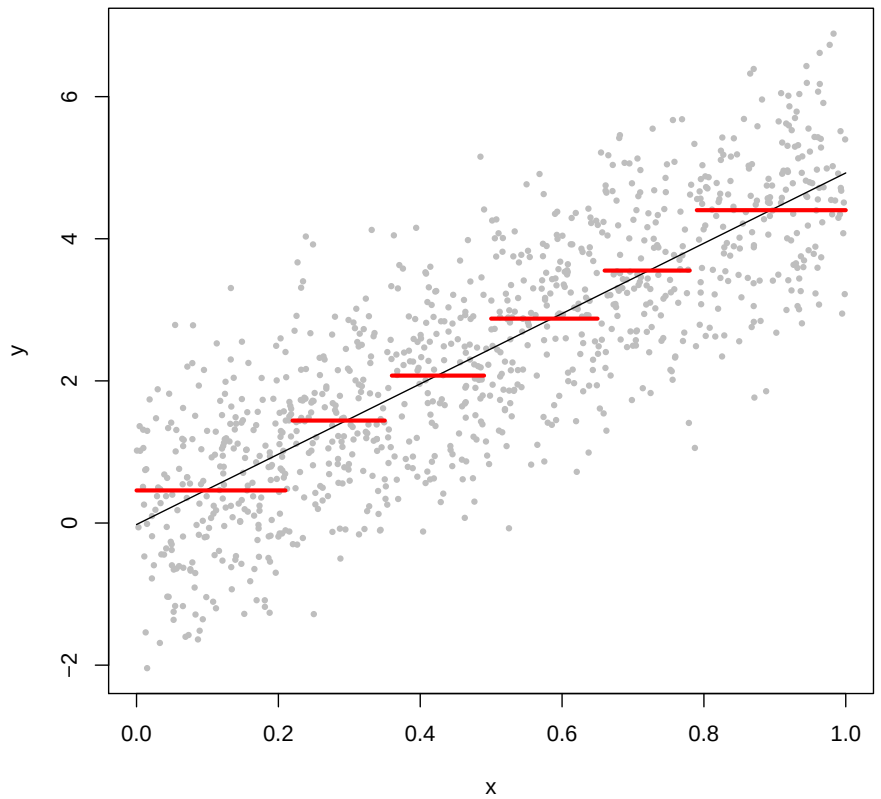
- Finally, trees have a rather elegant option for handling missing data besides the usual options of discarding or imputing observations
- For a given split, we can find *surrogate* splits, which best mimic the behavior of the original split
- Then, when sending an observation down the tree, if the splitting variable is missing, we simply use the surrogate split instead

Instability of trees

- The primary disadvantage of trees is that they are rather unstable (*i.e.*, have high variance)
- In other words, small change in the data often results in a completely different tree – something to keep in mind while interpreting trees
- One major reason for this instability is that if a split changes, all the splits under it are changes as well, thereby propagating the variability
- A related methodology, *random forests*, grows a large number of trees and then averages across them in order to stabilize the tree-based approach, although obviously there is a cost as far as interpretation is concerned

Difficulty in capturing additive structure

In addition, trees have a difficult time capturing simple additive structures:



Concluding remarks

- In other words, the weaknesses of tree-based methods are precisely the strengths of linear models, and vice versa
- For this reason, I have always personally found the two methods to be useful complements to each other
- For example, when embarking on an extensive analysis using a linear or generalized linear model, it doesn't hurt to check your results against a regression tree
- If you find that the regression tree chooses a very different set of important variables, you may want to reconsider the linear model (especially if the regression tree achieves a much better R^2)