STATISTICS 452/652: Statistical Learning and Prediction

November 11, 2020

Lecture 20: Tree-Based Classifiers

(Reading: Sections 8.1.2, 8.2)

1 Goals of lecture

1. We have seen that regression trees can automatically mimic any structure, given enough data.
2. With limited data, *ensembles* of trees—random forest and boosting—are usually vast improvements over individual trees
3. We will learn how to adapt trees and ensembles to perform classification

2 Classification Trees

Breiman et al. (1984) introduced both regression trees and classification trees in their book, *Classification and Regression Trees*, which has become abbreviated as "CART." Most of that book is about classification, rather than regression, because it's a much trickier problem.

- Exact same idea as regression trees
    1. Define an optimization criterion (RSS in regression)
    2. Look for best splits within each explanatory variable
        (a) "Best" means reducing criterion the most.
    3. Recursively partition the data into smaller nodes
    4. Prune tree back to "optimal" size according to criterion (using CV in a slightly complicated algorithm).
    5. Assign a value to each terminal node that optimizes the criterion (sample mean minimizes RSS!)
- For classification:

1. Steps (2)–(5) are purely mechanical and are carried out exactly the same way
2. Key is (1): RSS is irrelevant for categorical responses.
   (a) What should we be optimizing when we search for splits?
   (b) What CV error should be computed in pruning?
   (c) What values are assigned to terminal nodes?

Terminal nodes

- This is easy: in each terminal node we have some data
- The response values are the classes of the data in that node
- Whichever class occurs most is the chosen class,
- Process of splitting will work on making nodes more "pure" (uniform in class), so ties are *highly* unlikely (impossible?)

Splits

- There are actually several candidates that measure change in "purity" of a node.
  - Different implementations of trees may use different criteria
  - They *can* lead to different answers, perhaps mostly in minor ways
- Simplest idea is to minimize the misclassification rate in each node
  - In terminal node t, among n_t observations observe class counts $n_{t1}, n_{t2}, \ldots, n_{tK}$
  - As per above, class with largest n_{tk} among $k = 1, 2, \ldots, K$ is chosen class, $\hat{y}$
  - Misclassification rate is proportion of node members *not* in class $\hat{y}$
- Turns out to be insensitive to certain important changes
  - Example: Suppose that after exposure to SARS-Cov2, we have 200 asymptomatic people (Class 1) and 50 with symptoms (Class 2)
    - $\hat{y} = 1$
    - Misclassification rate is 50/250=1/5
  - Now suppose we can find some blood measurement that cleanly splits off 100 of Class 1
    - Perhaps this marker suggests built-in immunity
    - Without the marker, your chances of symptoms are higher.
    - This is *very* informative!!!
  - If I make this split, both child nodes remain in class 1: 100-0 and 100-50
    - Misclassification rate is still 50/250

    - So splitting on misclassification will ignore this split and only look for splits that might peel off more Class 2 than Class 1.
    - Maybe no such split exists!!!

- Two other measures are used instead ($\hat{p}_{tk} = n_{tk}/n_t$, the proportion of cases in node t that are in class k)

    1. Gini Index

$$\sum_{k=1}^{K} \hat{p}_{tk}(1 - \hat{p}_{tk}) \tag{1}$$

    measures how variable the class proportions are across the K classes in the node

        (a) Pure node Gini index is 0, maximum value is $(K-1)/K$
        (b) Usually used for splitting (default in `rpart()`)

    2. Cross-Entropy or Deviance

$$-\sum_{k=1}^{K} \hat{p}_{tk} \log(\hat{p}_{tk}) \tag{2}$$

        (a) Related to log-likelihood for a multinomial regression
        (b) Negative sign makes a positive measure with smallest possible value at 0
        (c) An alternative option for splitting (available in `rpart()`)

- Both measures are more sensitive to changes in information from a split even when classes don't change

Pruning

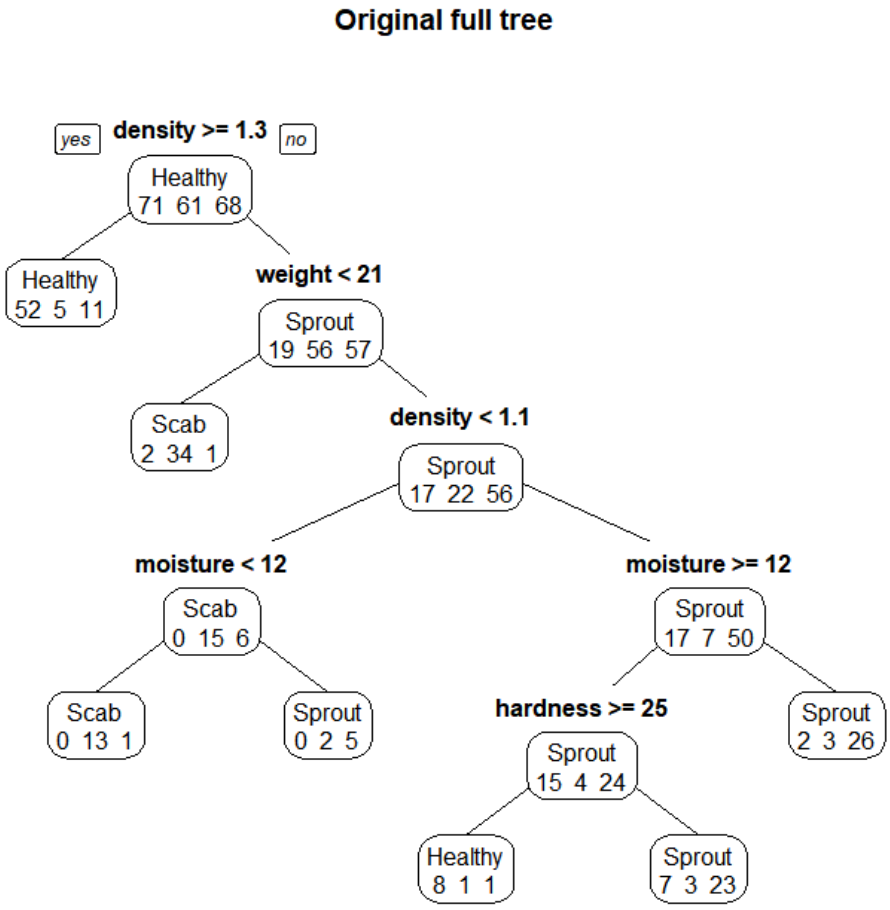
- Any of the measures can be used for computing CV error of trees of different sizes
- `rpart()` uses misclassification rate.

EXAMPLE: Classification Tree on Wheat Data (L20 Classification Tree Wheat.R)

The `rpart()` function can do classification trees by specifying `method="class"`. The response can be a factor or numeric representation of classes. Everything works pretty much the same as before, so I don't repeat the demonstration of most of the code.

The full tree is shown in Figure 1. Notice that there is a split near the bottom, `moisture>=12`, where the Gini index recognizes improvement even though both sides maintain the same best class. One of those nodes is later split into nodes of different classes. On the other hand, the left half of the tree seems to terminate with a large node that could be split further. I forced splits to happen by reducing the minimum size of a node from 7 to 5 (`minbucket=5`) and found that the algorithm terminated because none of the splits below that node caused the class to change when `minbucket=7`. So all would be pruned off by the pruning algorithm using misclassification error. But with the smaller nodes, we do eventually create at least one node with a different class.

Figure 1: Full classification tree from wheat data



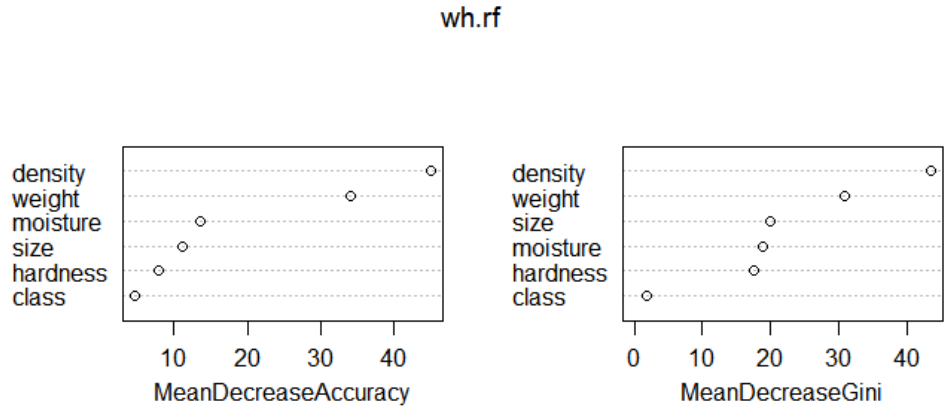
Both min and 1-SE CV error (not shown) remove only the `moisture < 12` split. The training and test error of all 3 trees are added to the table below. The trees are among the worst classifiers we've seen so far, but not by a lot.

Method	Training	Test	Test "SE"
KNN-Opt(12)	0.28	0.49	0.06
Logistic	0.28	0.40	
Logist-LASSO-min	0.28	0.43	
LDA	0.28	0.39	
QDA	0.26	0.43	
GAM	0.24	0.37	
NB-normal	0.32	0.39	
NB-kernel	0.23	0.43	
Tree-Full	0.20	0.47	
Tree-min	0.21	0.44	
Tree-1SE	0.21	0.44	

3 Classification with Random Forests

- Recall that BAGGING means "bootstrap aggregation"
  - Refit the same prediction machine many times to bootstrap resamples of the data
  - Average the prediction results
  - Especially useful when applied to regression trees because they are automatically nearly unbiased but very variable.
- RANDOM FOREST Improved bagging by adding a small "tweak":
  - In addition to using bootstrap resampling, *also randomly select a subsample of explanatory variables!*
  - **For each parent node (i.e. potential split):**
    - Randomly choose $m \leq p$ variables.
      - Common recommendation: $m = \sqrt{p}$ for classification, $= p/3$ for regression
      - m is a tuning parameter; can play with it.
      - Note that $m = p$ uses all of the variables, and so is exactly bagging.
    - Pick best split only from among candidate variables for that parent node.
  - Resampled data is chosen for whole tree; subsets of explanatories are chosen for each split.
- We can make a forest of classification trees to do classification.
- We can do the same thing with classification trees

Figure 2: Random Forest Variable Importance for Wheat Data



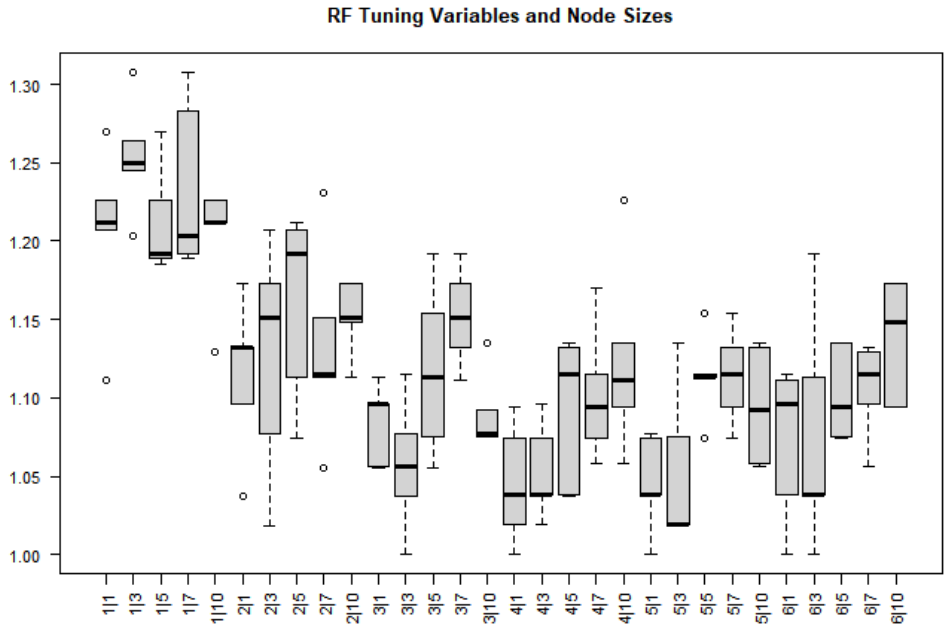
1. Trees are built as described above
2. For a given x the observation will fall into different terminal nodes in different trees
3. Hence, it is likely that different trees may classify x differently
   (a) How do you take a mean as we did in regression?
4. Use a "voting" system, where each tree in the ensemble votes for its predicted class
   (a) The class with the most votes wins
   (b) For example, suppose $K = 3$ and $B = 1000$ trees are used. If x falls into a terminal node that predicts Class 1 in 600 trees, Class 2 in 300 trees, and Class 3 in 100 trees, then Class 1 has the most votes, so $\hat{f}(x) = 1$
5. Variable Importance is measured as before with needed adjustments
   (a) % reduction in Gini Index (1) rather than RSS
   (b) Reduction in accuracy (=misclassification rate, now) measured in comparison to prediction of Out-of-bag samples using random permutation.

Example: Vehicle Image Recognition (Random Forest Classification.R) The `randomForest` package automatically detects whether the response is numerical or a factor and reacts with the appropriate loss functions and accessories. Note that it builds very large trees for classification, essentially isolating each observation into its own node. This may be too large in some cases, so it can be worthwhile to tune number of variables, `mtry`, and node sizes, `nodesize`.

The program first fits the default, using `mtry` as $\sqrt{6} = 2$ (rounded). The variable importance plot is shown in Figure 2. It supports what we have seen elsewhere, that `density` is most important and `weight` is also very important. The test error was just 0.47, though, so not very good.

Next I tuned `mtry` (1,2,...,6) and `nodesize` (1,3,5,7,10) using RF's built-in OOB error. I reran the forests 5 times to get a less variable estimate of error and to give us some measure

Figure 3: Random Forest Variable Importance for Wheat Data



of the variability. The relative misclassification rates (computed similarly to relative MSPE) are shown in Figure 3. We see that 1 is a poor choice for `mtry`, and that for good choices of `mtry` we generally want smaller nodesizes (larger trees). The two combinations with smallest mean misclassification rate were 4,1 and 5,1, so I used 4,1. The test error from this run is better than before, 0.41, but not better than other methods.

Method	Training	Test	Test "SE"
KNN-Opt(12)	0.28	0.49	0.06
Logistic	0.28	0.40	
Logist-LASSO-min	0.28	0.43	
LDA	0.28	0.39	
QDA	0.26	0.43	
GAM	0.24	0.37	
NB-normal	0.32	0.39	
NB-kernel	0.23	0.43	
Tree-min	0.21	0.44	
RF Default	0	0.47	
RF OOB-Tuned	0	0.41	

4 Boosting for Classification

(ISLR just says, "Boosting classification trees proceeds in a similar but slightly more complex way, and the details are omitted here.")

- Boosting was invented originally for classification
  - It was later extended to regression
- There are *lots* of different algorithms for both classification and regression
  - The one we learned for regression is called GRADIENT BOOSTING
- There is a version of gradient boosting that does classification
  - ...which seems weird, because the whole thing was about building trees that explain *residuals*
  - You can't make "residuals" with categorical response variables
  - Instead, build trees that explain the the optimization criterion (usually the "deviance" criterion from equation (2))
  - Trees in sequence try to relate how explanatory variables make this smaller instead of making RSS smaller
  - Seems weird, but it actually works!
- Tuning parameters are the same as before.

EXAMPLE: Gradient Boosting Classification on Wheat Data (L20 Boosting Wheat.R)

Gradient boosting through `gbm()` USED TO HAVE a `distribution="multinomial"` argument value that uses the deviance criterion (2). But a warning says it is now broken. Will it ever get repaired? I don't know.

So for the moment, we can only do 2-class classification problems. Use `distribution="bernoulli"`, which expects inputs to be either 0-1 or TRUE-FALSE. *If you use factors, `gbm()` will assume `multiniomial` !!* Lots of useful options for optimizing the number of trees. Measure of variable importance available in `summary()`. Tuning parameters (described in program) are the same as before, and three of them—`n.trees`, `shrinkage`, `interaction.depth`—must be tuned before you can have any confidence that the machine will work right.

The program does gradient boosting for the binary I(`type=Healthy`), called `Hbin` in the program. The mechanics are the same as before, except that misclassification error needs a little bit of programming. The predicted values are $P(Y = 1)$, so you need to classify according to whether this probability is >0.5.

Details are in the program. I won't cover the output here, as is it the same as for regression. Since the predictions are for a different response variable, we can't really compare the error rate them to other methods' error rates on the 3-class problem.

---

5 What to take away from this

1. Trees-based ensembles are useful classifiers
2. Random forests are relatively easy to apply.
3. The `gbm()` function for boosting has some issues at the moment for problems with $K > 2$.

6 Exercises

Application

Return to the Vehicle data used in the previous lecture. Use the same split as before.

```
set.seed(46685326, kind="Mersenne-Twister")
perm <- sample(x=nrow(vehdata))
set1 <- vehdata[which(perm <= 3*nrow(vehdata)/4), ]
set2 <- vehdata[which(perm > 3*nrow(vehdata)/4), ]
```

We will fit single trees and random forests to the training data and test them at the end using the test set.

1. Reset the seed for the Mersenne Twister to 8646824. Code two models: a classification tree using all defaults, followed by a second tree setting `cp=0`. This way, every student should have the same CV results within `rpart`.

   (a) **Print out the CP table for the default tree.** Is there any evidence that larger trees are needed than what the defaults allow? **Explain.**

   We will no longer use this run. All remaining questions refer to the `cp=0` object.

   (b) **Print out the CP table for the tree with `cp=0`. How many splits are suggested as best?**

   (c) Use Tom's code to find the optimal CP value for pruning using minimum CV and using the 1-SE rule. **Report these two numbers**

   (d) **Report training and test error of all three of these trees, full and two pruned.**

   (e) **How does test error compare to other models?**

2. Fit a default random forest

   (a) **Report variable importance and comment: which variables seem most important and which seem least important?**

   (b) **Report test error and compare it to other methods**

3. Tune the number of variables and the node sizes for the random forest using its internal out-of-bag error. Since $p = 18$ and the default is `mtry`$= \sqrt{18} \approx 4$, use a grid of 2,4,6,10,18. For node size, default is 1, so try 1,3,5,7,10. Rerun 5 random forests of 500 trees each.

(a) **Report the average misclassification rate for each combination**

(b) **Show relative error boxplots**

(c) **Which combination do you think is best?**

(d) **Report test error for that combination. Compare it to the default RF: did it help? If so, how does it compare to other methods?**