STATISTICS 452/652: Statistical Learning and Prediction

October 18, 2020

Lecture 14: Ensembles - Bootstrap Aggregation

(Reading: Section 8.2.1)

1 Goals of lecture

- Regression trees are automatic machines with the capacity to produce nearly unbiased predictions
  - Especially true when n/p ratio is moderately large, so that enough steps can be formed in each variable as needed
- However, trees are very variable and sensitive to errors in data
  - Step function tends to make bigger errors at ends of steps than in middle
- We will see how we can exploit the automatic near-unbiasedness of trees while substantially reducing their variability

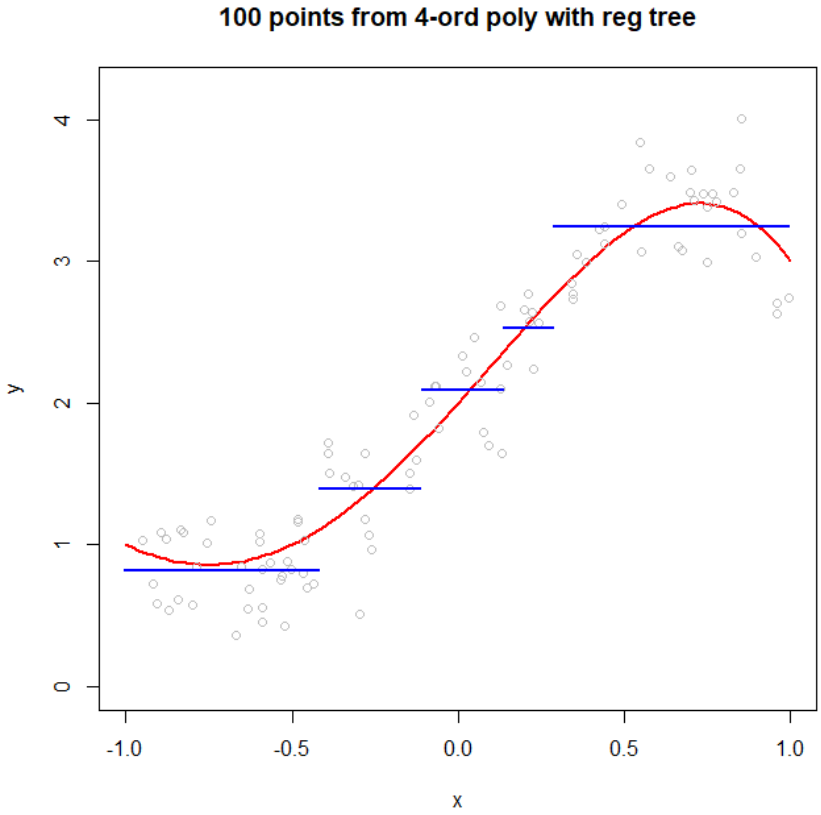
2 The Problem with Trees

I have said that regression trees are notoriously unstable. What does that mean?

Example: Regression Trees on Polynomial Data (L14 - Regression Tree Instability.R)

We return to our old 4-th order polynomial model. Let's see what a regression tree does on those data. Figure 1 shows a sample of $n = 100$ from the same polynomial model we have studied before. A fitted, min-CV-pruned regression tree is overlaid in the figure.

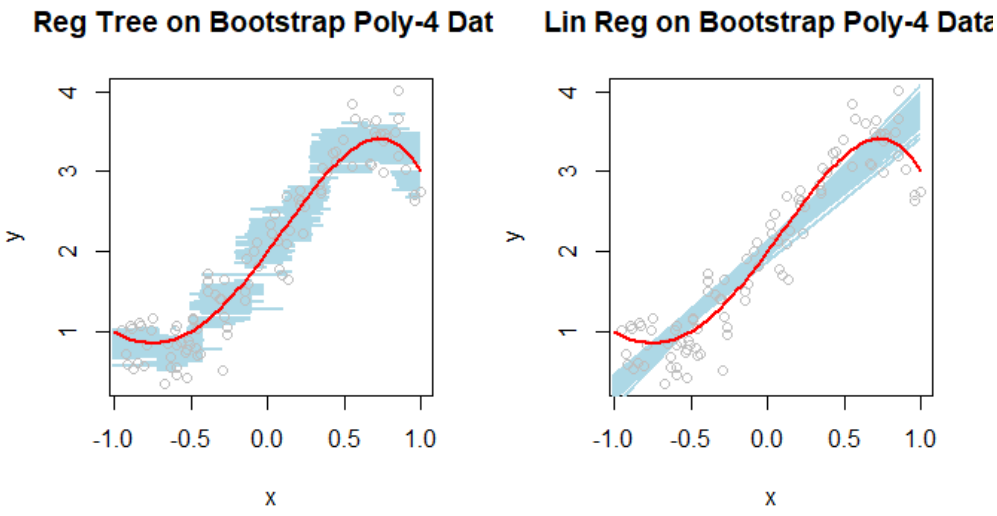
Figure 1: Fitted regression tree with minimum-CV pruning for sample of 100 from 4th-order polynomial studies previously



We see that the tree definitely follows the general pattern of the data automatically, but is quite far from the curve in places, particularly where the jumps take place. It is easy to imagine that different samples would similarly follow the same pattern, but with jumps in randomly different places.

We can see this phenomenon clearly if we rerun the tree on bootstrap resamples from the original data. Recall that we use bootstrap as a way to measure test error of a machine by refitting the machine on resamples drawn with replacement from the original data, and predicting the data left out. Here, I draw 100 resamples and refit the tree, showing these 100 fits in Figure 2. I have added linear regression fits to these resamples for comparison.

Figure 2: Results of regression tree and linear model fits to 100 resamples from the original sample.



The results show two things:

1. Compared to linear regression, there is much more variability across the resamples in the predicted values from the regression tree.
2. The tree nonetheless follows the general shape of the polynomial surface pretty well, something that the linear regression can't do very well.

- This phenomenon generalizes to arbitrary shapes in arbitrary numbers of dimensions.
  - *If there is enough data relative to the number of dimensions, regression trees are automatically nearly unbiased, but very variable.*
  - Most of their error comes from variability rather than bias.
    - Quite the opposite of rigid models, as we see above.
- This begs the question: **Is there some way to make trees less variable?**
  - YES!!!

3 Bootstrap Aggregating (Bagging) a Machine

Given what we have just seen in the previous example, an obvious thing to do might be to take the average of all of the bootstrap-tree predictions at each x. Let's formalize this idea:

- Let (X, Y) represent a random sample of size n from some population

- Formally, suppose we have *any* prediction machine operating in the p-dimensional space of X , $f(X)$
  - From a set of data, we get $\hat{f}(X)$

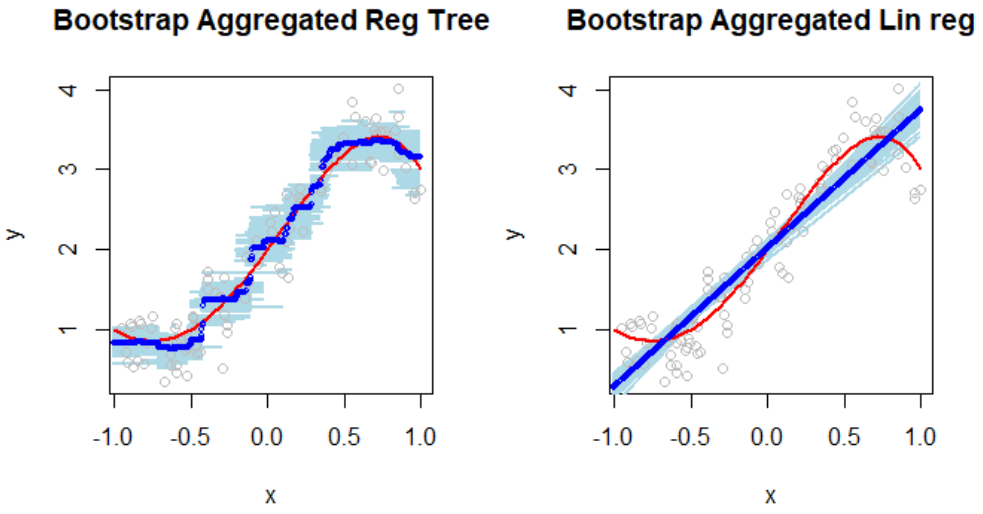
1. Now let's take a resample (with replacement) from (X, Y). Call this (X_1^*, Y_1^*)
   (a) Fit the machine to (X_1^*, Y_1^*). Call this $f^{*1}(X)$
2. Do it again and again and again, say B times.
   (a) Draw resamples $(X_2^*, Y_2^*), (X_3^*, Y_3^*), \ldots, (X_B^*, Y_B^*)$.
   (b) Refit machines $\hat{f}^{*2}(X), \hat{f}^{*3}(X), \ldots, \hat{f}^{*B}(X)$
3. For any value x, compute predicted values from each of the bootstrap fits, $\hat{y}^{*b} = \hat{f}^{*b}(x)$
4. Aggregate the predicted values together into an average, $\hat{y}_{agg} = \frac{1}{B}\sum_{b=1}^{B} \hat{y}^{*b}$
5. More generally, we can create a new machine, $\hat{f}_{agg}(X) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(X)$

This process is called BOOTSTRAP AGGREGATING a prediction machine, known less formally as BAGGING. The function f used as the machine for aggregation is called the BASE LEARNER.

Example: Bagging regression tree and linear regression (L14 - Regression Tree Instability.R)

Returning to the bootstrap-replicated predictors that we created in Figure 2, we now add the averages across the 100 resamples at each x. See Figure 3

Figure 3: Fitted tree and surface for Prostate data using only `lcavol` and `pgg45`.



The bagged regression tree is remarkable. Without knowing anything about the true structure, and using a clunky step function as the base learner, the average tracks the true structure remarkably well. It is still a step function, but the jumps are much smaller for the most part, and flatness is somewhat smoothed out by the averaging.

The linear regression is *not* improved by bagging. Since ALL of the lines over- or under-estimate the true structure in certain regions, the average is no better.

We can see this in the root-MSPEs for each method, using a 1000-observation test set of new data generated from the same model:

Model	root-MSPE	Relative rMSPE
Original Tree	0.339	1.11
Bagged Tree	0.315	1.03
Original LM	0.395	1.29
Bagged LM	0.395	1.29
Fitted Polynomial	0.306	1

The bagged tree comes very close the the predictive performance of the correct model fitted to the data—within 3% in root MSPE—but without having to know the actual shape. The original tree is not too bad, at +10% over the best model, but we clearly can do better by aggregating. Both the original and bagged linear models bad, with nearly 30% excess root-MSPE.

3.1 Properties of Bagging

- You can bag *any* machine: linear regression, polynomial, GAM, PPR, NN, or whatever.
- Bagging is a way to reduce the *variance* of a prediction machine. It cannot fix bias.
  - Averaging is a way to reduce variance in general
  - If the shape of a machine prevents it from ever matching the true structure $g(\mathbb{X})$, no amount of additional sampling or resampling can fix that
    - See Figure 2 to see this
- Bagging is most effective when
  - It is applied to machines that have low bias.
  - It is applied to machines that are easy to fit—fast and with minimal tuning—since the process needs to be automated and replicated many times.
- This sounds like a regression tree!
  - In order to be unbiased, need enough data to make enough splits in each dimension where the mean changes appreciably
  - Not especially good when n/p is not very large
  - Pruning is usually NOT done

    - Pruning is a way to balance bias and variance in a single predictor
    - We would rather have lower bias and are willing to accept higher variance, which bagging can reduce
    - Larger tree is less biased; hence, no pruning.

- B needs to be large enough to bring the variance down, but small enough to make computing feasible.
  - Usually not much benefit to taking $B > 50$ or so.
- A cool thing about bagging is that it does come with a built-in method for estimating test error
  - Recall that each bootstrap replicate leaves some data out of the resample
    - These were called "out-of-bag" (OOB) data
    - Now you know why!
  - Consider all the resamples where observation i is OOB (about $B/3$ of them)
    - Create a version of the bagged function using *only those resamples,*
    - Evaluate that special version at x_i to predict y_i
    - Compute prediction error between observed y_i and that prediction
    - Square these and sum them
  - This is called the OOB error

4 What to take away from this

1. Bagging is a way to automatically reduce prediction error of a machine
   (a) Nearly always works, as long as
      i. there is enough data
      ii. the base learner is relatively unbiased
   (b) Only costs computing time
2. Bagging is *one way* to combine multiple fitted machines into one improved machine.
   (a) This is a general strategy, and there are numerous other ways to do this
   (b) The general concept is called ENSEMBLE LEARNING
      i. A musical ensemble like an orchestra combines many different instruments into one "sound-producing machine".
      ii. By this analogy, bagging is more like a choir, in that the base learners (instruments) are all the same type (voices)
      iii. There do exist methods that operate on base learners of different types
         A. STACKING
3. Some other methods are actually a bit better than bagging, so it is not often used exactly as described here.

5 Exercises

Concepts

Refer to the program **L14 - Regression Tree Instability.R**.

1. Bagging the 4th-order polynomial model

   (a) **When we know that the true structure is also a 4th-order polynomial, might it be beneficial to apply bagging to the 4th-order polynomial model? Explain.**

   (b) **If we had no idea what the true structure was, would this still be a good idea? Explain.**

   (c) Add code into the bootstrap loop to bag the 4th-order polynomial as the base learner.

      i. **What is the root MSPE for the model? Is this much of an improvement over the single model?**

      ii. **Create a plot that shows all of the bootstrap fits along with the average. Comment on apparent bias and variance.**

   (d) Rerun the code with the added code for bagging the 4th-order polynomial, but change the sample size to 20. Also, in the regression tree, add the option, `minsplit=9`, which allows the tree to split nodes with as few as 10 observations in them. Otherwise, with $n = 20$ the default `minsplit=20` will prevent ANY splits form happening!

      i. **Make a table comparing root MSPE for all six models—bagged and original data versions of linear model, tree, and poly-4. Add a column for all rMSPEs relative to the 4th-order polynomial on the original data (even if this means that some results may be less than 1, indicating relative improvement).**

      ii. **Create plots that show all of the bootstrap fits along with the averages for each model. Comment on apparent bias and variance.**

      iii. **Comment on the results: with a much smaller sample size, what has changed? Why?** (2-3 sentences might be needed here)

2. BONUS: Bagging a NN. Consider two different combinations of size and decay: (2,1) and (6,0.001)

   (a) **Which combination do you think might make a better base learner for a bagging ensemble? Why?**

   (b) Add code for each into the bagging code with $n = 100$ again (running multiple restarts within each resample). Also fit each one to the original data, **Get test errors root MSPEs for these four models and add them to the table from the previous exercise, along with error relative to the original-data 4th-order polynomial.**

(c) **Create plots that show all of the bootstrap fits along with the averages for each model. Comment on apparent bias and variance.**