Boosting

36-290 – Statistical Research Methodology

Week 8 Thursday – Fall 2021

Context

"Can a set of weak learners create a single strong learner?"

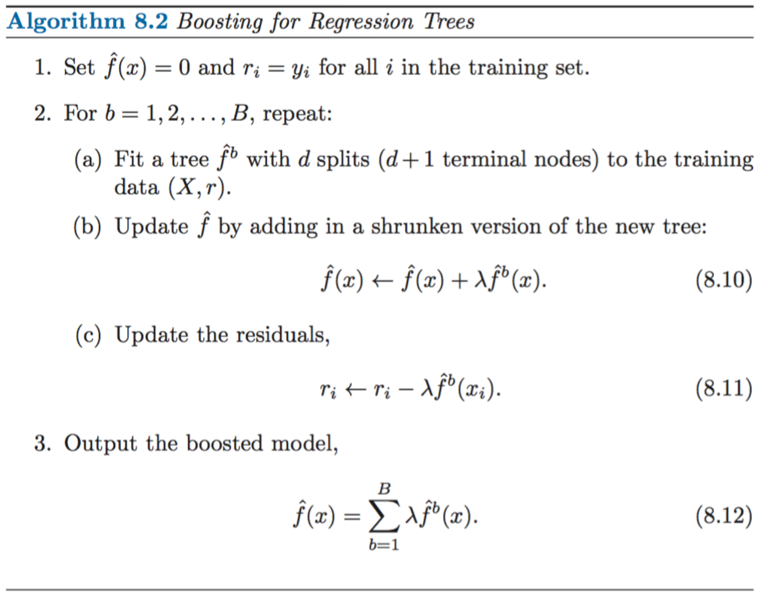
— Kearns & Valiant

An example of a "weak learner" is, e.g., a decision tree with a single split (i.e., a "decision stump"). The "set of weak learners" is, e.g., the repeated generation of stumps given some iterative rule, such as "let's upweight the currently misclassified observations next time around." (This is the core of the AdaBoost algorithm.) Through iteration, a strong learner is created.

Boosting is a so-called "meta-algorithm": it is an algorithm that dictates how to repeatedly apply another algorithm. As such, boosting can be applied with many models, like linear regression. However, boosting is most associated with trees. In this context, the contrast of boosting with bagging is that bagging involves growing many separate deep trees that are aggregated, while boosting grows one tree sequentially by adding a weighted series of stumps.

There are also many different kinds of boosting (i.e., many different ways to define the meta-algorithm: upweight observations the next time, etc.). The most oft-used boosting algorithm currently is *gradient boosting*, which we describe on the next slide.

Gradient Boosting



Algorithm 8.2 *Boosting for Regression Trees*

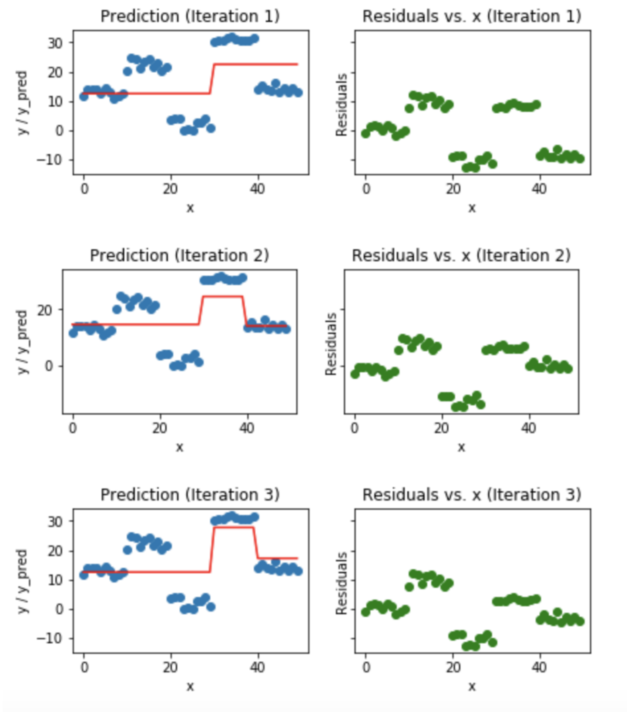
```
1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all $i$ in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
    (a) Fit a tree $\hat{f}^b$ with $d$ splits ($d+1$ terminal nodes) to the training data $(X, r)$.
    (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
            $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$    (8.10)
    (c) Update the residuals,
            $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$    (8.11)
3. Output the boosted model,
            $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$    (8.12)
```

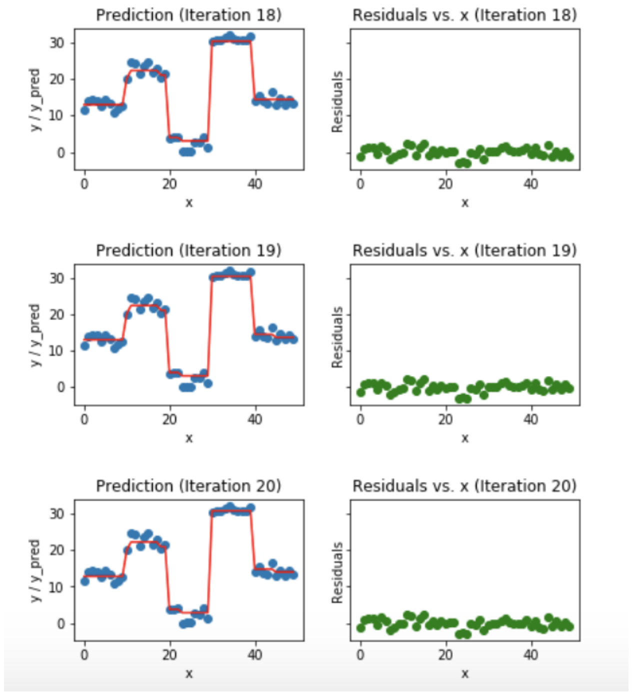
(Algorithm 8.2, *Introduction to Statistical Learning* by James et al.)

The core idea of regression tree boosting: it *slowly* learns a model by fitting the *residuals* of the previous fit. In other words, it fits a stumpy tree to the original data, shrinks that tree (note the λ parameter), updates the residuals, sets the data to be those residuals, and repeats. Hence each iteration of boosting attempts *to hone in on those data that were not well fit previously*, i.e., those data for which the residual values r_i continue to be large.

The smaller the value of λ, the more slowly and conservatively the final tree is grown.

Gradient Boosting





(These pictures were taken from this web page.)

Gradient Boosting: Regression Example

We reuse the example from the random forest notes. The dataset contains 10000 galaxies, of which 7000 will be used for training the random forest model and 3000 will be used to assess it. The predictors are u-, g-, r-, i-, z-, and y-band magnitudes (brightnesses from the near-ultraviolet through the optical regime to the near-infrared) and the response is redshift, a measure of the distance of the galaxies from Earth. Note that classification uses similar code, with some argument alterations that are detailed in today's lab.

Note: the function calls given below "look weird." That's because the `xgboost` developers didn't bother to try to at least approximately match `R` modeling function syntax. If you are in a position to contribute to `R`: be better!

```
library(xgboost)
train = xgb.DMatrix(data=as.matrix(pred.train),label=resp.train)
test  = xgb.DMatrix(data=as.matrix(pred.test),label=resp.test)
set.seed(101)
xgb.cv.out = xgb.cv(params=list(objective="reg:squarederror"),train,nrounds=30,nfold=5,verbose=0)
cat("The optimal number of trees is ",which.min(xgb.cv.out$evaluation_log$test_rmse_mean))
```

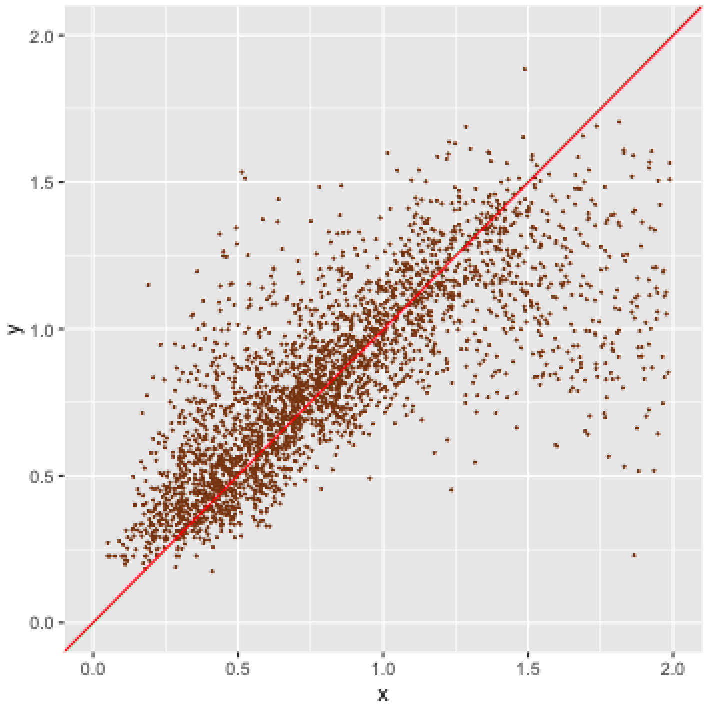
```
## The optimal number of trees is  26
```

```
xgb.out = xgboost(train,nrounds=which.min(xgb.cv.out$evaluation_log$test_rmse_mean),params=list(objective="r
resp.pred     = predict(xgb.out,newdata=test)
round(mean((resp.pred-resp.test)^2),3)
```

```
## [1] 0.075
```

Gradient Boosting: Regression Example

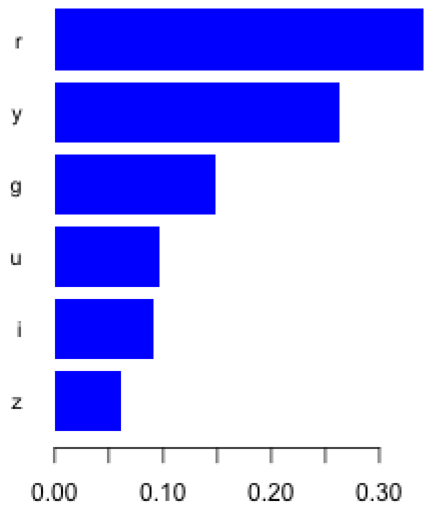
```r
suppressMessages(library(tidyverse))
ggplot(data=data.frame("x"=resp.test,"y"=resp.pred),mapping=aes(x=x,y=y)) +
  geom_point(size=0.1,color="saddlebrown") + xlim(0,2) + ylim(0,2) +
  geom_abline(intercept=0,slope=1,color="red")
```



Gradient Boosting: Regression Example

Like random forest, boosting allows one to measure variable importance. This importance measure is called the "gain," defined as the "fractional contribution of each feature to the model based on the total gain of this feature’s splits. Higher percentage means a more important predictive feature." (Definition from `xgboost` documentation.)

```
imp.out = xgb.importance(model=xgb.out)
xgb.plot.importance(importance_matrix=imp.out,col="blue")
```


r
y
g
u
i
z
0.00
0.10
0.20
0.30