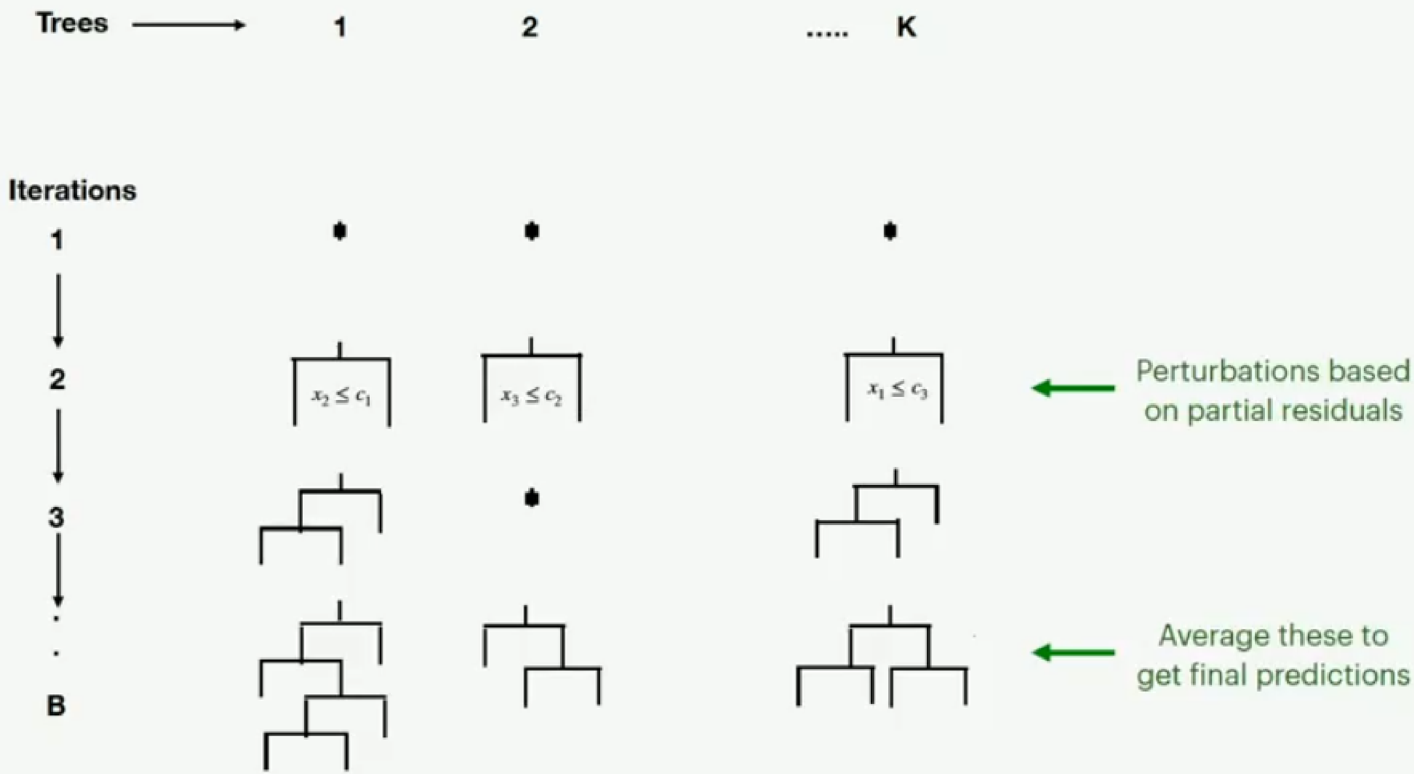
Bayesian Additive Regression Trees

- We discuss *Bayesian additive regression trees* (BART), an ensemble method that uses decision trees as its building blocks.
- Recall that *bagging* and *random forests* make predictions from an average of regression trees, each of which is built using a random sample of data and/or predictors. Each tree is built separately from the others.
- By contrast, *boosting* uses a weighted sum of trees, each of which is constructed by fitting a tree to the residual of the current fit. Thus, each new tree attempts to capture signal that is not yet accounted for by the current set of trees.

Bayesian Additive Regression Trees — Details

- BART is related to both random forests and boosting: each tree is constructed in a random manner as in bagging and random forests, and each tree tries to capture signal not yet accounted for by the current model, as in boosting.
- The main novelty in BART is the way in which new trees are generated.
- BART can be applied to *regression*, *classification* and other problems; we will focus here just on regression.

BART algorithm — the idea



Bayesian Additive Regression Trees — Some Notation

- We let K denote the number of regression trees, and B the number of iterations for which the BART algorithm will be run.
- The notation $\hat{f}_k^b(x)$ represents the prediction at x for the kth regression tree used in the bth iteration. At the end of each iteration, the K trees from that iteration will be summed, i.e. $\hat{f}^b(x) = \sum_{k=1}^{K} \hat{f}_k^b(x)$ for $b = 1, \ldots, B$.

BART iterations

- In the *first iteration* of the BART algorithm, all trees are initialized to have a single root node, with $\hat{f}_k^1(x) = \frac{1}{nK}\sum_{i=1}^n y_i$, the mean of the response values divided by the total number of trees. Thus,

$$\hat{f}^1(x) = \sum_{k=1}^{K} \hat{f}_k^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i$$

- *In subsequent iterations*, BART updates each of the K trees, one at a time. In the bth iteration, to update the kth tree, we subtract from each response value the predictions from all but the kth tree, in order to obtain a *partial residual*

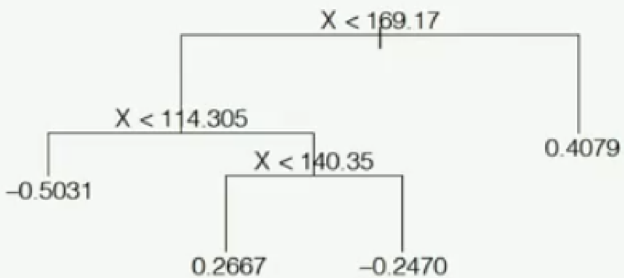
$$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^b(x_i) - \sum_{k'>k} \hat{f}_{k'}^{b-1}(x_i),\ i = 1, \ldots, n$$

New trees are chosen by perturbations

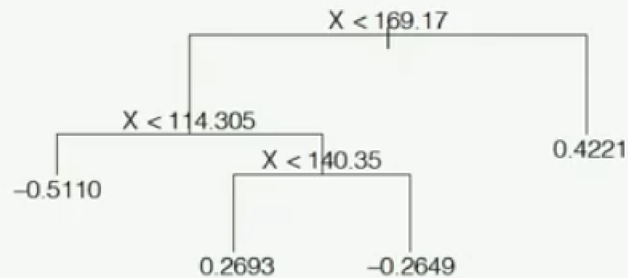
- Rather than fitting a fresh tree to this partial residual, BART randomly chooses a perturbation to the tree from the previous iteration ($\hat{f}_k^{b-1}$) from a set of possible perturbations, favoring ones that improve the fit to the partial residual.
- There are two components to this perturbation:
    1. We may change the structure of the tree by adding or pruning branches.
    2. We may change the prediction in each terminal node of the tree.

Examples of possible perturbations to a tree

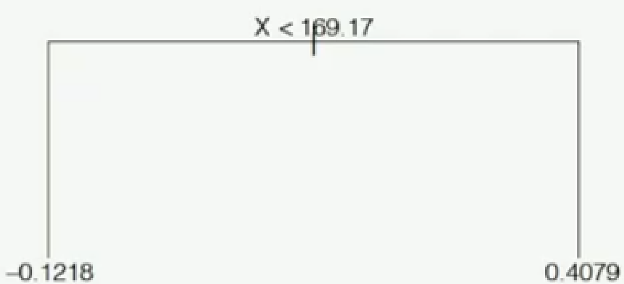
(a): $\hat{f}_k^{b-1}(X)$



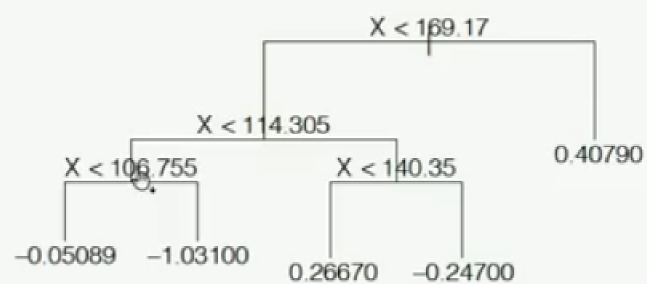
(b): Possibility #1 for $\hat{f}_k^{b}(X)$



(c): Possibility #2 for $\hat{f}_k^{b}(X)$



(d): Possibility #3 for $\hat{f}_k^{b}(X)$



What does BART Deliver?

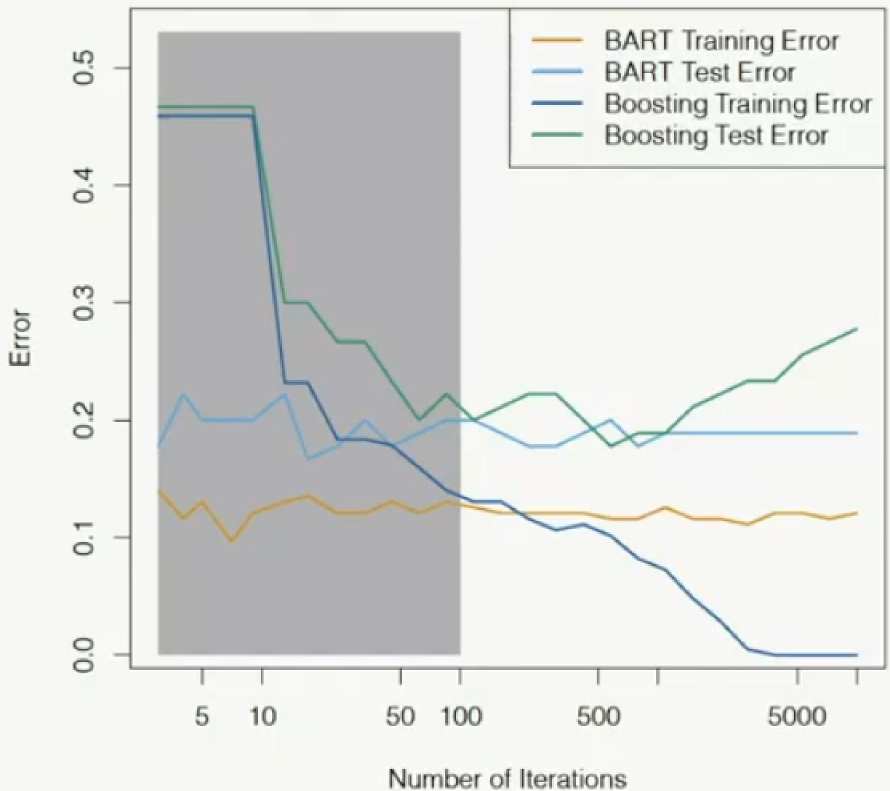
- The output of BART is a collection of prediction models,

$$\hat{f}^b(x) = \sum_{k=1}^{K} \hat{f}_k^b(x), \text{ for } b = 1, 2, \ldots, B.$$

- To obtain a single prediction, we simply take the average after some L *burn-in iterations*, $\hat{f}(x) = \frac{1}{B-L}\sum_{b=L+1}^{B} \hat{f}^b(x)$.
- The perturbation-style moves guard against overfitting since they limit how *hard* we fit the data in each iteration.
- We can also compute quantities other than the average: for instance, the *percentiles* of $f^{L+1}(x), \cdots f^B(x)$ provide a measure of uncertainty of the final prediction.

BART applied to the Heart data

$K = 200$ trees; the number of iterations is increased to $10,000$.
During the initial iterations (in gray), the test and training errors jump around a bit. After this initial burn-in period, the error rates settle down.
The tree perturbation process largely avoids overfitting.



BART is a Bayesian Method

- It turns out that the BART method can be viewed as a *Bayesian* approach to fitting an ensemble of trees: each time we randomly perturb a tree in order to fit the residuals, we are in fact drawing a new tree from a *posterior* distribution.
- Furthermore, the BART algorithm can be viewed as a *Markov chain Monte Carlo* procedure for fitting the BART model.
- We typically choose large values for B and K, and a moderate value for L: for instance, $K = 200$, $B = 1{,}000$, and $L = 100$ are reasonable choices. BART has been shown to have impressive out-of-box performance — that is, it performs well with minimal tuning.