Bagging, Random Forests, and Boosting

(ISLR 8.2)

Yingbo Li

Southern Methodist University

STAT 4399

Outline

Decision Trees Have High Variance

- If we randomly split the training data into 2 parts, and fit decision trees on both parts, the results could be quite different.
- How to lower variance? Taking average!
- Recall that given a set of n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2, the variance of the mean $\bar{Z} = (Z_1 + \cdots + Z_n)/n$ of the observations is given by σ^2/n.
- How to average a set of decision trees based on just one dataset?

Bootstrap Aggregation (Bagging)

- *Bagging* is a general-purpose procedure for *reducing the variance* of a statistical learning method (not necessarily a tree model).

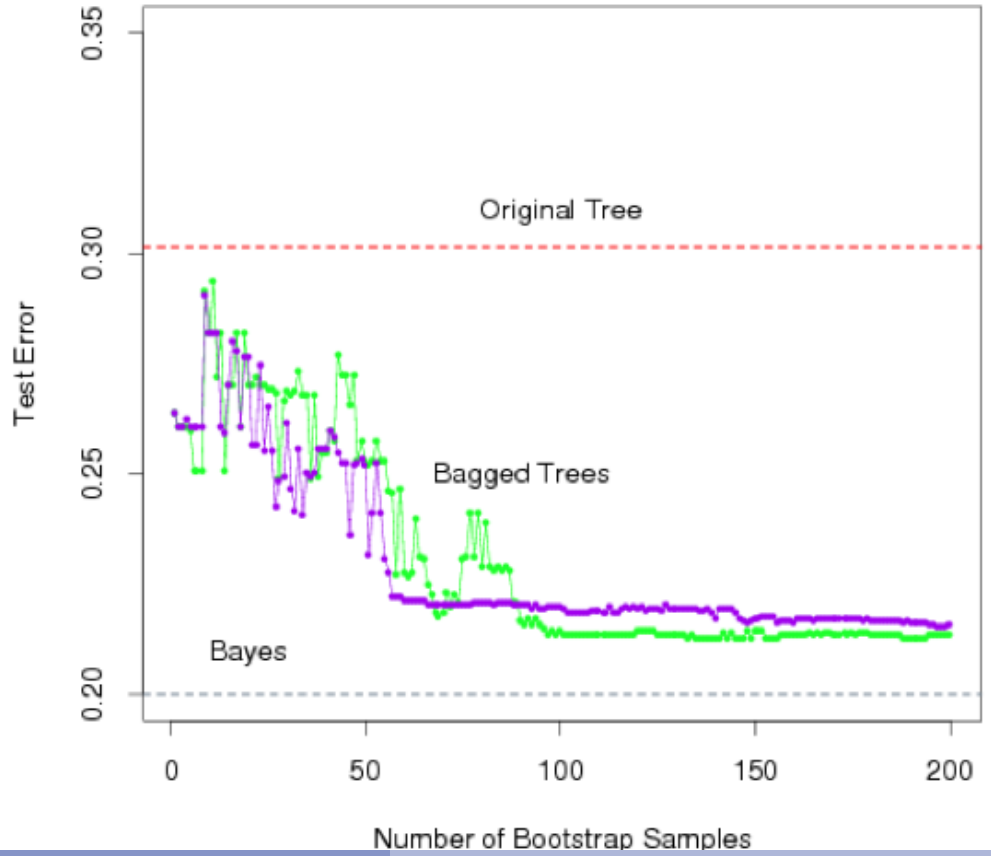
- Generate B different bootstrapped training datasets
- For each $b = 1, \ldots, B$, train the statistical learning method on the bth bootstrapped set, and obtain the prediction $\hat{f}^b(x)$ for a point at x.
- Average all B predictions (for regression):

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^b(x)$$

Bagging for Tree Models

- When constructing B trees from the bootstrapped training datasets, we do not need to prune the trees.
  - So each individual tree has high variance but low bias.
  - Averaging these trees yields both low variance and bias.

- For classification, there are two approaches:
  - Majority vote: report the most commonly class among B predictions (green).
  - Average probabilities, predict to the class with the highest probability (purple).
- Both work well.

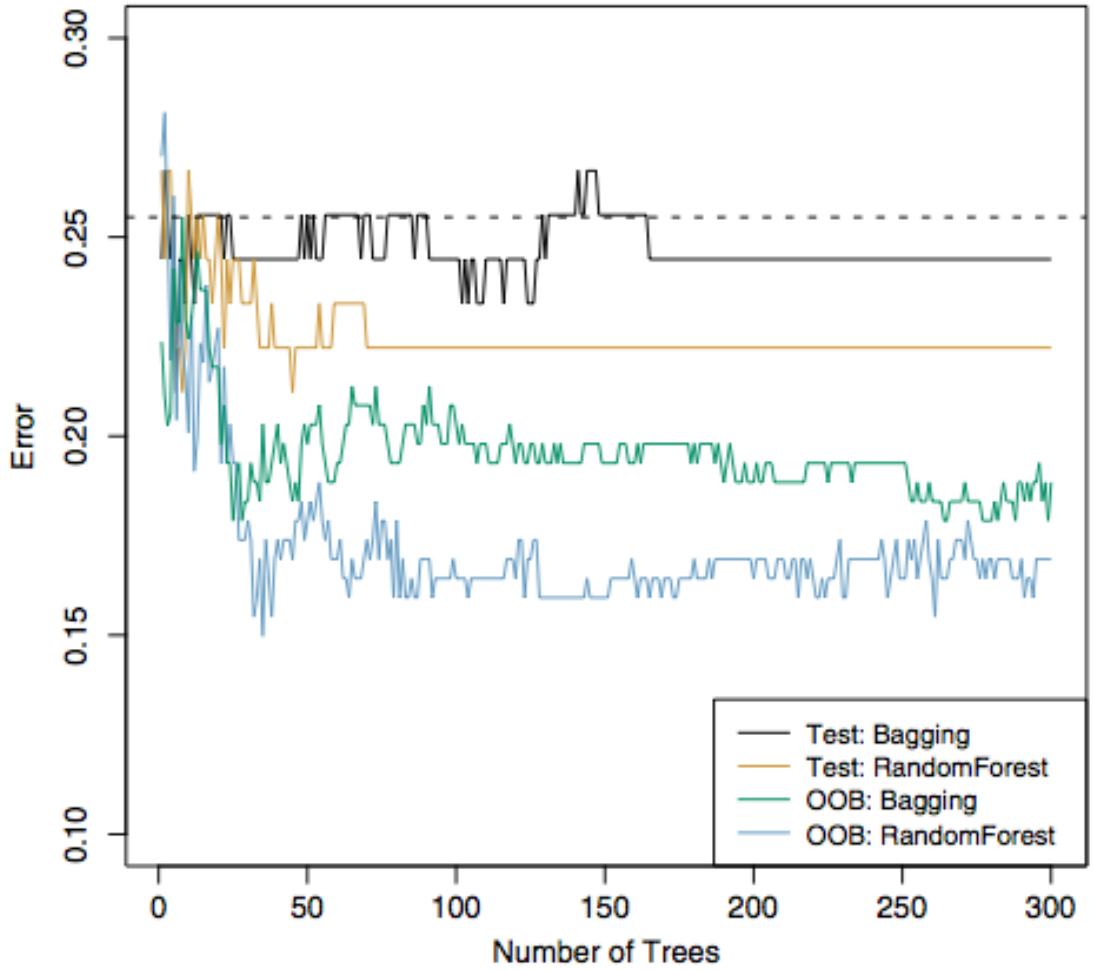


Out-of-bag (OOB) Error

- There is a very easy way to estimate the test error of a bagged model.
- On average, each bootstrap data have about $2/3$ of the observations.
- The remaining observations not used to fit a given bagged tree are referred to as the *out-of-bag (OOB) observations*.
- We can predict the response for the ith observation using each of the trees in which that observation was OOB. This will yield around $B/3$ predictions for the ith observation, which we average.
- When B is large, OOB error is essentially the LOOCV error.

The Heart Data Example

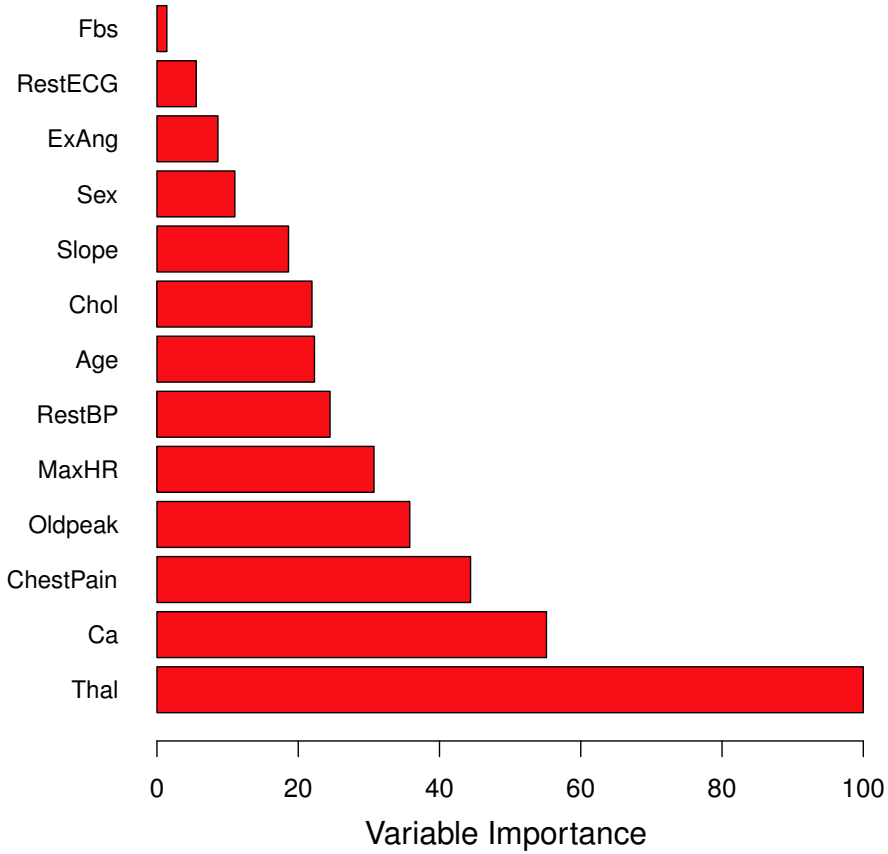
- Dashed: test error from a single tree.
- Black: test error, bagging.
- Green: OOB error, bagging.



- The value of B is not critical for bagging.
- For this dataset, $B = 200$ is sufficient.

Variable Importance Measure

- Bagging improves prediction accuracy at the cost of interpretability.
- *Variable importance*: the total amount that the RSS (or Gini index) is decreased due to splits over a given predictor, averaged over all B trees.

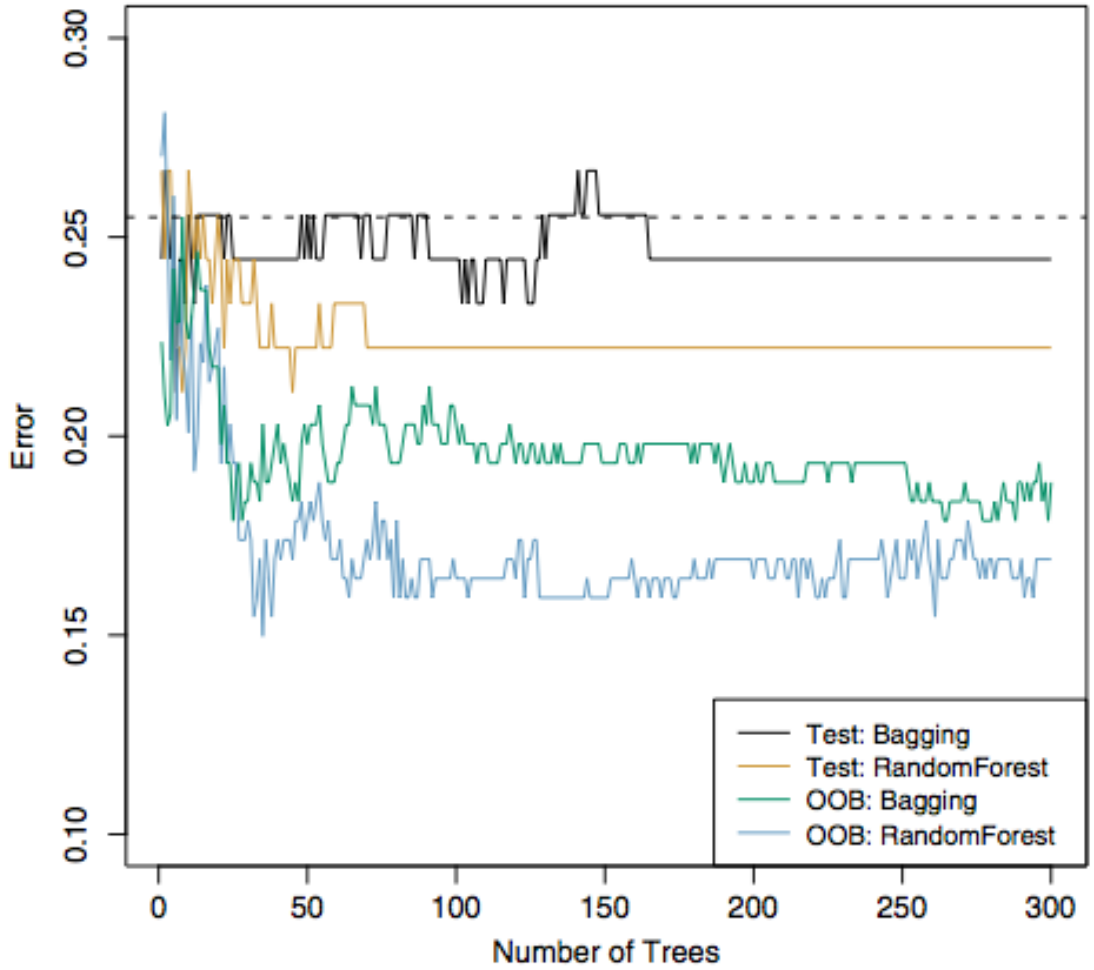


Random Forests

- It is a very efficient statistical learning method.
- It builds on the idea of bagging, but it provides an improvement because it *de-correlates* the trees.
- How does it work?
  - We build a number of decision trees on bootstrapped training samples.
  - But when building these trees, each time a split in a tree is considered, *a random sample of m predictors* is chosen as split candidates from the full set of p predictors.
  - A fresh selection of m predictors is taken at each split, and usually $m \approx \sqrt{p}$. For the `Heart` data, $4 \approx \sqrt{13}$.

The Heart Data Example

- Dashed: test error from a single tree.
- Black: test error, bagging.
- Green: OOB error, bagging.
- Yellow: test error, random forest.
- Blue: OOB error, random forest.



- The value of B is not critical for random forest, either.
- For this dataset, $B = 200$ is sufficient.
- Random forest outperforms bagging in both test error and OOB error.

Why Not Using All p Predictors?

- All bagged trees will look similar. Hence all the predictions from the bagged trees will be highly correlated.
- Averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities.

- In building a random forest, at each split in the tree, the algorithm is not even allowed to consider a majority of the available predictors.
- Thus the random forest *de-correlates* the bagged trees leading to more reduction in variance.

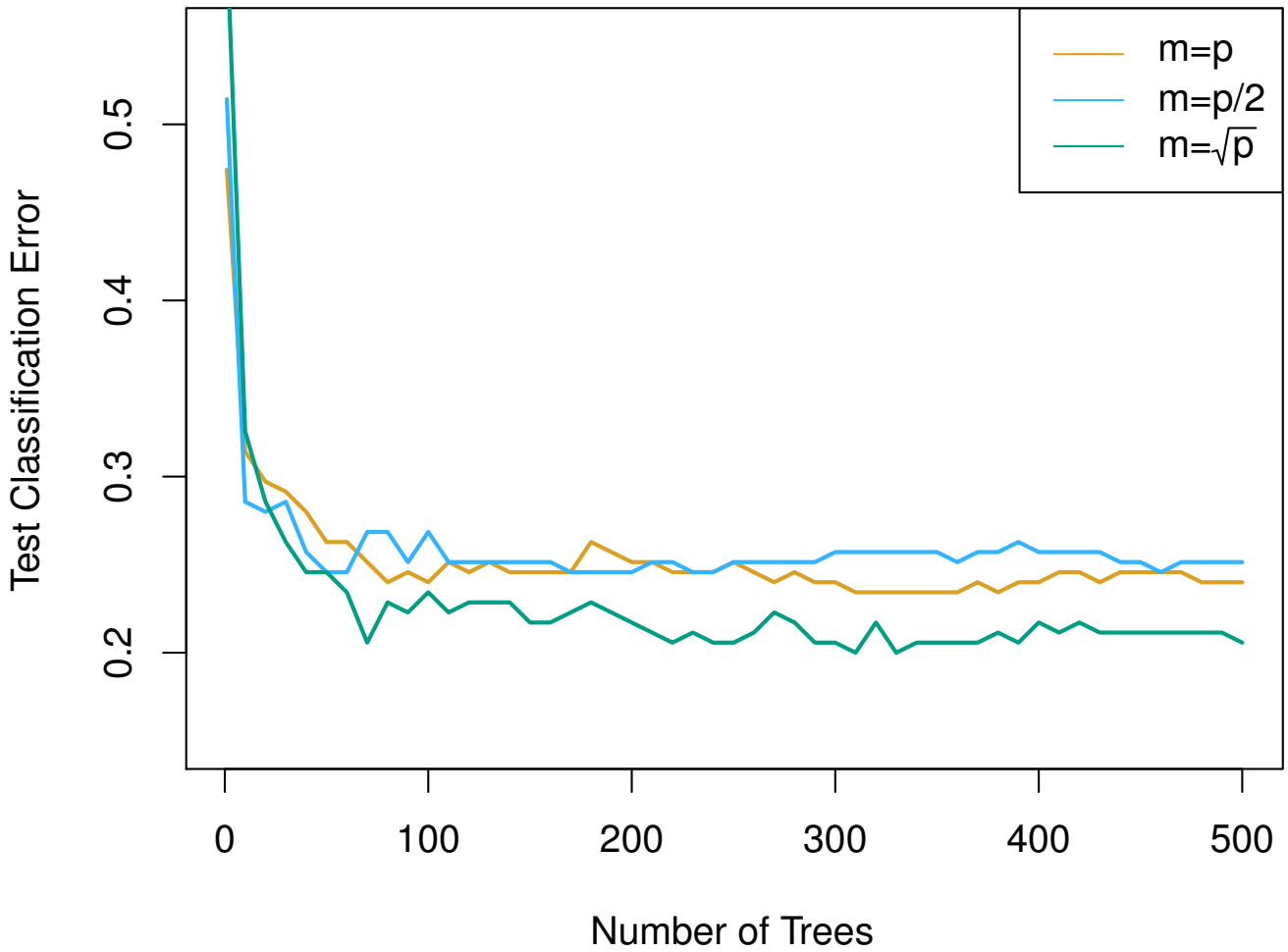
- Using a small value of m in building a random forest will typically be helpful when we have a large number of correlated predictors.

Gene Expression Data

- A high-dimensional data set consisting of expression measurements of 4718 genes measured on tissue samples from 349 patients.
- Each of the patient samples has a qualitative label with 15 different levels: either normal or one of 14 different types of cancer.
- We use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.
- We randomly divided the observations into a training and a test set, and applied random forests to the training set for three different values of the number of splitting variables m.

Random Forest with Different m

Gene Expression Data



Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification, not limited to decision trees.
- For bagging, each tree is built on a bootstrap data set, independent of the other trees.
- Boosting works in a similar way, except that the trees are grown *sequentially*: each tree is grown using information from previously grown trees.

Boosting Algorithm for Regression Trees

1. Set $f(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   1. Fit a tree $\hat{f}^b$ with d splits ($d + 1$ leafs) to the training data (X, r).
   2. Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \longleftarrow \hat{f}(x) + \lambda \hat{f}^b$$

   3. Update the residuals:

$$r_i \longleftarrow r_i - \lambda \hat{f}^b(x_i)$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

What Is the Idea Behind Boosting?

1. Unlike fitting a single large decision tree to the data, which amounts to fitting the data hard and potentially overfitting, the boosting approach instead *learns slowly*.

2. Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function in order to update the residuals.

3. Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm.

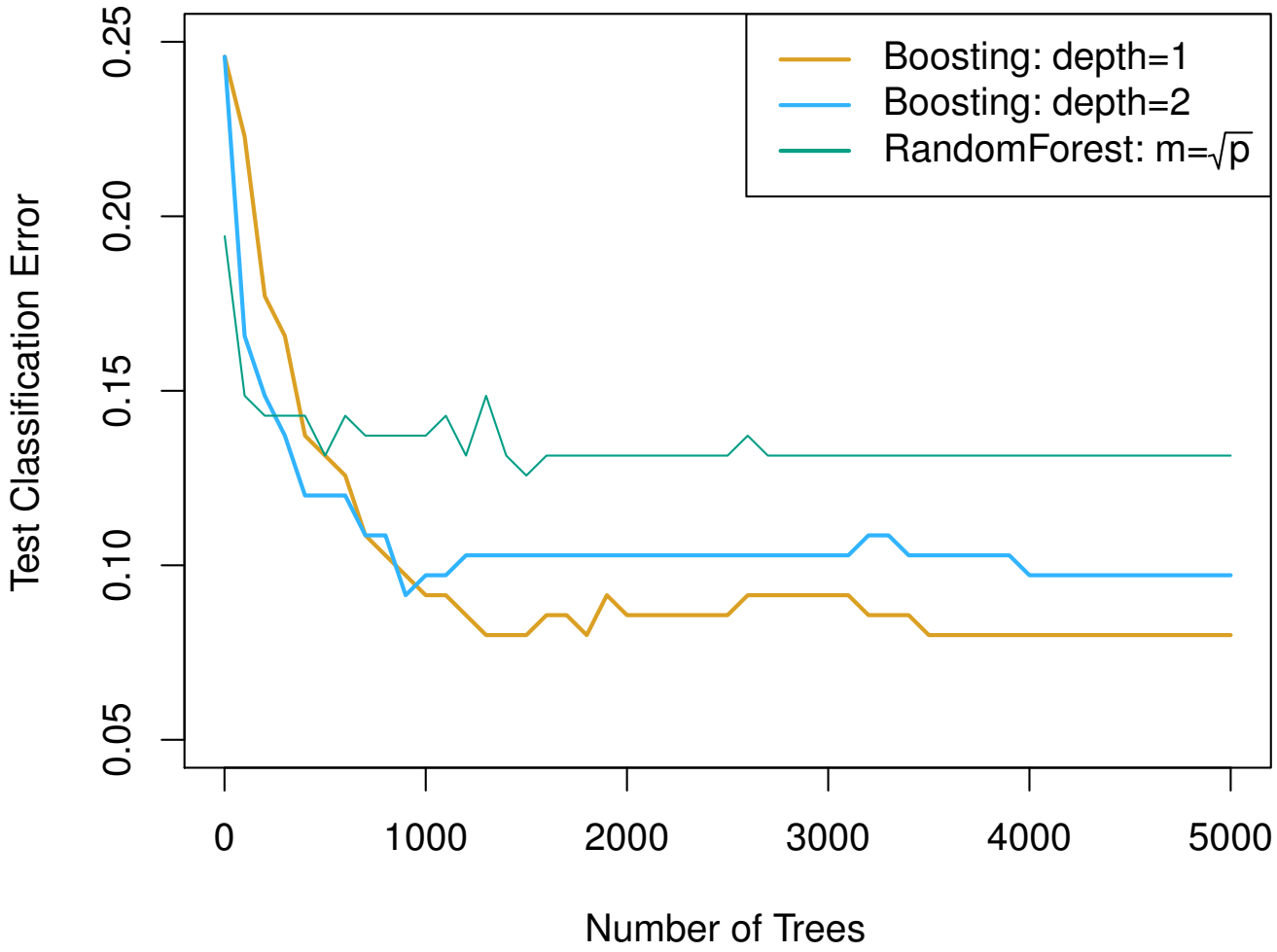
4. *By fitting small trees to the residuals, we slowly improve* $\hat{f}$ *in areas where it does not perform well.* The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.

Tuning Parameters for Boosting

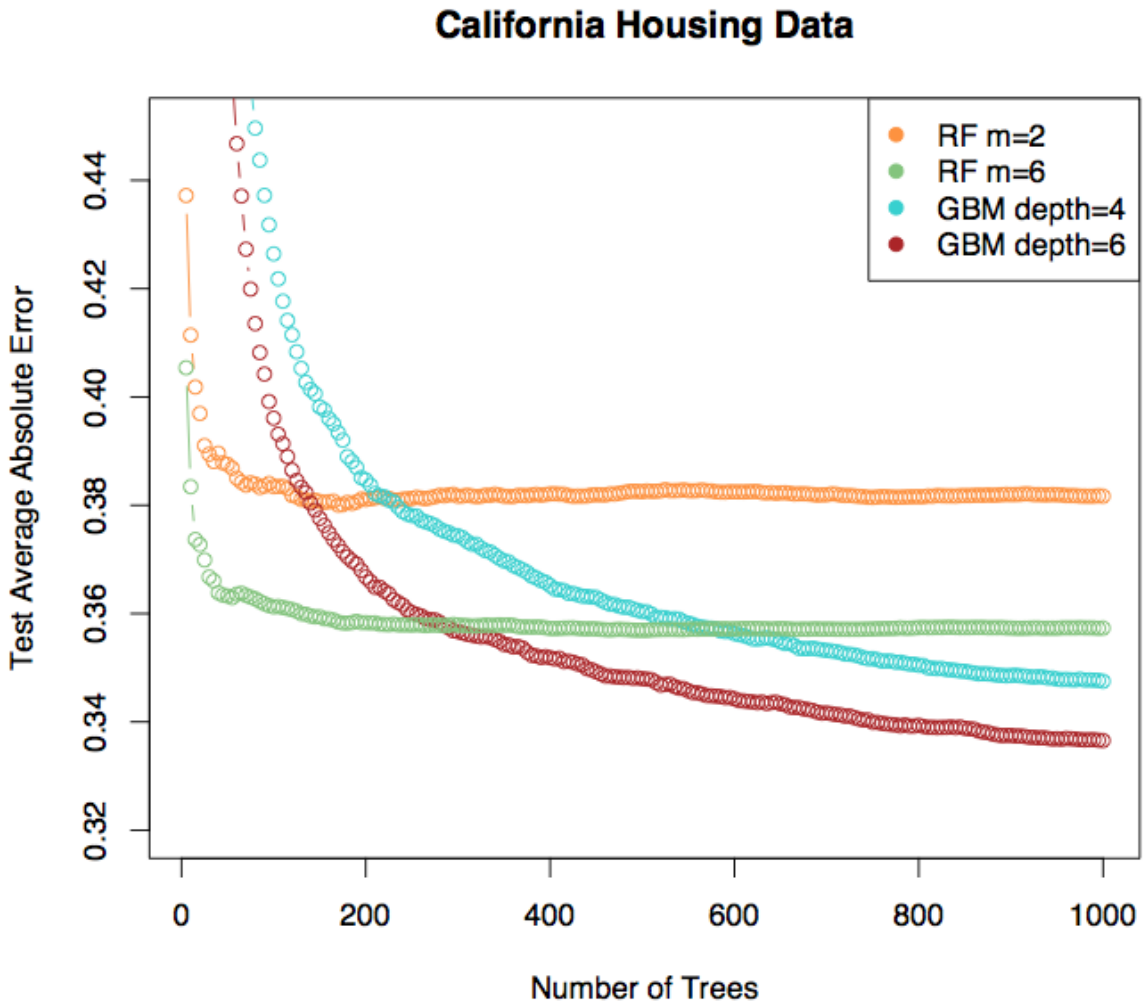
- The number of trees B.
  - Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all.
  - We use cross-validation to select B.
- The shrinkage parameter λ, a small positive number.
  - This controls the rate at which boosting learns.
  - Typical values are 0.01 or 0.001; choice can depend on the problem.
  - Very small λ can require a very large B to achieve good performance.
- The number of splits d in each tree.
  - d controls the complexity of the boosted ensemble.
  - Often $d = 1$ works well, in which case each tree is a stump, consisting of a single split and resulting in an additive model.
  - More generally d is the interaction *depth*, and controls the interaction order of the boosted model — d splits can involve at most d variables.

Boosting with Different Depth d

Gene Expression Data: to predict cancer versus normal.

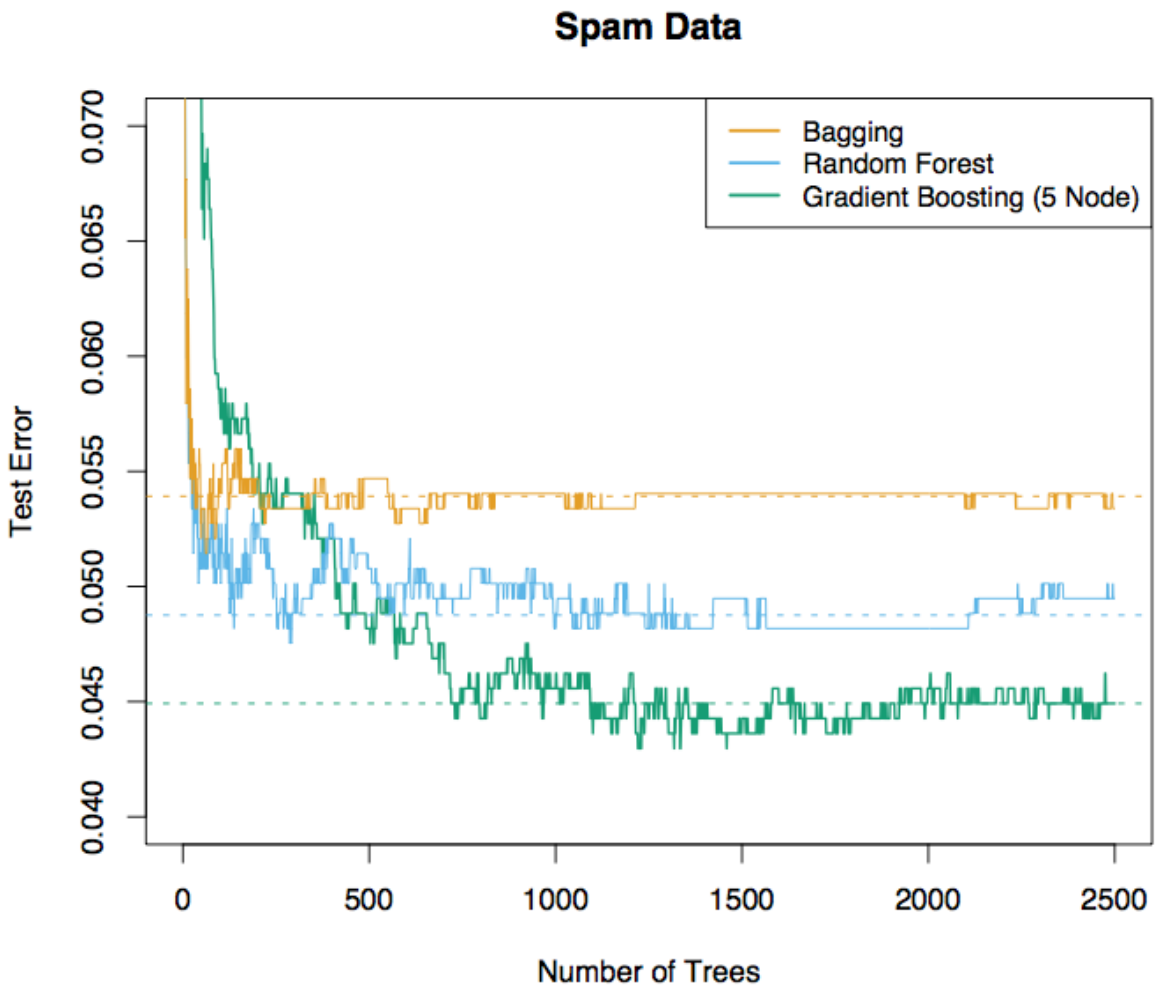


A Regression Example



from *Elements of Statistical Learning*, Chapter 15.

Another Classification Example



from *Elements of Statistical Learning*, Chapter 15.