MY474: Applied Machine Learning for Social Science

Lecture 7: Ensemble Methods, Bagging, Random Forests, Boosting

Blake Miller

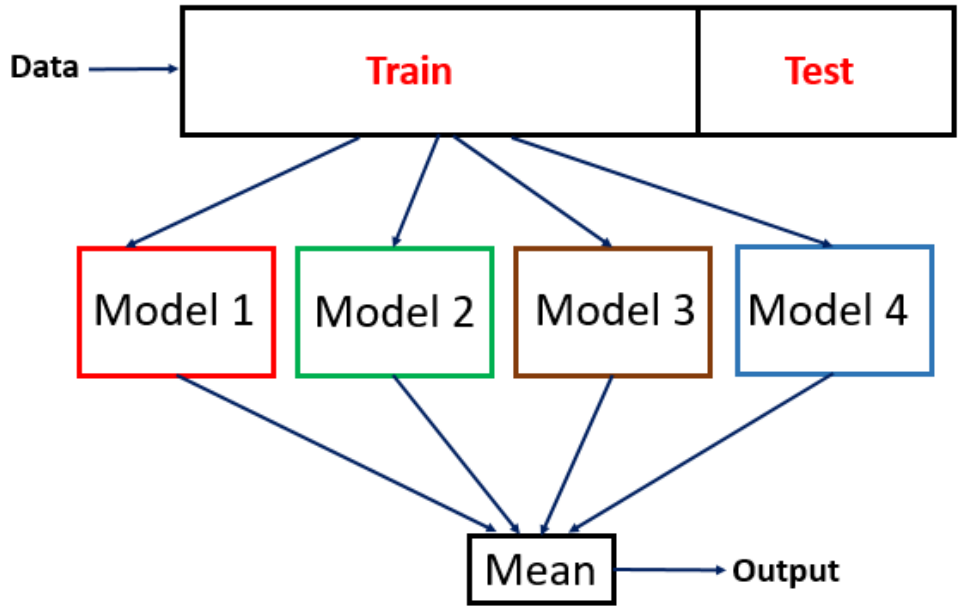
27 November 2019

Agenda

1. Ensembles
2. The Bootstrap
3. Bagging
4. Random Forests
5. Boosting

Ensembles

Simple Ensembles



- The idea of ensemble learning is to build a prediction model by combining the strengths of a collection of simpler base models.
- Ensembles are constructed using a **committee** of learners whose output are combined to formulate a decision
- For a regression task, this might involve taking the **mean** of each learner's output $\hat{y}_1 \ldots \hat{y}_C$
- For a classification task, we would take the **majority vote** of each learner's output

Ensemble Classifiers Applied to Trees

- Classification trees are simple but often unstable. Solution: combine many trees
- Bagging (Breiman 1996): Fit many large trees to bootstrap resampled versions of the training data, and classify by majority vote.
- Random forests (Breiman 1999): Improvements over bagging with randomized trees.
- Boosting (Freund & Schapire 1996): Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote.

The Bootstrap

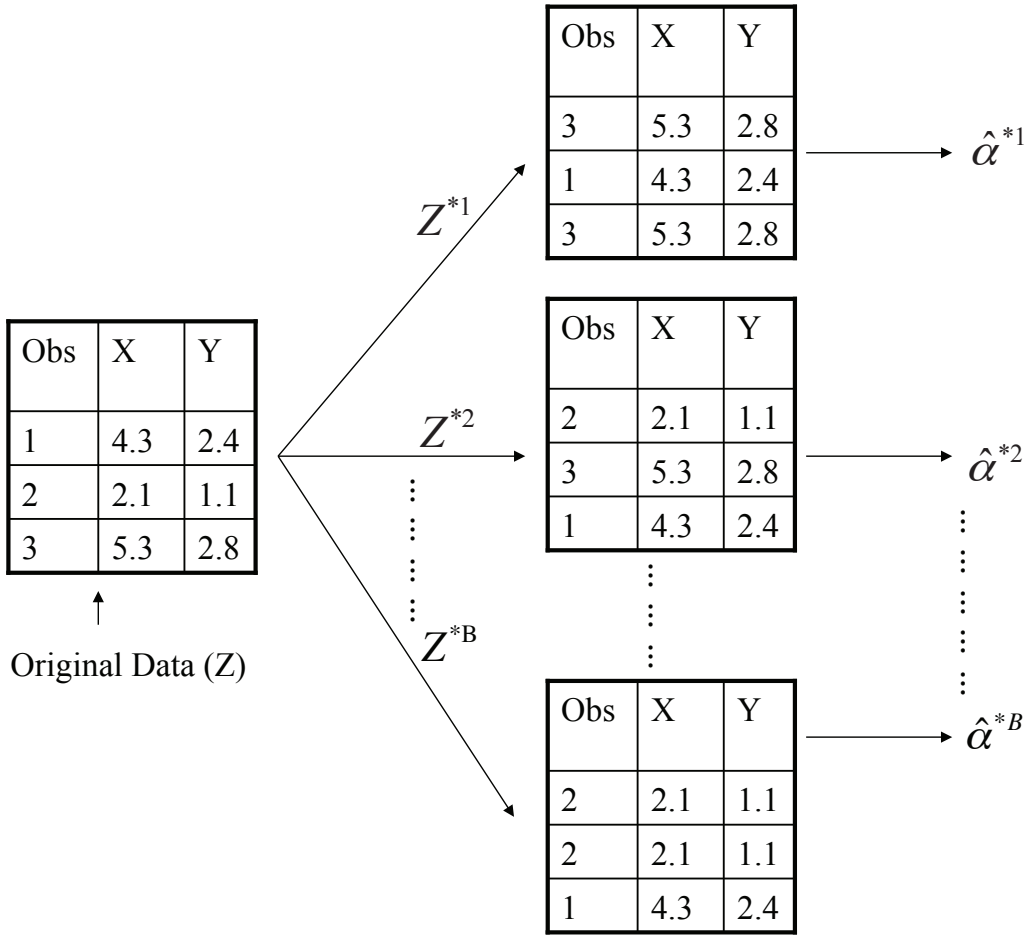
The Bootstrap

- The bootstrap is a flexible and powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- For example, it can provide an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.

Where does the name came from?

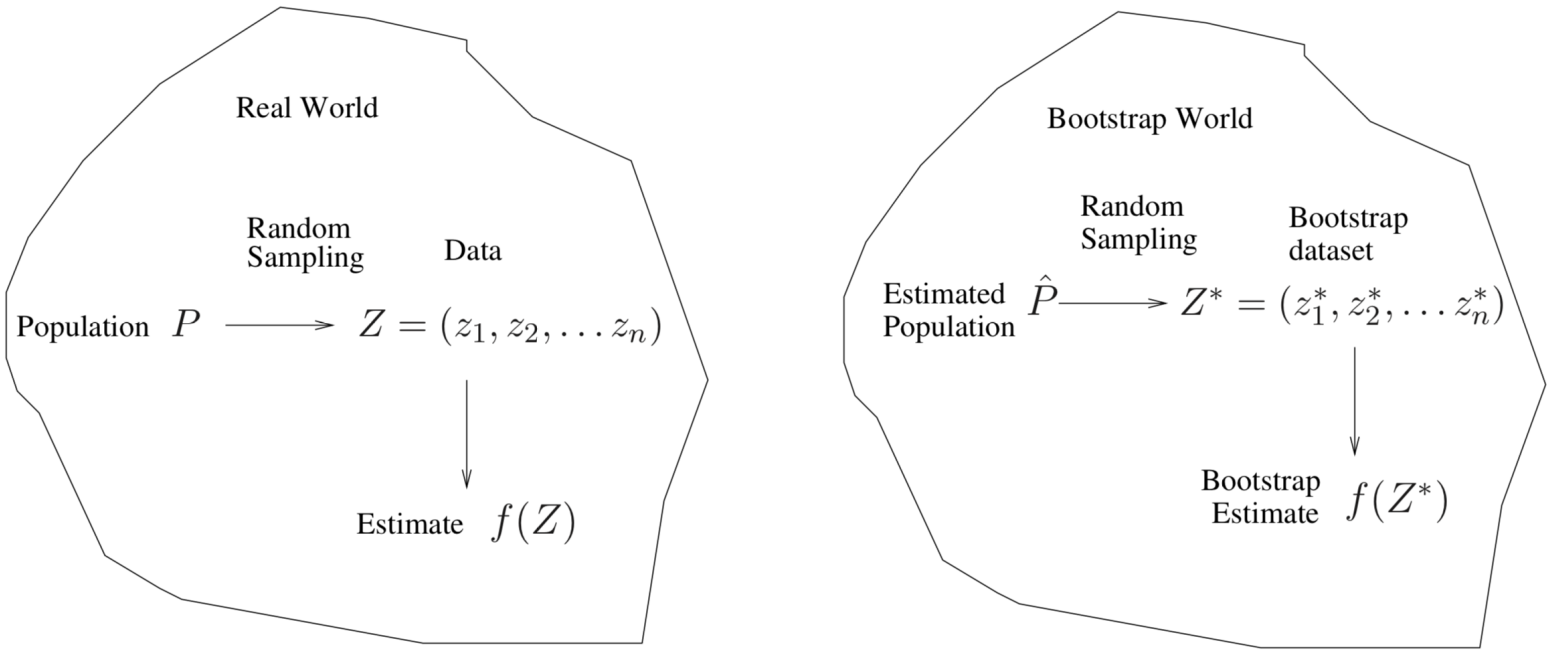
- The use of the term bootstrap derives from the phrase to pull oneself up by one's bootstraps, widely thought to be based on one of the eighteenth century "The Surprising Adventures of Baron Munchausen" by Rudolph Erich Raspe: The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps.
- It is not the same as the term "bootstrap" used in computer science meaning to "boot" a computer from a set of core instructions, though the derivation is similar.

Example with just 3 observations



A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of α

A general picture for the bootstrap



Bootstrap procedure

- Denoting the first bootstrap data set by Z^{*1}, we use Z^{*1} to produce a new bootstrap estimate for α, which we call $\hat{a}^{*1}$
- This procedure is repeated B times for some large value of B (say 100 or 1000), in order to produce B different bootstrap data sets, $Z^{*1}, Z^{*2}, \ldots, Z^{*B}$, and B corresponding α estimates, $\hat{a}^{*1}, \hat{a}^{*2}, \ldots, \hat{a}^{*B}$.
- We estimate the standard error of these bootstrap estimates using the formula

$$SE_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}(\hat{\alpha}^{*r} - \bar{\hat{\alpha}}^{*})^2}$$

- This serves as an estimate of the standard error of $\hat{\alpha}$ estimated from the original data set.

Bagging

Bagging

- **Bootstrap aggregation**, or **bagging**, is a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.
- Recall that given a set of n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by σ^2/n.
- In other words, **averaging a set of observations reduces variance**. Of course, this is not practical because we generally do not have access to multiple training sets.

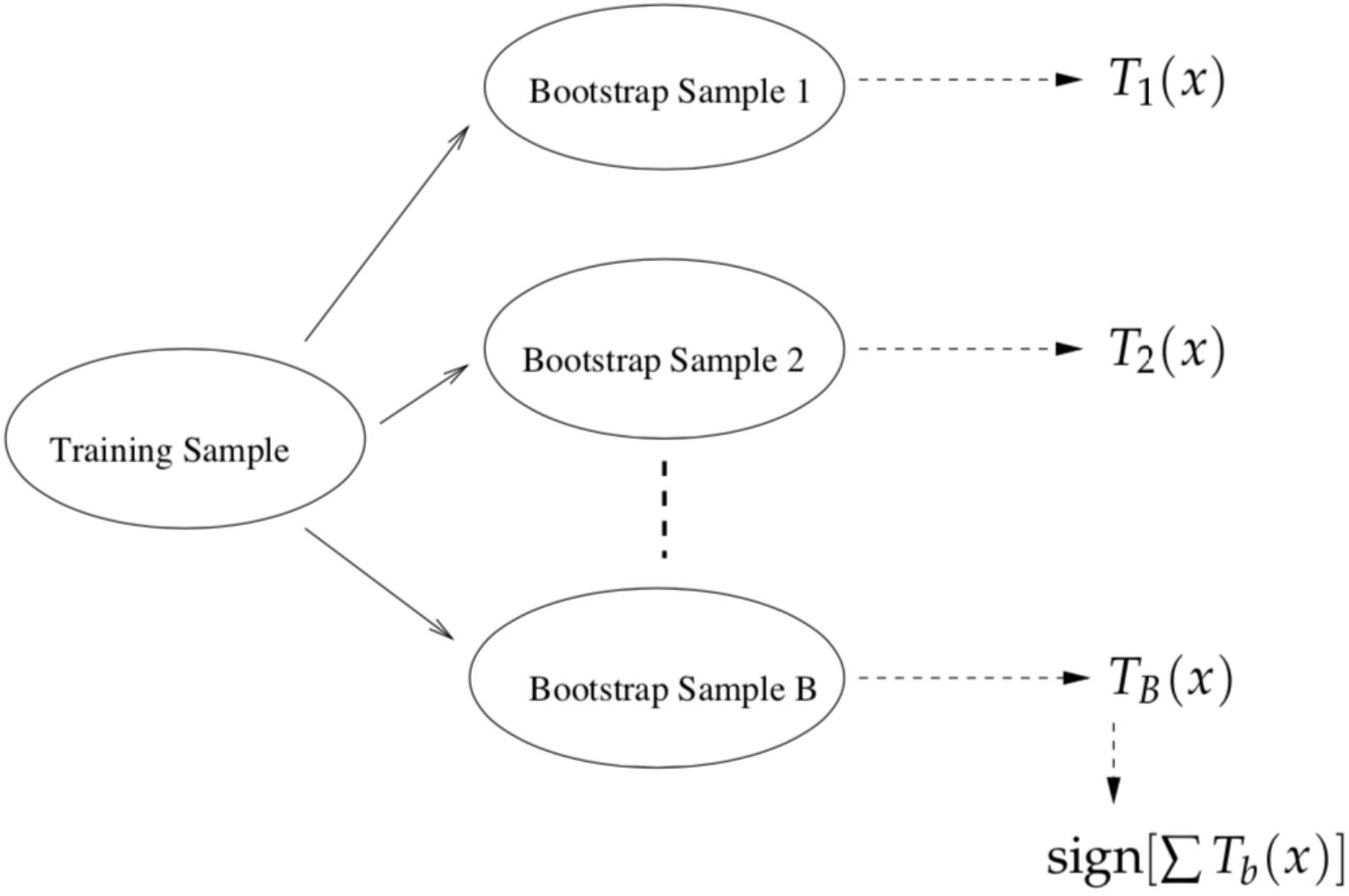
Bagging— continued

- Instead, we can bootstrap, by taking repeated samples from the (single) training data set.
- In this approach we generate B different bootstrapped training data sets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, the prediction at a point x. We then average all the predictions to obtain

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

This is called **bagging**.

Schematics of Bagging with Trees



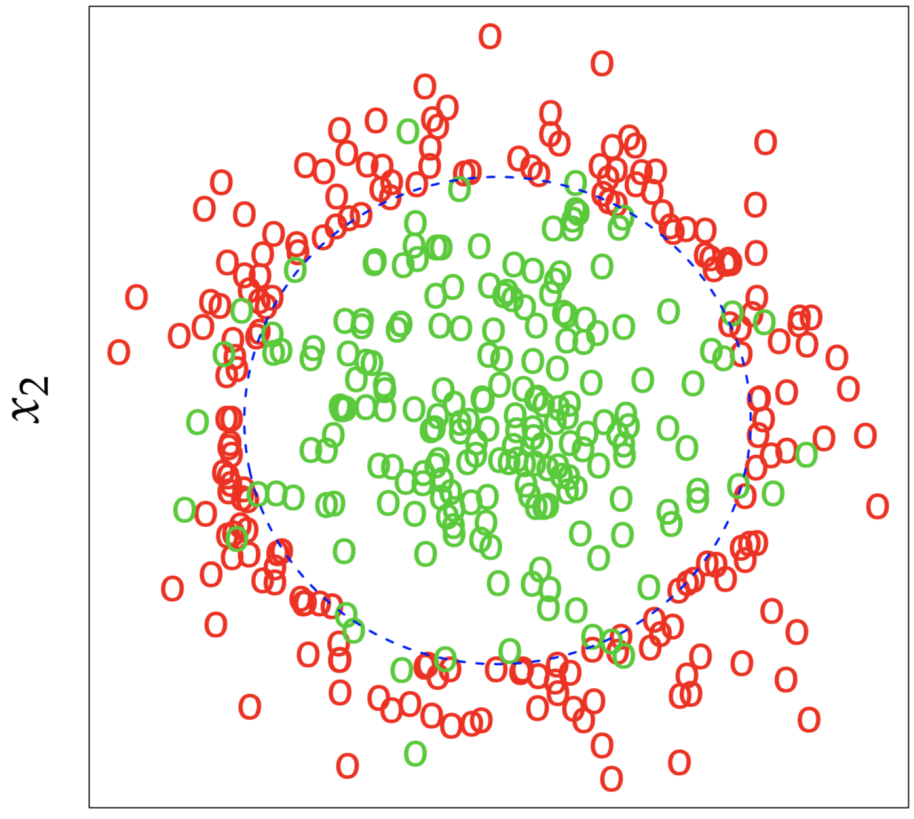
Schematics of Bagging with Trees

- Bagging or **bootstrap aggregation** averages a given procedure over many samples, to **reduce its variance**.
- Sample with replacement from the training data $(x_1, y_1), \ldots, (x_n, y_n)$ to obtain a bootstrap sample $(x_1^*, y_1^*), \ldots, (x_n^*, y_n^*)$.
- Construct a new tree T_1^*.
- Repeat everything B times, obtaining B trees $T_1^*, \ldots, T_B^*$.
- Given a new point x, classify by majority vote among $T_1^*(x), \ldots, T_B^*(x)$.

Schematics of Bagging with Trees

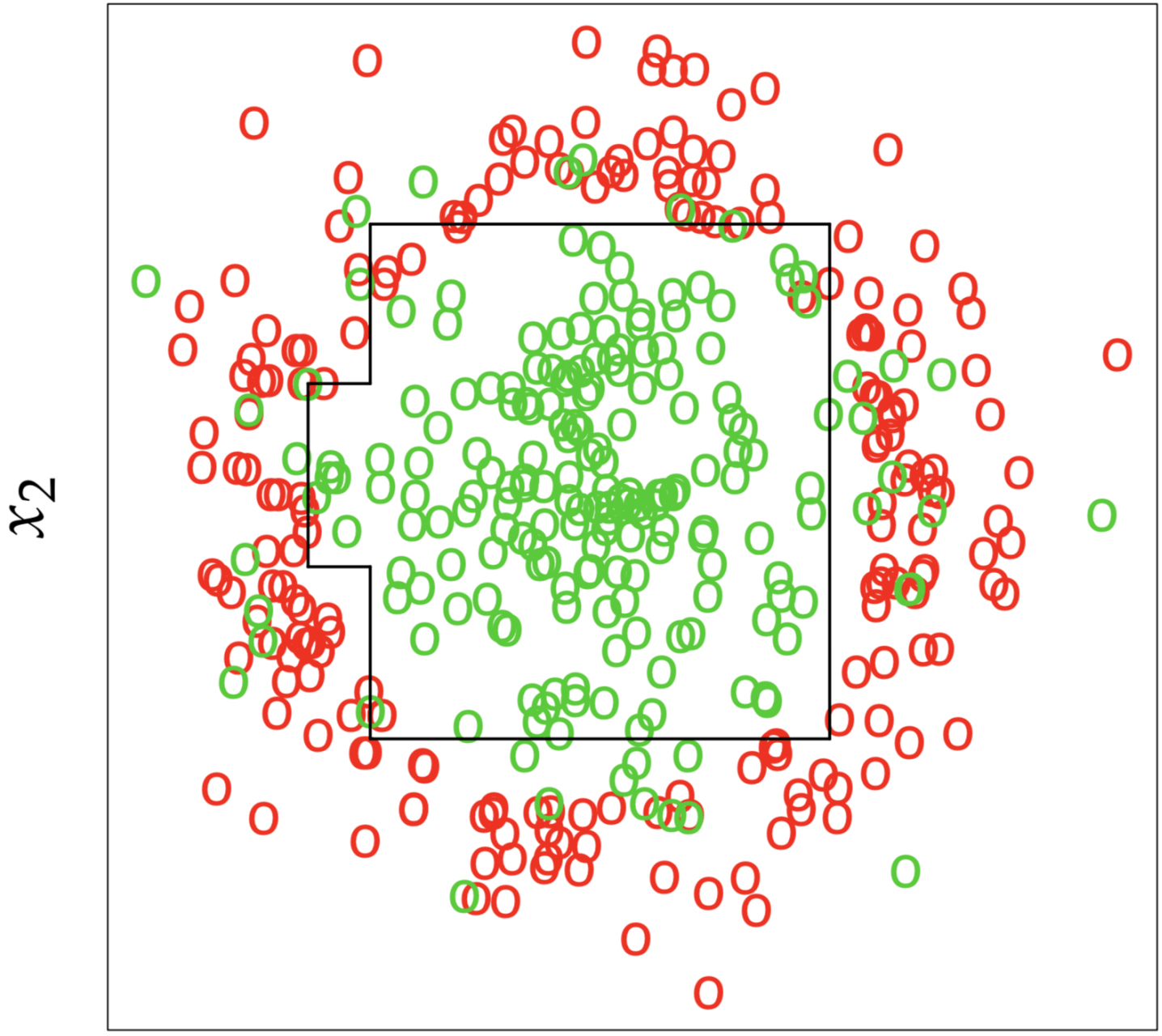
- No need to prune, pruning reduces variance and increases bias
- The process of averaging bootstrap samples reduces variance, so pruning is unnecessary
- For regression trees: for each test observation, we return the average the predictions of each of the B trees as our bagging estimate.
- For classification trees: for each test observation, we record the class predicted by each of the B trees, and take a **majority vote**: the overall prediction is the most commonly occurring class among the B predictions.

Example: Donut Data

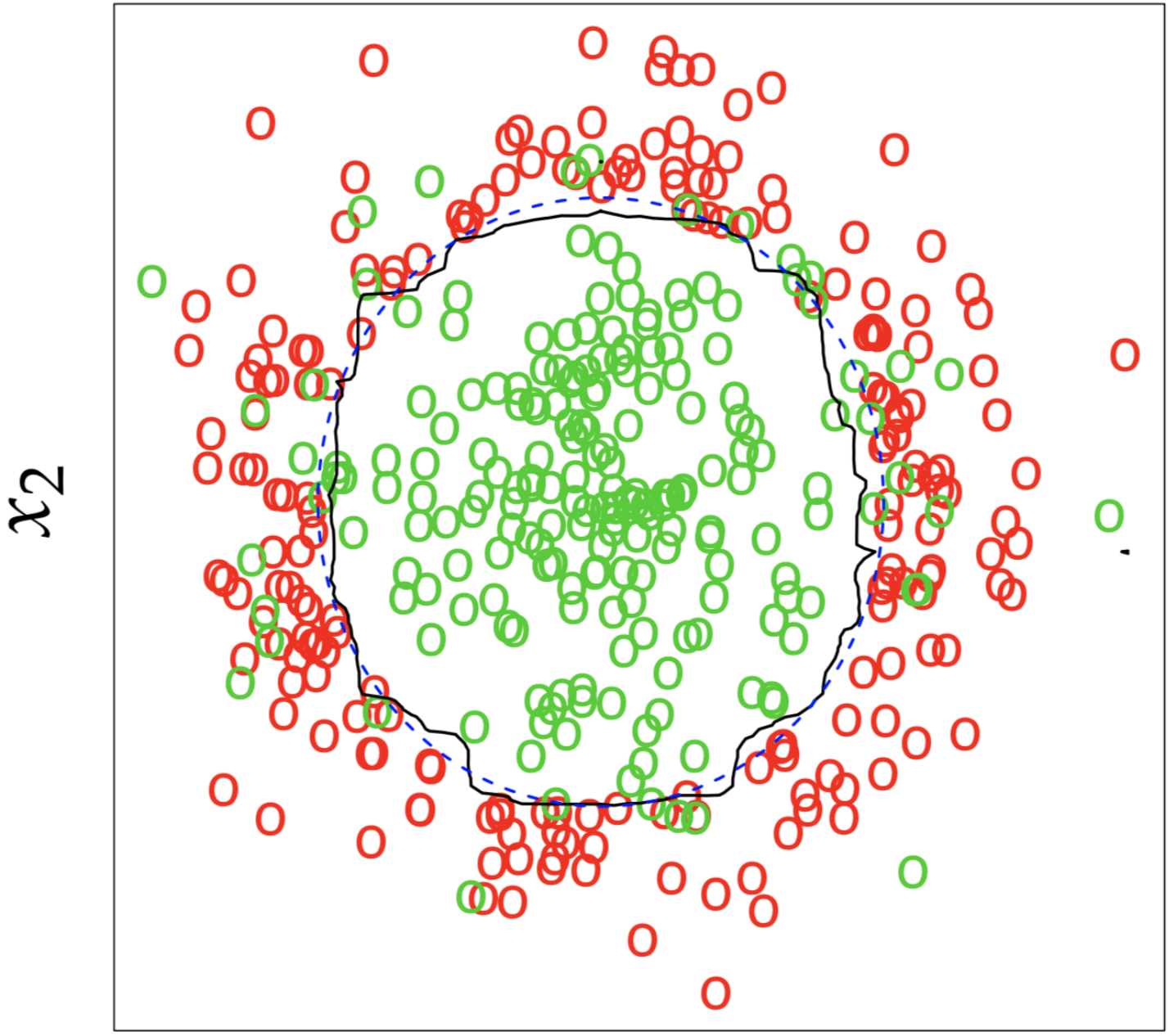


- Green class: two independent standard normal inputs X1, X2
- Red class: conditional on $X_1^2 + X_2^2 \geq 4.6$

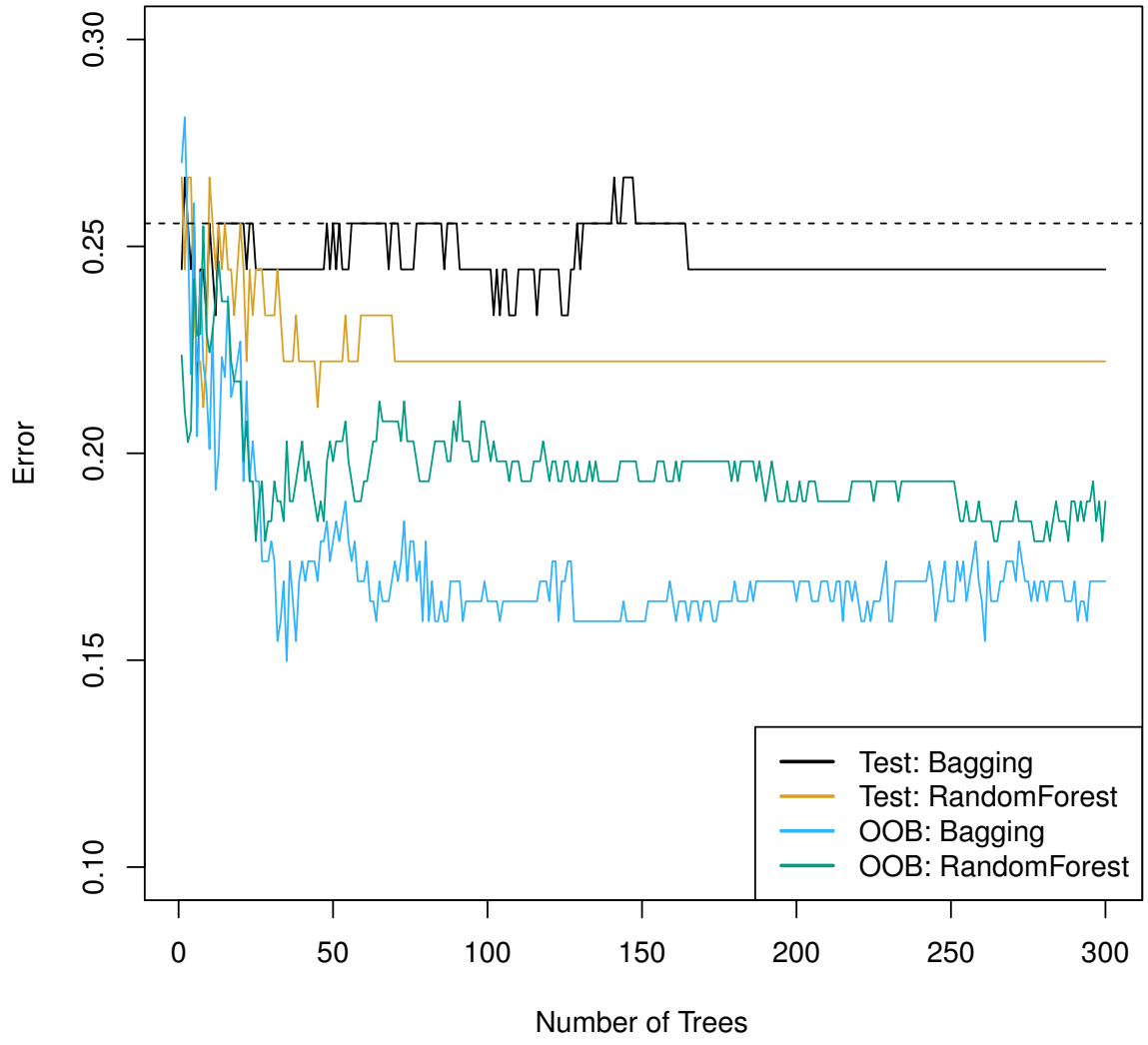
Donut Data: Single tree decision boundary



Donut Data: Bagging decision boundary



Bagging the heart data



Bagging and random forest results for the Heart data.

Details of previous figure

- The test error (black and orange) is shown as a function of B, the number of bootstrapped training sets used.
- Random forests were applied with $m = \sqrt{p}$.
- The dashed line indicates the test error resulting from a single classification tree.
- The green and blue traces show the **OOB error**, which in this case is considerably lower

Out-of-Bag Error Estimation

- It turns out that there is a very straightforward way to estimate the test error of a bagged model.
- Recall that the key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations. One can show that on average, each bagged tree makes use of around two-thirds of the observations.
- The remaining one-third of the 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.
- This estimate is essentially the LOO cross-validation error for bagging, if B is large.

Remarks on Bagging

- Bagging can dramatically reduce the variance of a classification procedure, but does not affect its bias.
- Bagging a good classifier (e.g. a decision tree) usually makes it better, but bagging a bad classifier (e.g. a random decision rule) does not, and can make it worse.
- Interpretable structure of a tree is lost.

Random forests

Random forests: main idea

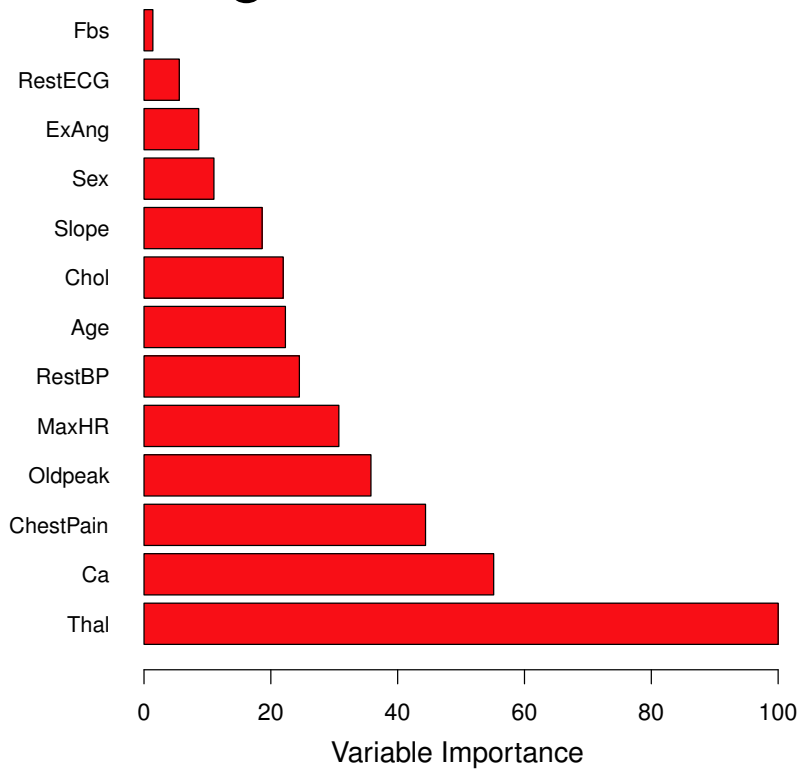
- Create B boostrapped training sets as in bagging
- Grow a decision tree on each bootstrapped training data set, but consider a **random subset of variables** at each split. The trees are not pruned.
- Use majority vote among all the trees to classify a new object.

Random Forests

- **Random forests** provide an improvement over bagged trees by way of a small tweak that **decorrelates** the trees. This reduces the variance when we average the trees.
- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, **a random selection of m predictors** is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.
- A fresh selection of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$ — that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors (4 out of the 13 for the Heart data).

Variable importance measure

- For bagged/RF regression trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.
- Similarly, for bagged/RF classification trees, we add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.



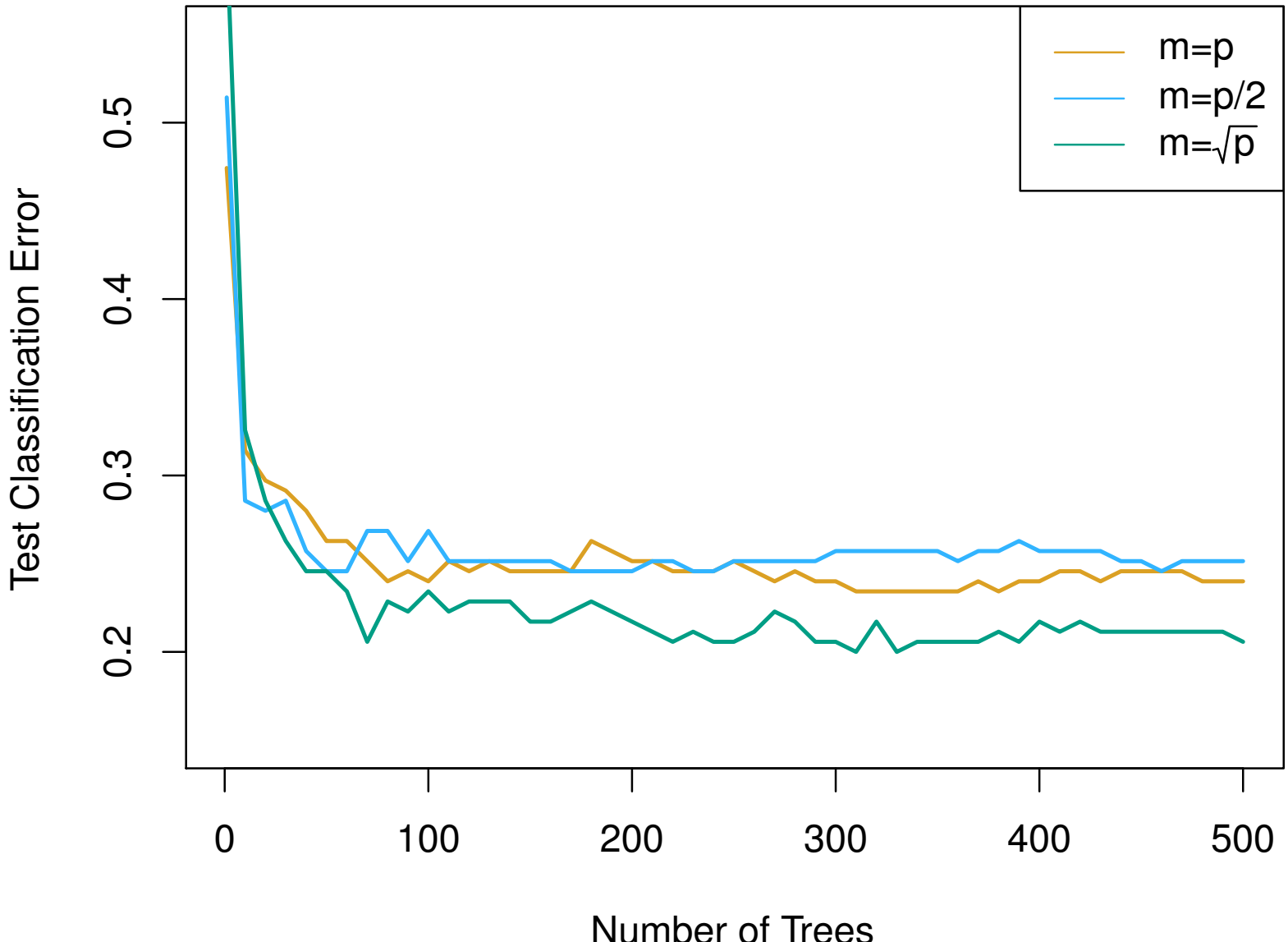
Variable importance in detail

- A forest has no interpretation (unlike a single tree)
- A substitute: variable importance can be computed for each variable
  - For each b, permute the values of the variable in question in the OOB sample at random
  - Classify the original OOB sample and the permuted OOB sample using the tree T_b
  - Measure drop in classification performance (using either classification error or gini)
  - Average over all bootstrap samples B
- Final output: increase in error for each variable if it is randomly permuted

Example: gene expression data

- We applied random forests to a high-dimensional biological data set consisting of expression measurements of 4,718 genes measured on tissue samples from 349 patients.
- There are around 20,000 genes in humans, and individual genes have different levels of activity, or expression, in particular cells, tissues, and biological conditions.
- 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.

Results: gene expression data



Details of previous figure

- Results from random forests for the fifteen-class gene expression data set with $p = 500$ predictors.
- The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of m, the number of predictors available for splitting at each interior tree node.
- Random forests ($m < p$) lead to a slight improvement over bagging ($m = p$). A single classification tree has an error rate of 45.7%.

Remarks on random forests

- Random forests are considered one of the most competitive classifiers and are popular
- Random selection of variables controls overfitting
- For most data sets results seem not too sensitive to m, the number of variables used for splitting
- While there is no interpretation, variables can be ranked by importance
- However, importance rankings can be much more variable that the classifcation results themselves

Boosting

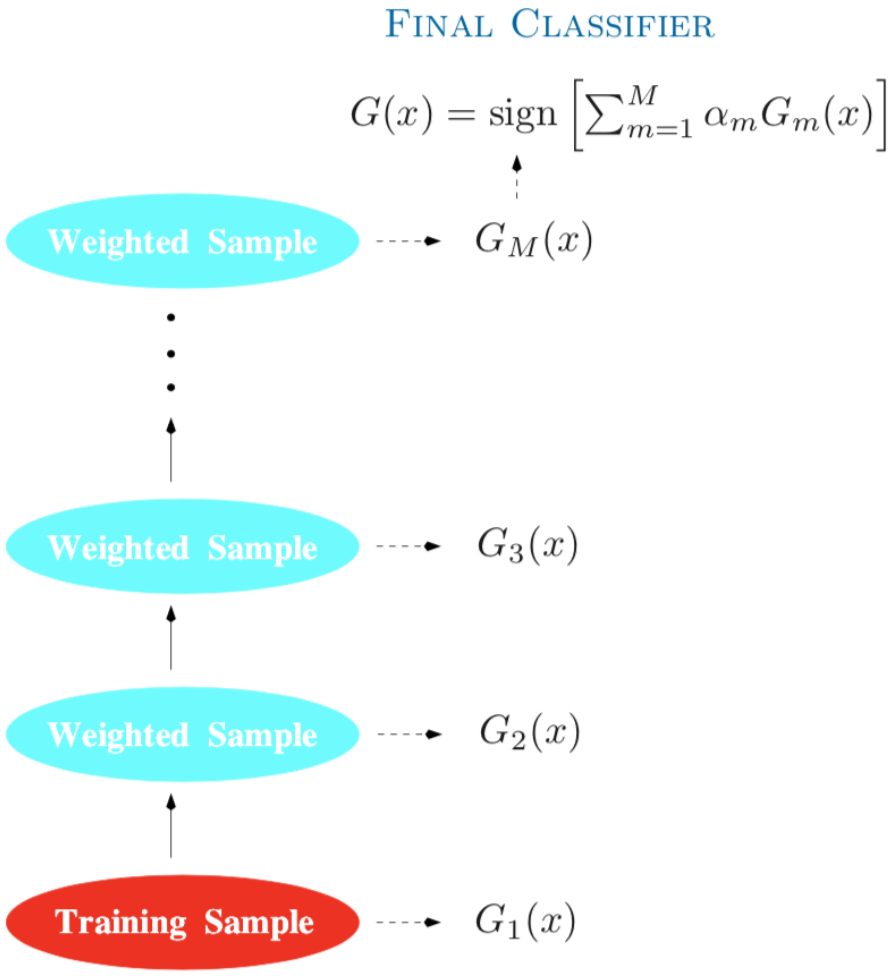
Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification. We only discuss boosting for decision trees.
- Recall that bagging involves creating multiple copies of the original training data set using the bootstrap, fitting a separate decision tree to each copy, and then combining all of the trees in order to create a single predictive model.
- Notably, 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

- Focus new learners on examples that others get wrong
- Train leamers sequentially
- Errors of early predictions indicate the “hard” examples
- Focus later predictions on getting these examples right
- Combine the whole set in the end
- Convert many “weak” leamers into a complex predictor

Schematics of Boosting (classification)



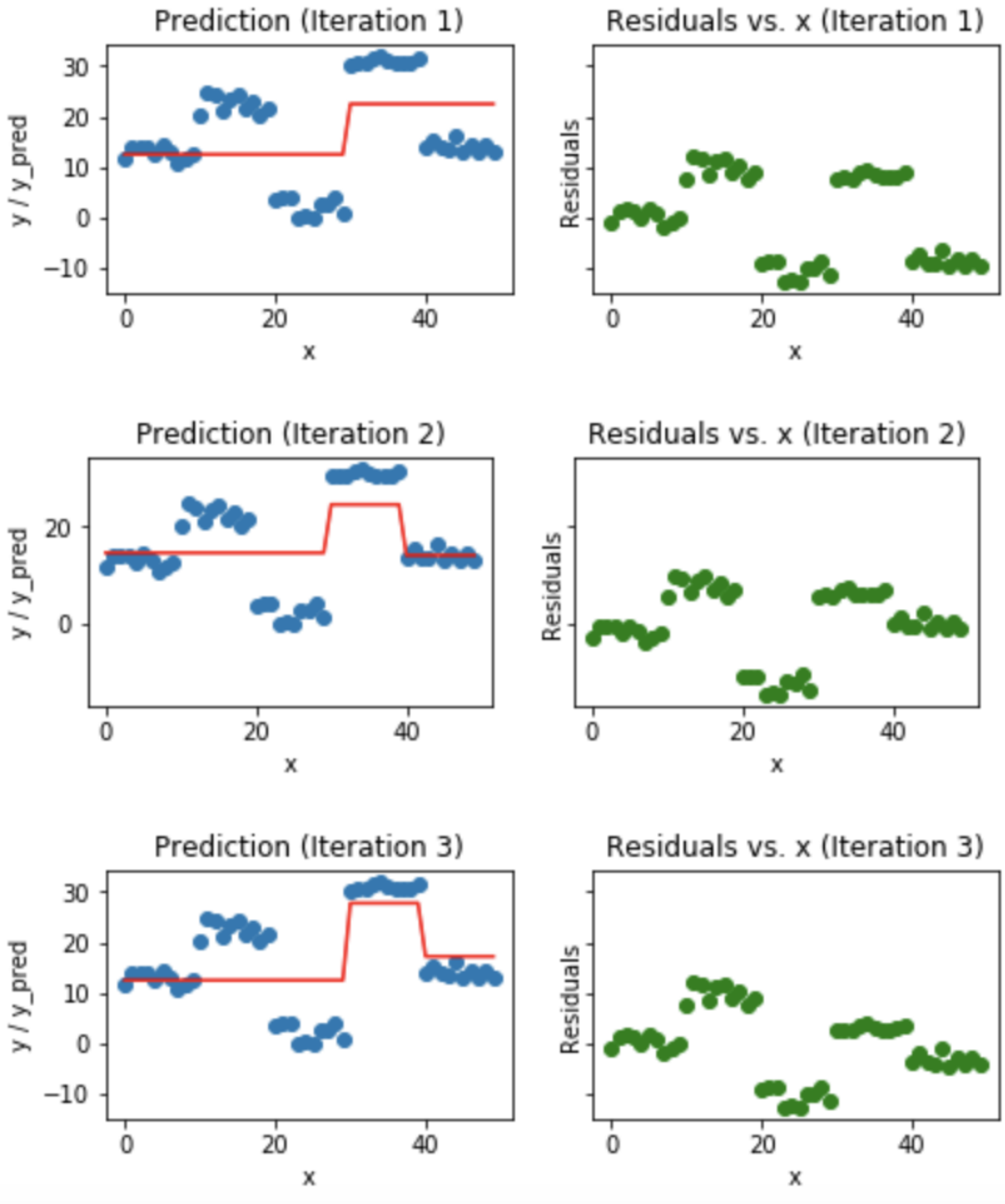
Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.

Boosting algorithm for regression trees

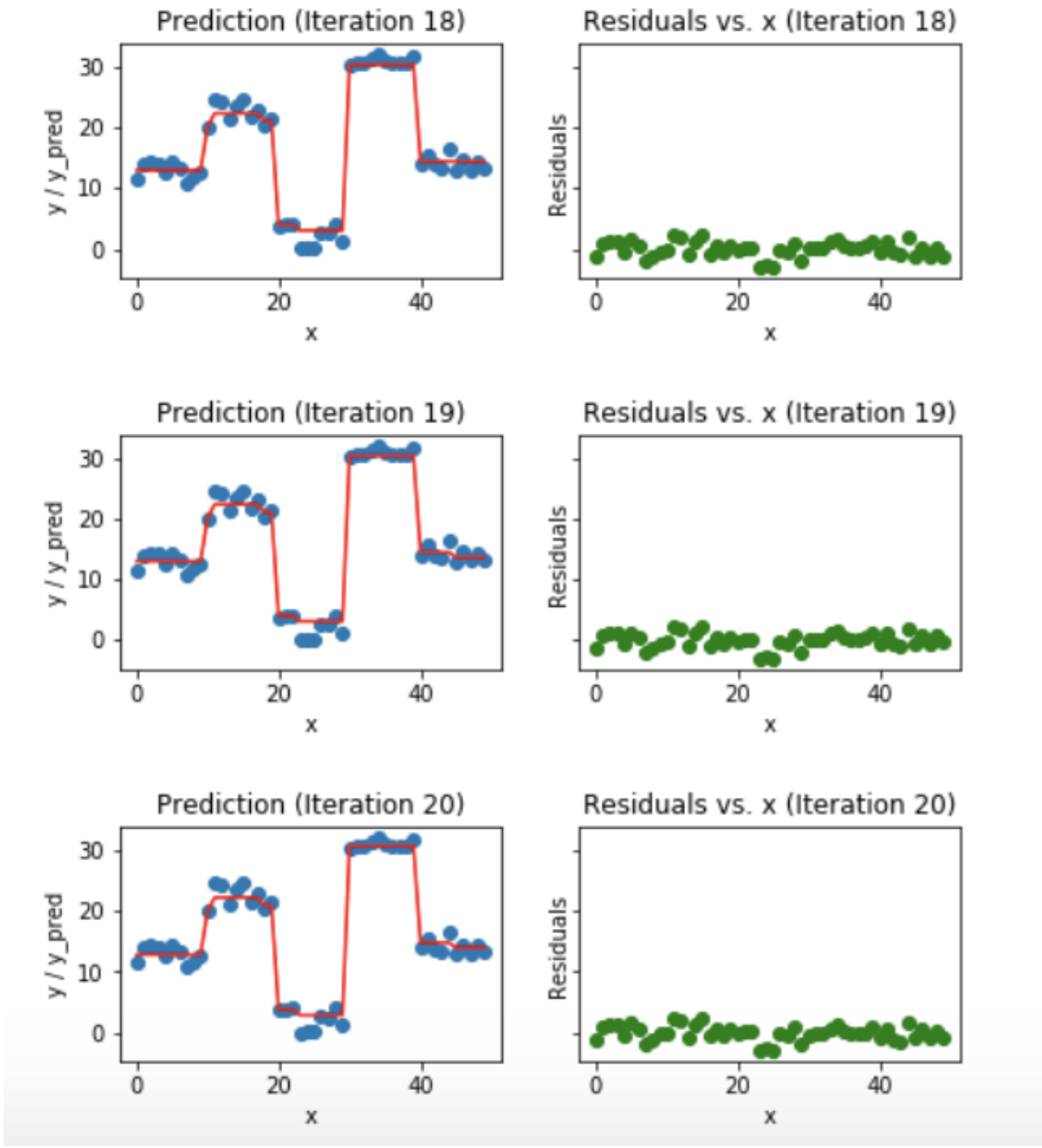
1. Set $f(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   - Fit a tree $\hat{f}_b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r).
   - Update $\hat{f}$ by adding in a shrunken version of the new tree: $\hat{f}(x) \leftarrow \hat{f}(x)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

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

Visualizing Boosting



Visualizing Boosting



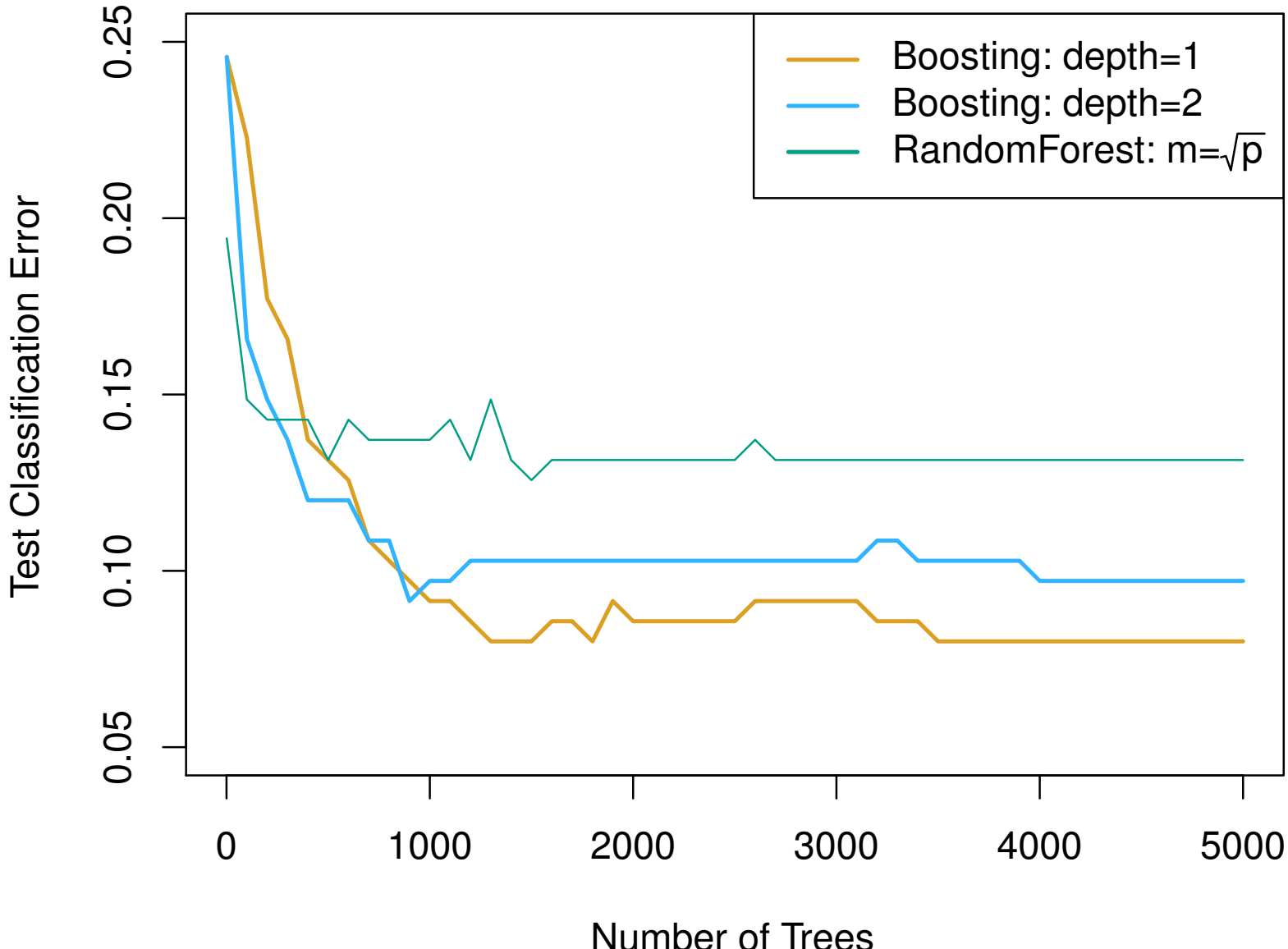
What is the idea behind this procedure?

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

Boosting for classification

- Boosting for classification is similar in spirit to boosting for regression, but is a bit more complex. We will not go into detail here, nor do we in the text book.
- Students can learn about the details in **Elements of Statistical Learning, chapter 10**.
- The R package `gbm` (gradient boosted models) handles a variety of regression and classification problems.

Gene expression data continued



Details of previous figure

- Results from performing boosting and random forests on the fifteen-class gene expression data set in order to predict cancer versus normal.
- The test error is displayed as a function of the number of trees. For the two boosted models, $\lambda = 0.01$. Depth-1 trees slightly outperform depth-2 trees, and both outperform the random forest, although the standard errors are around 0.02, making none of these differences significant.
- The test error rate for a single tree is 24%.

Tuning parameters for boosting

1. 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.
2. The **shrinkage parameter** λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.
3. The **number of splits** d in each tree, which 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, since d splits can involve at most d variables.

Remarks on boosting

- Boosting works well with trees, but can in principle be applied to any classifier
- Boosting can overfit, so it is important to limit B (the total number of iterations) and the maximum allowed depth of each tree
- Interpretable structure is also lost
- Multi-class extensions are available

Summary

- Decision trees are simple and interpretable models for regression and classification
- However they are often not competitive with other methods in terms of prediction accuracy
- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting ensemble of trees.
- The latter two methods— random forests and boosting— are among the state-of-the-art methods for supervised learning. However their results can be difficult to interpret.