Machine Learning and Data Analytics
ME 5013- Fall 2019

Lectures 16

- Bagging
- Random Forests
- Boosting

UTSA®

The University of Texas at San Antonio™

Adel Alaeddini, PhD

Associate Professor of Mechanical Engineering

Advanced Data Engineering Lab

adel.alaeddini@utsa.edu

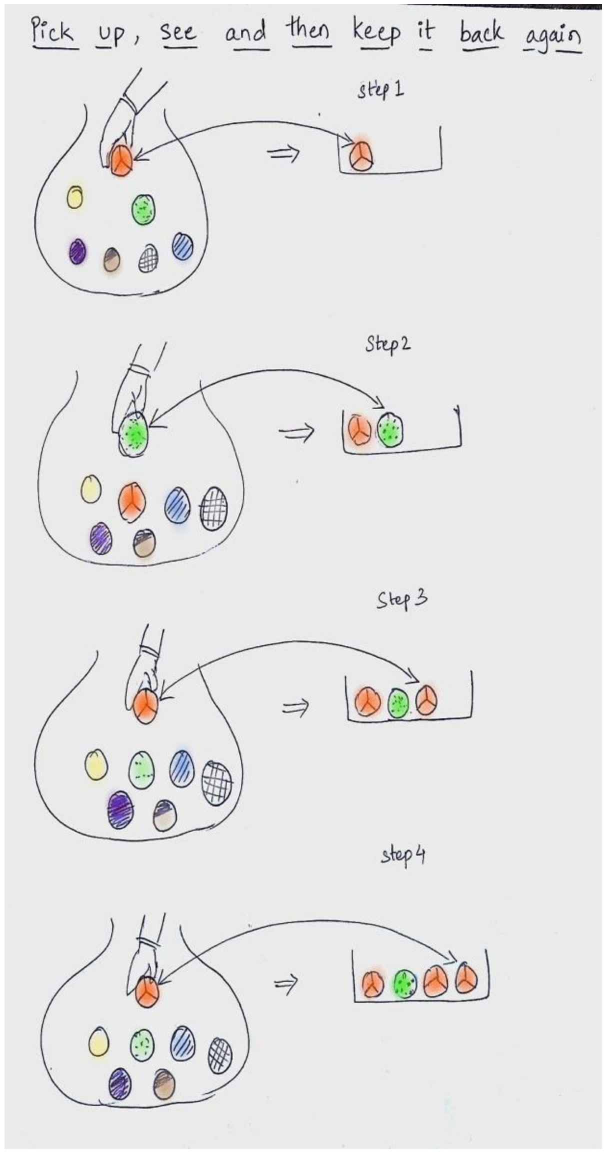
Disclosure: the slides are adopted from various online resources

Pros and Cons

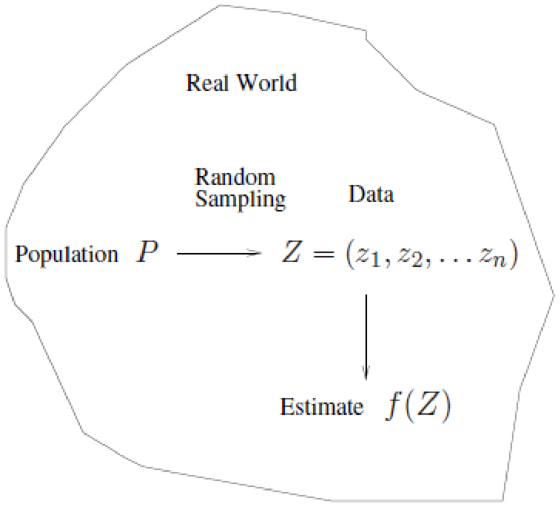
- Tree-based methods are simple and useful for interpretation.
- However they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy.
- *Bagging*, *random forests*, and *boosting* grow multiple trees which are then combined to yield a single consensus prediction.
- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss interpretation.

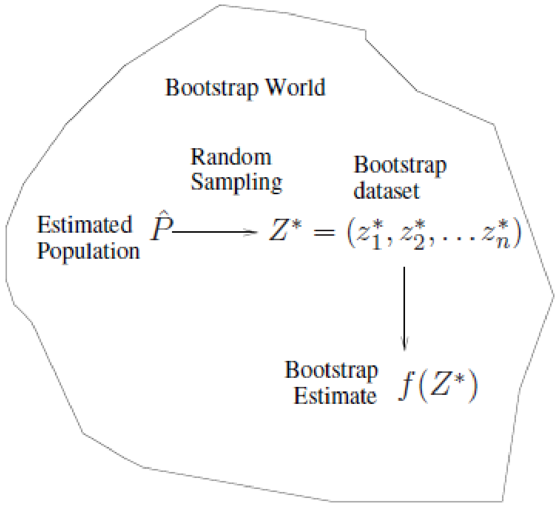
Bootstrap

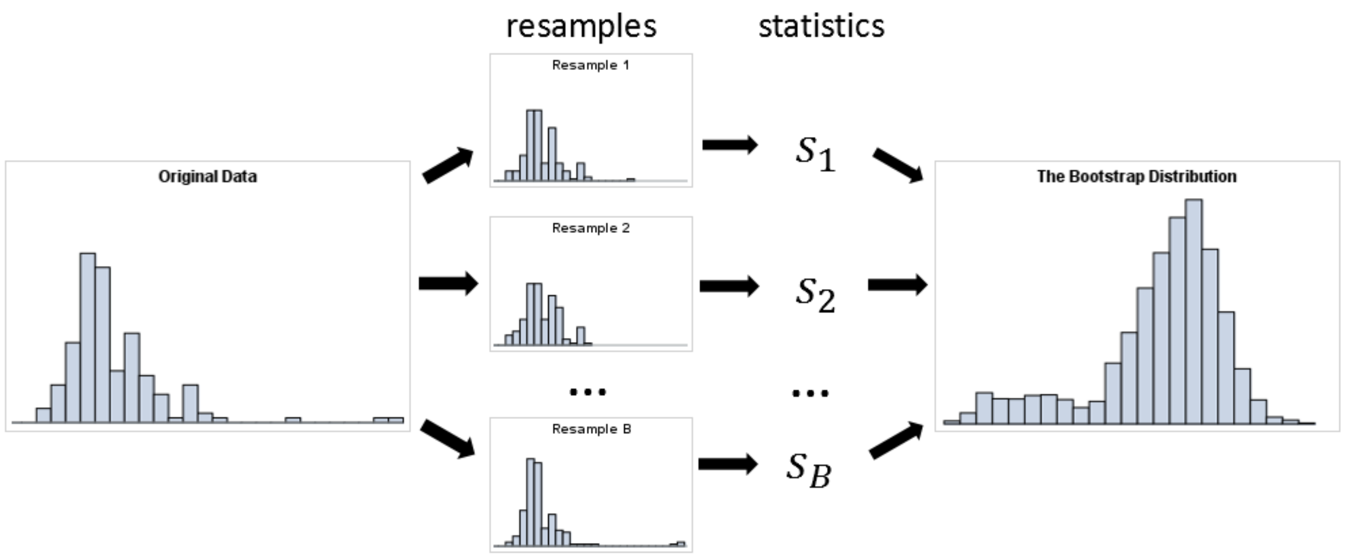
- The bootstrap is a flexible and powerful statistical tool that can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
  - it can provide an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.
- The bootstrap approach allows us to use a computer to mimic the process of obtaining new data sets, so that we can estimate the variability of our estimate without generating additional samples.
- Rather than repeatedly obtaining independent data sets from the population, we instead obtain distinct data sets by repeatedly sampling observations from the original data set with replacement.
- Each of these "bootstrap data sets" is created by sampling with replacement, and is the same size as our original dataset. As a result some observations may appear more than once in a given bootstrap data set and some not at all.



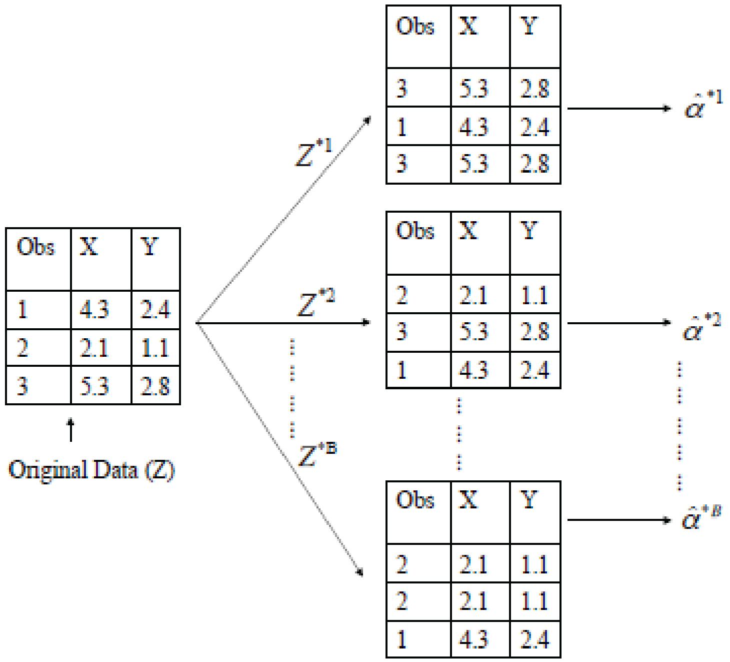
BootStrap


Real World
Random Sampling
Data
Population P
$Z = (z_1, z_2, \dots z_n)$
Estimate $f(Z)$


Bootstrap World
Random Sampling
Bootstrap dataset
Estimated Population $\hat{P}$
$Z^* = (z_1^*, z_2^*, \dots z_n^*)$
Bootstrap Estimate $f(Z^*)$


resamples
statistics
Original Data
Resample 1
Resample 2
Resample B
s_1
s_2
s_B
The Bootstrap Distribution

Example



Obs	X	Y
1	4.3	2.4
2	2.1	1.1
3	5.3	2.8

Obs	X	Y
3	5.3	2.8
1	4.3	2.4
3	5.3	2.8

Obs	X	Y
2	2.1	1.1
3	5.3	2.8
1	4.3	2.4

Obs	X	Y
2	2.1	1.1
2	2.1	1.1
1	4.3	2.4

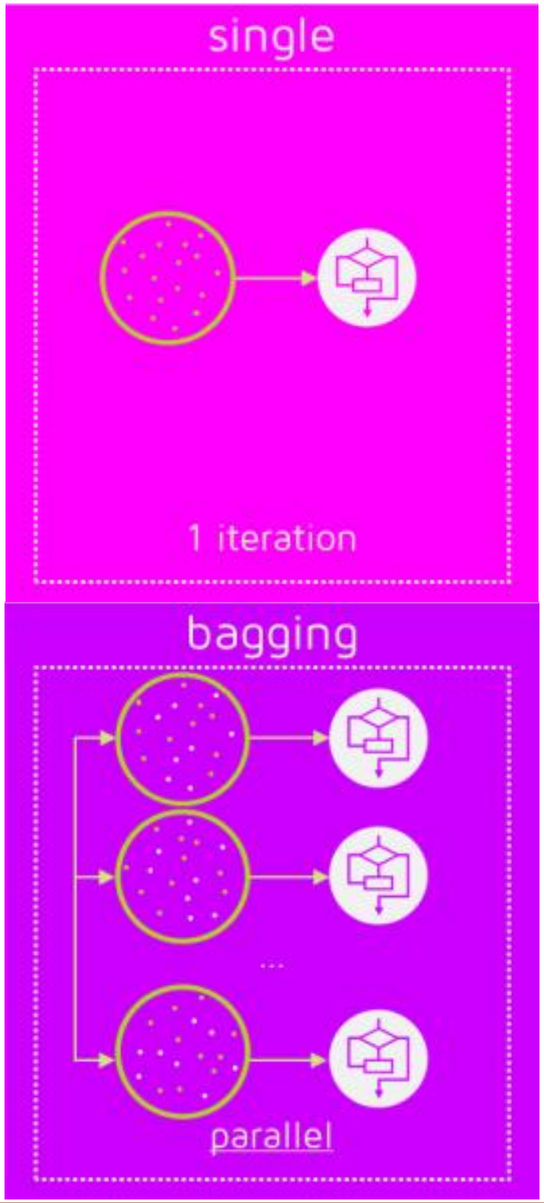
A graphical illustration of the bootstrap approach on a small sample containing n = 3 observations. Each bootstrap data set contains n observations, sampled with replacement from the original data set. Each bootstrap data set is used to obtain an estimate of the parameter of interest.

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.

- 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; *Averaging a set of observations reduces variance*.

- Often time, his is not practical because we generally do not have access to multiple training sets.



Bagging

- Instead, we can bootstrap, by taking repeated samples from the (single) training data set.
- Regression trees: 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

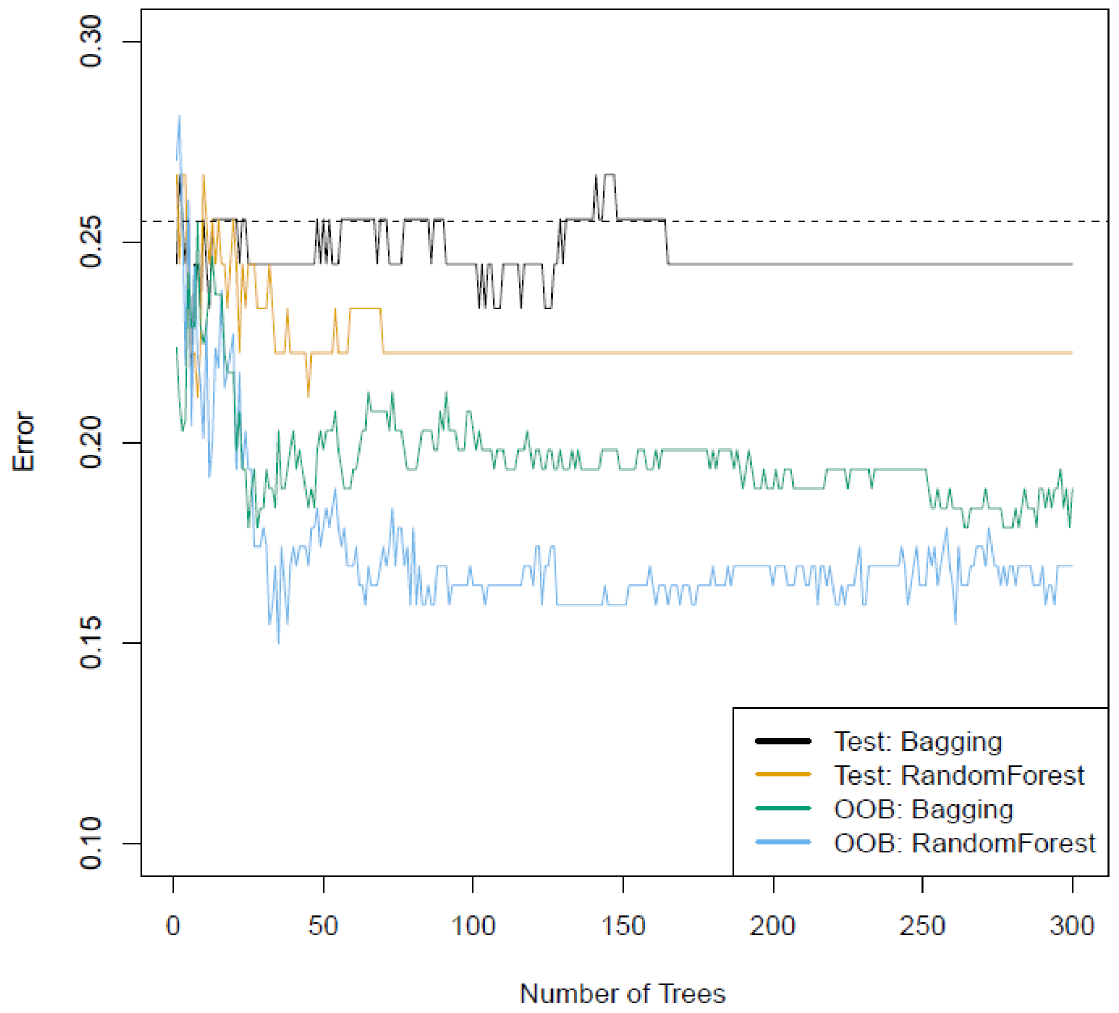
$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

- 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.

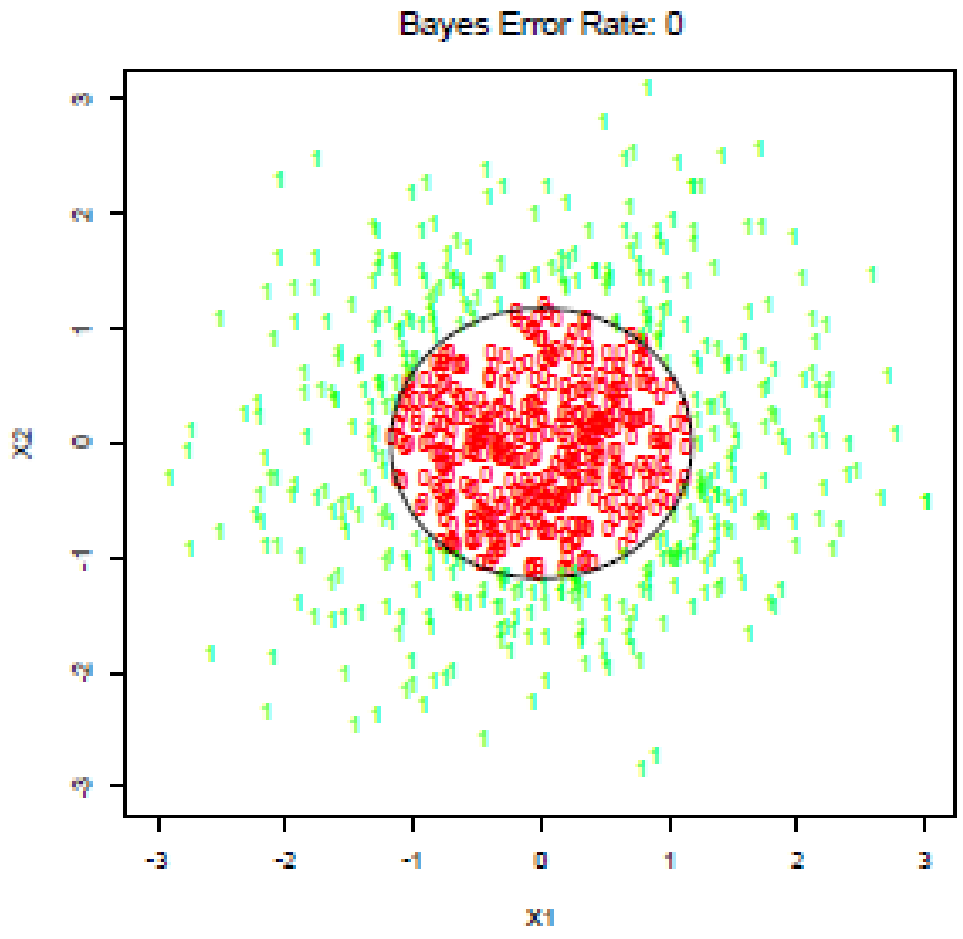
Out-of-Bag Error Estimation

- 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.

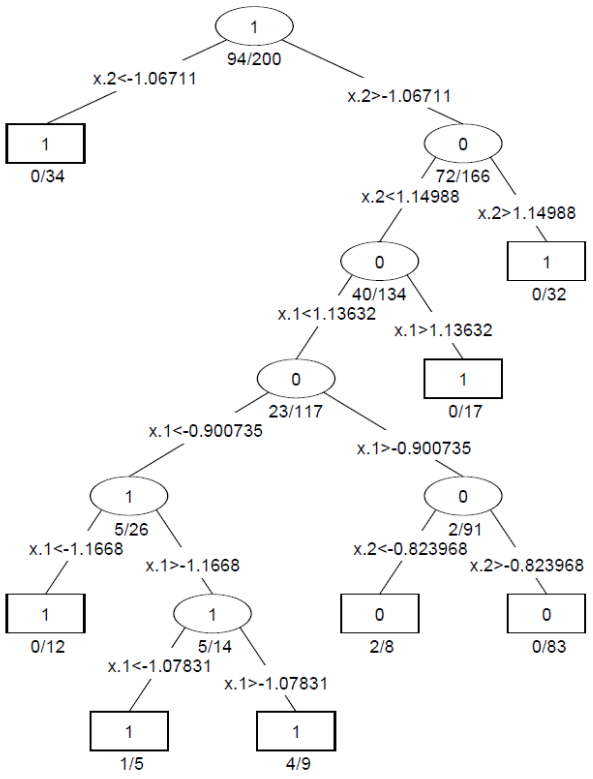
Bagging the heart data

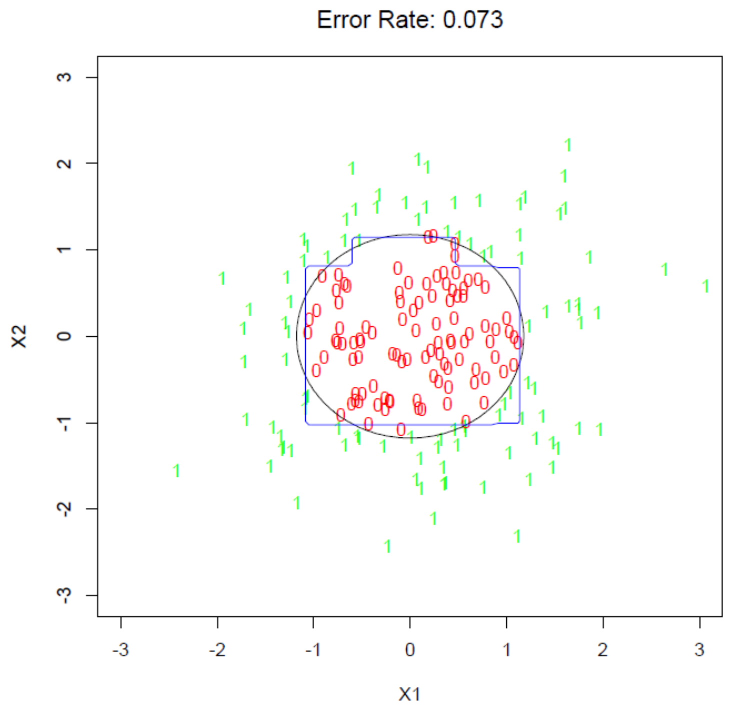


Example

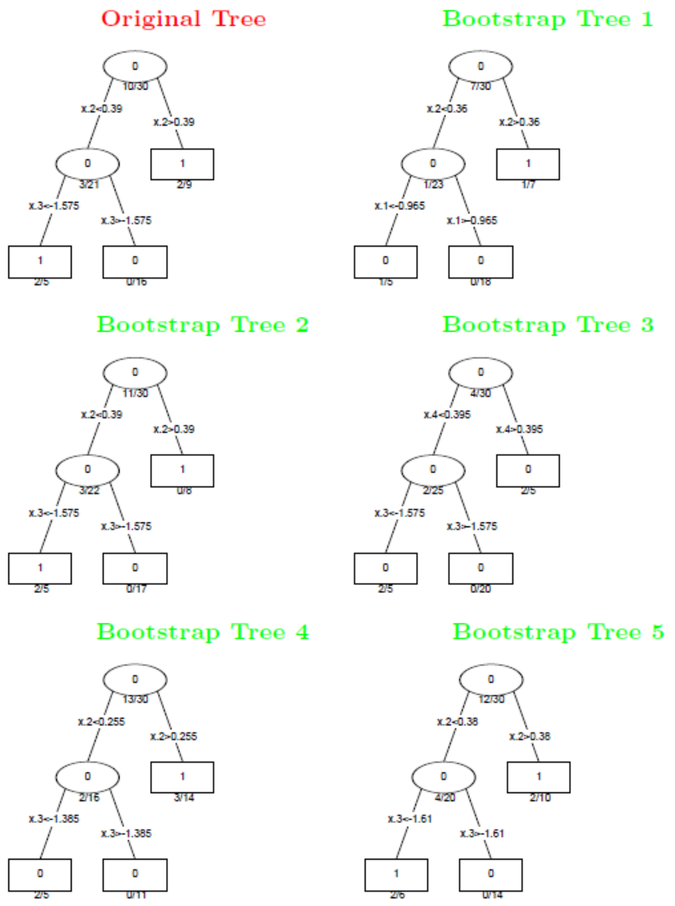


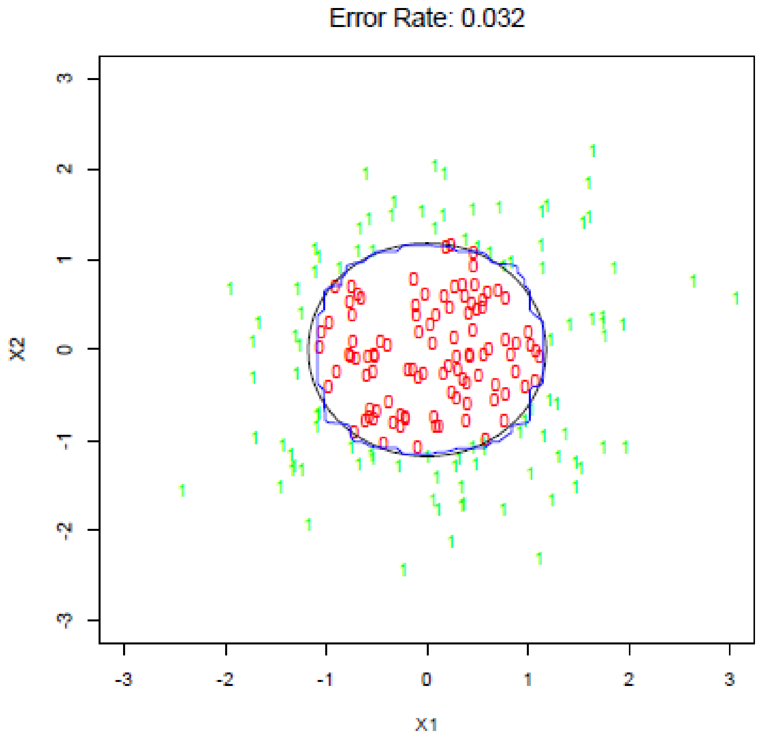
DT


1
94/200
x.2<-1.06711
x.2>-1.06711
1
0/34
0
72/166
x.2<1.14988
x.2>1.14988
0
40/134
1
0/32
x.1<1.13632
x.1>1.13632
0
23/117
1
0/17
x.1<-0.900735
x.1>-0.900735
1
5/26
0
2/91
x.1<-1.1668
x.1>-1.1668
x.2<-0.823968
x.2>-0.823968
1
0/12
1
5/14
0
2/8
0
0/83
x.1<-1.07831
x.1>-1.07831
1
1/5
1
4/9

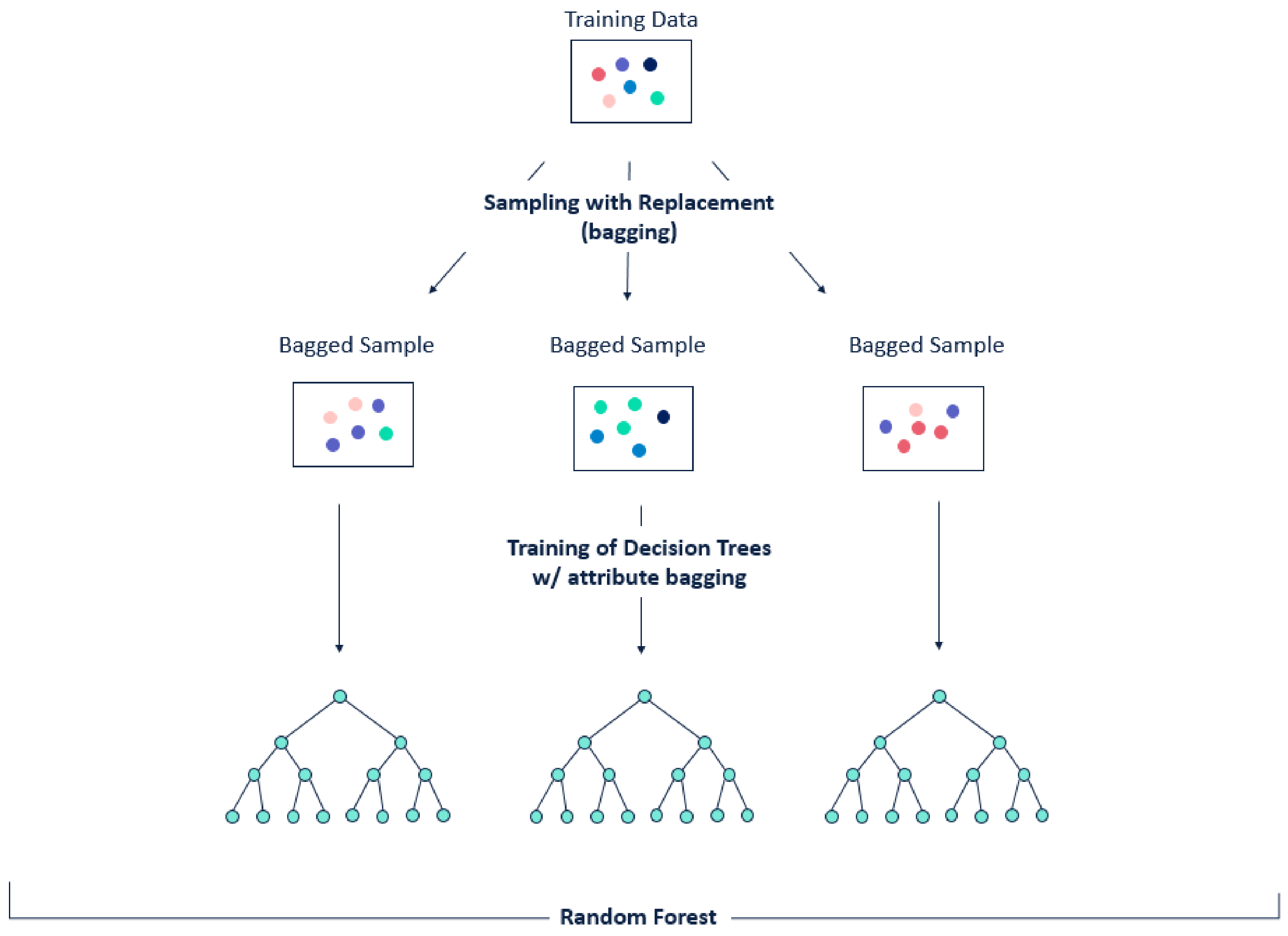

Error Rate: 0.073
X2
X1
-3
-2
-1
0
1
2
3

Bagging


Original Tree
Bootstrap Tree 1
Bootstrap Tree 2
Bootstrap Tree 3
Bootstrap Tree 4
Bootstrap Tree 5


Error Rate: 0.032
X1
X2

Random Forest

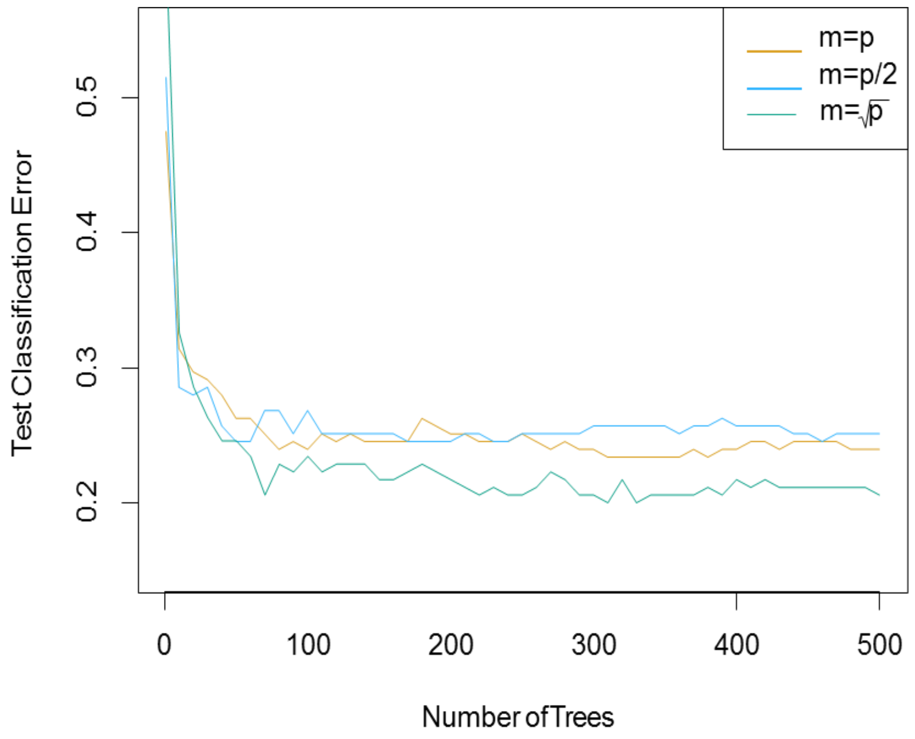

Training Data
Sampling with Replacement (bagging)
Bagged Sample
Bagged Sample
Bagged Sample
Training of Decision Trees w/ attribute bagging
Random Forest

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).

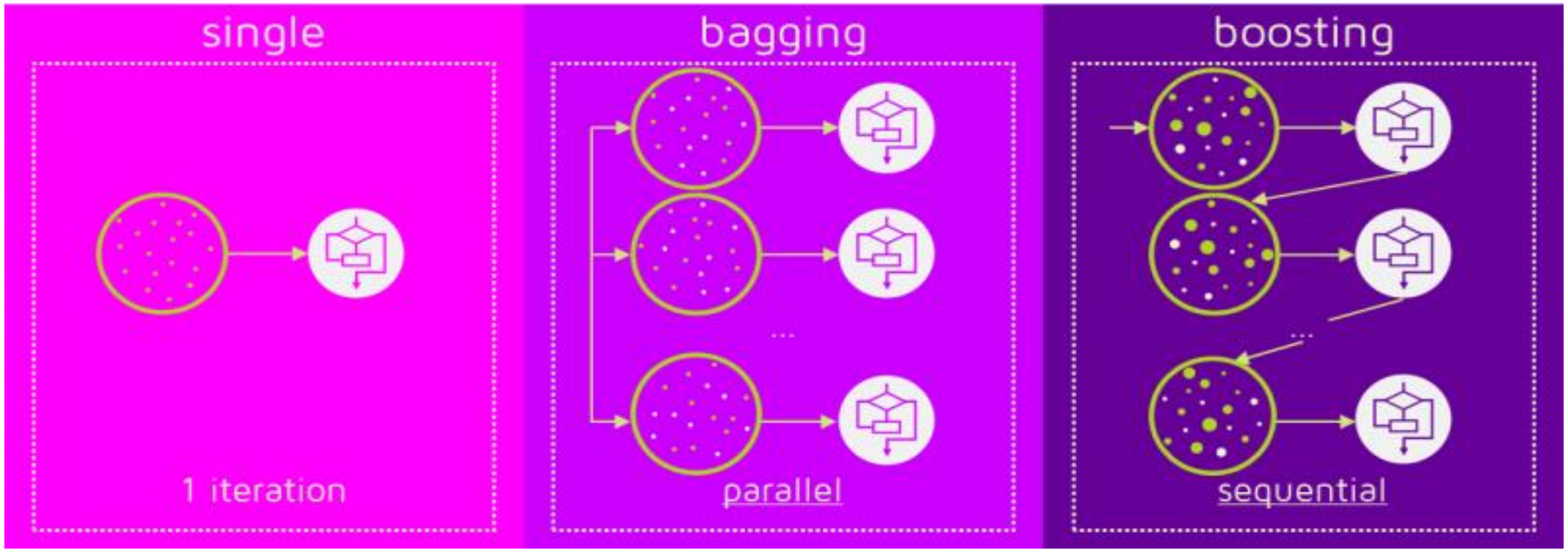
Example: gene expression data

- 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*.



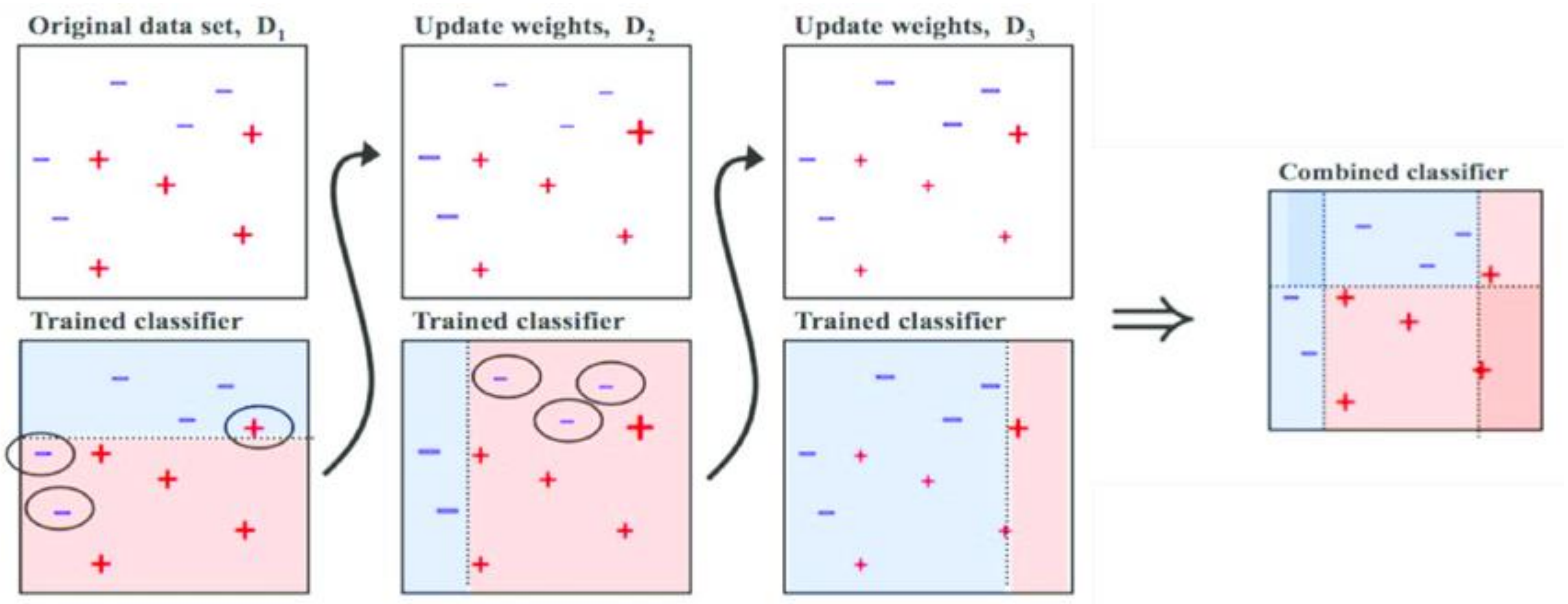
Boosting

- Like bagging, boosting is a general approach that can be applied to many statistical learning methods for regression or classification.
- Boosting works in a similar way, except that the trees are grown *sequentially*: each tree is grown using information from previously grown trees.
- The basic principle behind the working of the boosting algorithm is to generate multiple weak learners and combine their predictions to form one strong rule.
- These weak rules are generated by applying base Machine Learning algorithms on different distributions of the data set.
- These algorithms generate weak rules for each iteration. After multiple iterations, the weak learners are combined to form a strong learner that will predict a more accurate outcome.



How boosting work

- **Step 1:** The base algorithm reads the data and assigns equal weight to each sample observation.
- **Step 2:** False predictions made by the base learner are identified. In the next iteration, these false predictions are assigned to the next base learner with a higher weightage on these incorrect predictions.
- **Step 3:** Repeat step 2 until the algorithm can correctly classify the output.



Boosting algorithm 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:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$
   c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$
3. Output the boosted model,
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

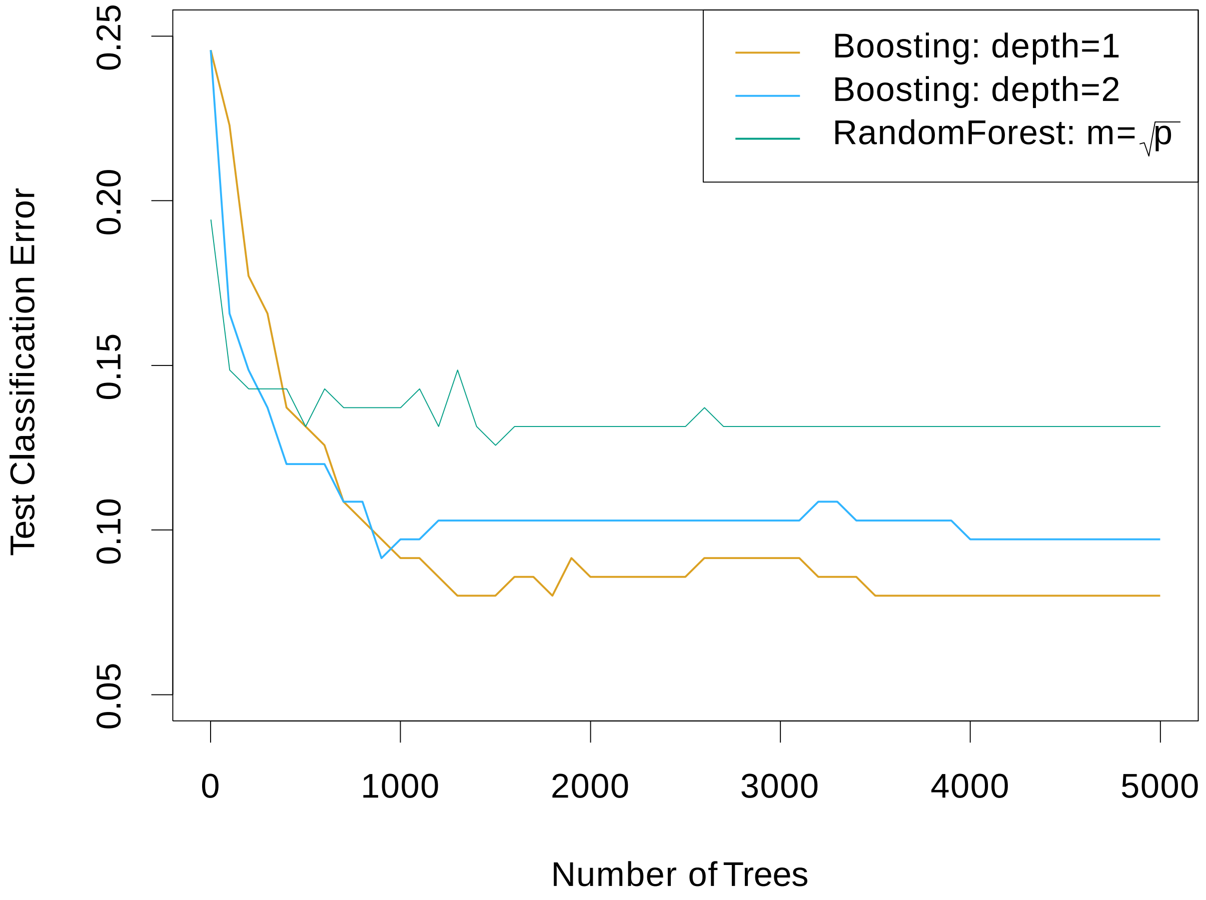
What is the idea behind this procedure?

- 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.

Tuning parameters for boosting

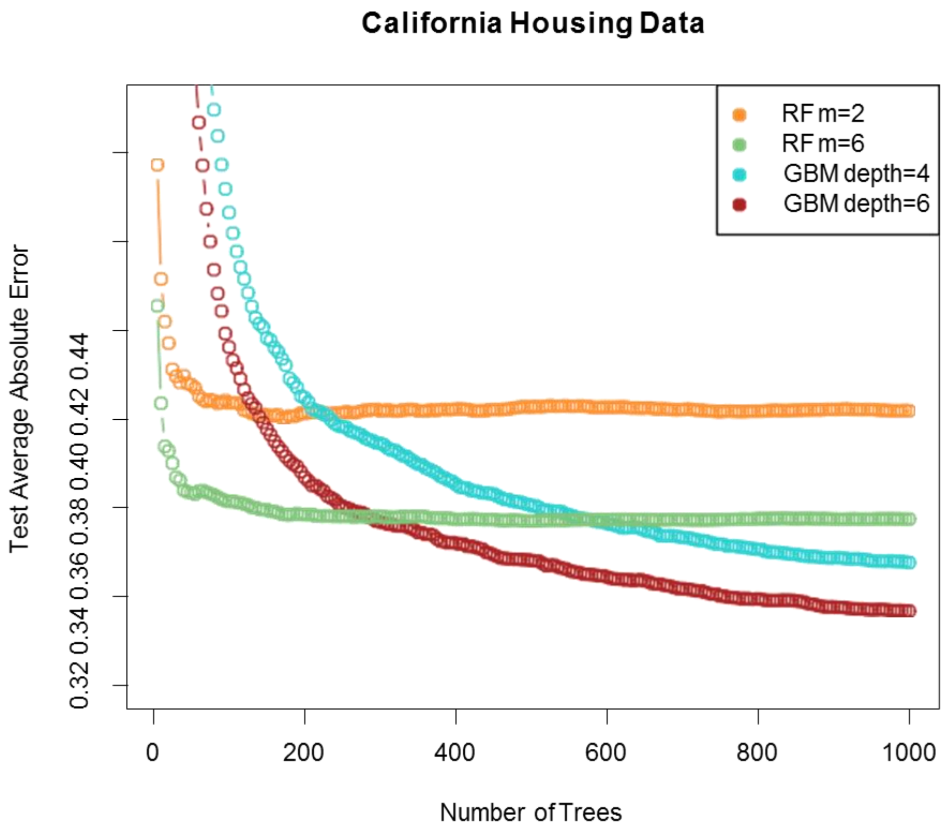
1. The *number of trees* B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
2. The *shrinkage parameter* λ, a small positive number. This controls the rate at which boosting learns. 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.

Gene expression data continued



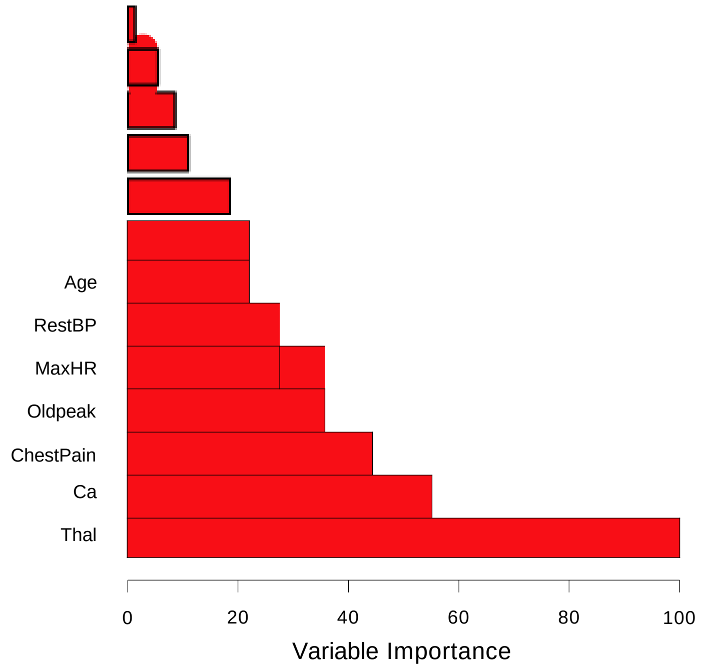
Details of previous figure

- 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%.



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 plot for the Heart data

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.