Bagging and Random Forests

Auburn University

PMII_Fall 2019

Dr. Pei Xu

Content list

- Ensemble Models
  - Bagging
  - Random Forest
  - Boosting

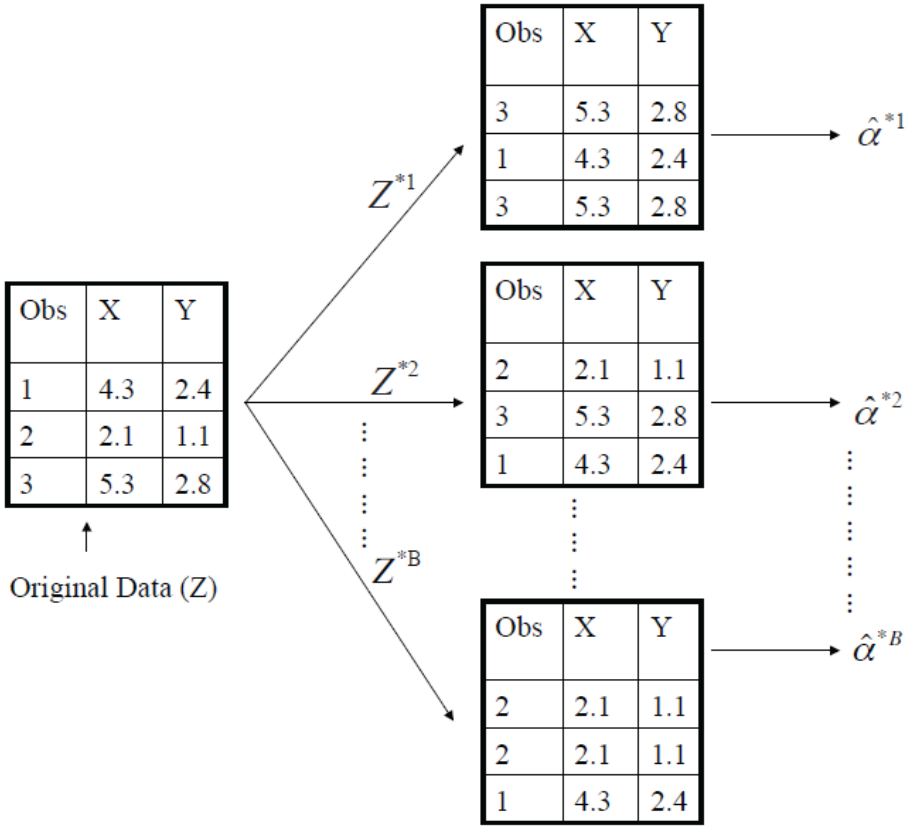
Part I. Bagging

Problem!

- Decision trees discussed earlier suffer from 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
- We would like to have models with low variance
- To solve this problem, we can use bagging (***b***ootstrap ***agg***regat***ing***).

Bootstrapping is simple!

- Resampling of the observed dataset (and of equal size to the observed dataset), each of which is obtained by random sampling with replacement from the original dataset.



What is bagging?

- Bagging is an extremely powerful idea based on two things:
  - Averaging: reduces variance!
  - Bootstrapping: plenty of training datasets!
- Why does averaging reduces variance?
  - Averaging a set of observations reduces variance. 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

How does bagging work?

- Generate B different bootstrapped training datasets
- Train the statistical learning method on each of the B training datasets, and obtain the prediction
- For prediction:
  - Regression: average all predictions from all B trees
  - Classification: majority vote among all B trees

Bagging for Regression Trees

- Construct B regression trees using B bootstrapped training datasets
- Average the resulting predictions

- Note: These trees are not pruned, so each individual tree has high variance but low bias. Averaging these trees reduces variance, and thus we end up lowering both variance and bias ☺

Bagging for Classification Trees

- Construct B regression trees using B bootstrapped training datasets
- For prediction, there are two approaches:
  1. Record the class that each bootstrapped data set predicts and provide an overall prediction to the most commonly occurring one (majority vote).
  2. If our classifier produces probability estimates we can just average the probabilities and then predict to the class with the highest probability.
- Both methods work well.

Review Question

Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of X, produce 10 estimates of P(Class is Red|X):

0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, and 0.75.

There are two common ways to combine these results together into a single class prediction.

- One is the majority vote approach discussed in this chapter.
- The second approach is to classify based on the average probability.

In this example, what is the final classification under each of these two approaches?

Review Question

Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of X, produce 10 estimates of P(Class is Red|X):

0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, and 0.75.

There are two common ways to combine these results together into a single class prediction.

- One is the majority vote approach discussed in this chapter.
- The second approach is to classify based on the average probability.

In this example, what is the final classification under each of these two approaches?

Solution: With the majority vote approach, we classify XX as Red as it is the most commonly occurring class among the 10 predictions (6 for Red vs 4 for Green).

With the average probability approach, we classify XX as Green as the average of the 10 probabilities is 0.45.

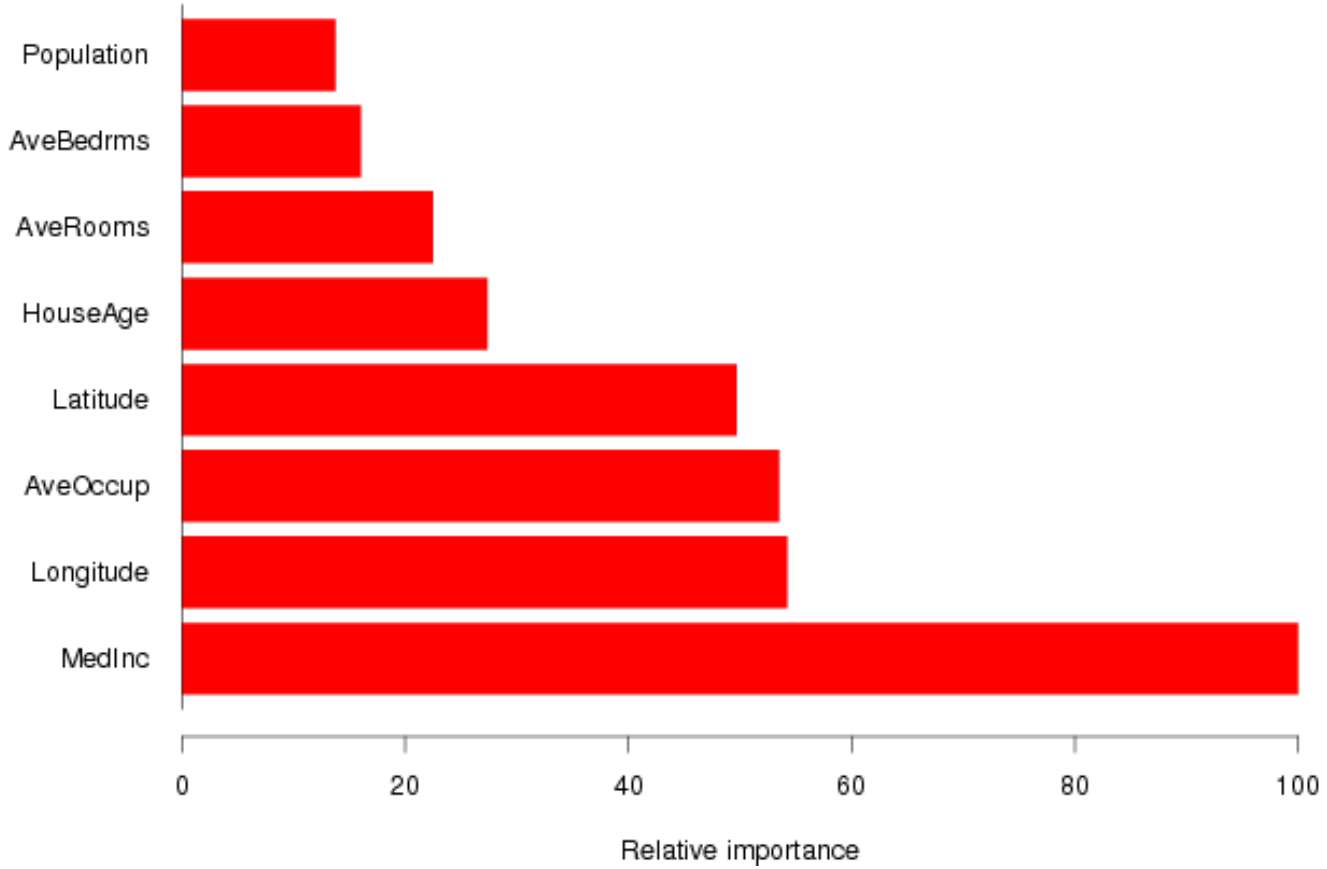
Variable Importance Measure

- Bagging typically improves the accuracy over prediction using a single tree, but it is now hard to interpret the model!
- We have hundreds of trees, and it is no longer clear which variables are most important to the procedure
- Thus bagging improves prediction accuracy at the expense of interpretability
- But, we can still get an overall summary of the importance of each predictor using Relative Influence Plots

Relative Influence Plots

- How do we decide which variables are most useful in predicting the response?
  - We can compute something called relative influence plots.
  - These plots give a score for each variable.
  - These scores represents the decrease in MSE when splitting on a particular variable
  - A number close to zero indicates the variable is not important and could be dropped.
  - The larger the score the more influence the variable has.

Example: Housing Data



- Median Income is by far the most important variable.
- Longitude, Latitude and Average occupancy are the next most important.

Part II. Random Forests

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?
  - Build a number of decision trees on bootstrapped training sample, 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 (Usually $m \approx \sqrt{p}$)

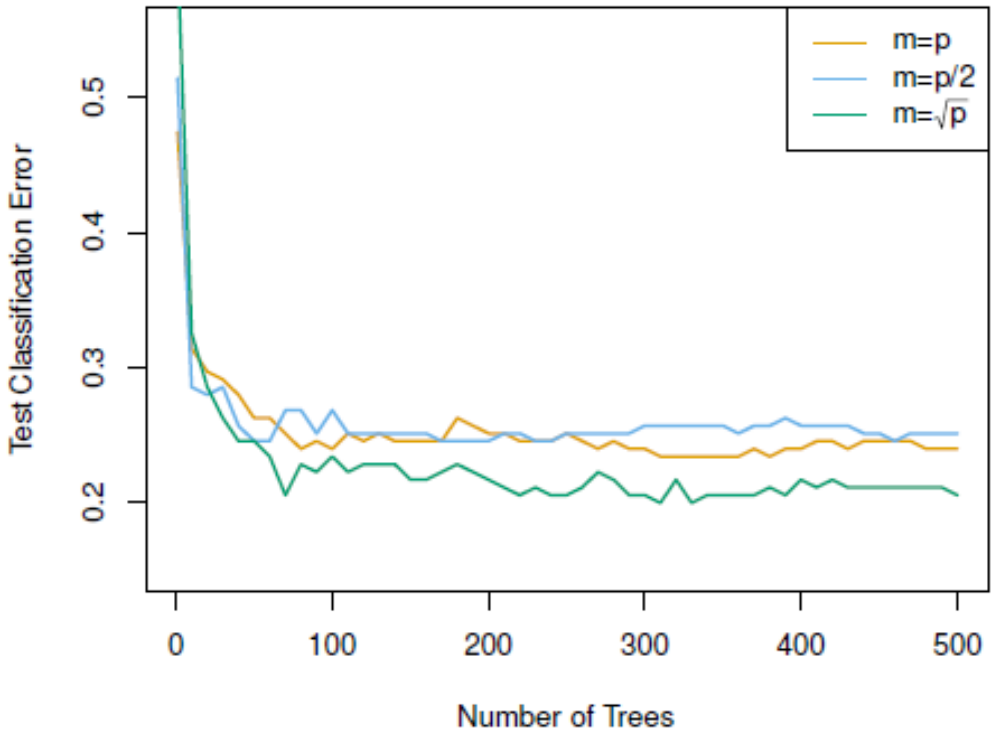
Random Forests

Why are we considering a random sample of m predictors instead of all p predictors for splitting?

- Suppose that we have a very strong predictor in the data set along with a number of other moderately strong predictor, then in the collection of bagged trees, most or all of them will use the very strong predictor for the first split!
- 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 a large variance reduction, and thus random forests "de-correlates" the bagged trees leading to more reduction in variance

Random Forest with different values of “m”

- Notice when random forests are built using m = p, then this amounts simply to bagging.



Example: Gene Expression Data

Sample code for Random Forest

```
from sklearn.ensemble import RandomForestClassifier

rfc1 = RandomForestClassifier(max_features=3, random_state=1)
rfc1.fit(X_train, y_train)
pred1 = rfc1.predict(X_test)
print(roc_auc_score(y_test, pred1))
```

Part III. Boosting

Boosting

- 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 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:
   - 2.1 Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r).
   - 2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:

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

   - 2.3 Update the residuals,

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

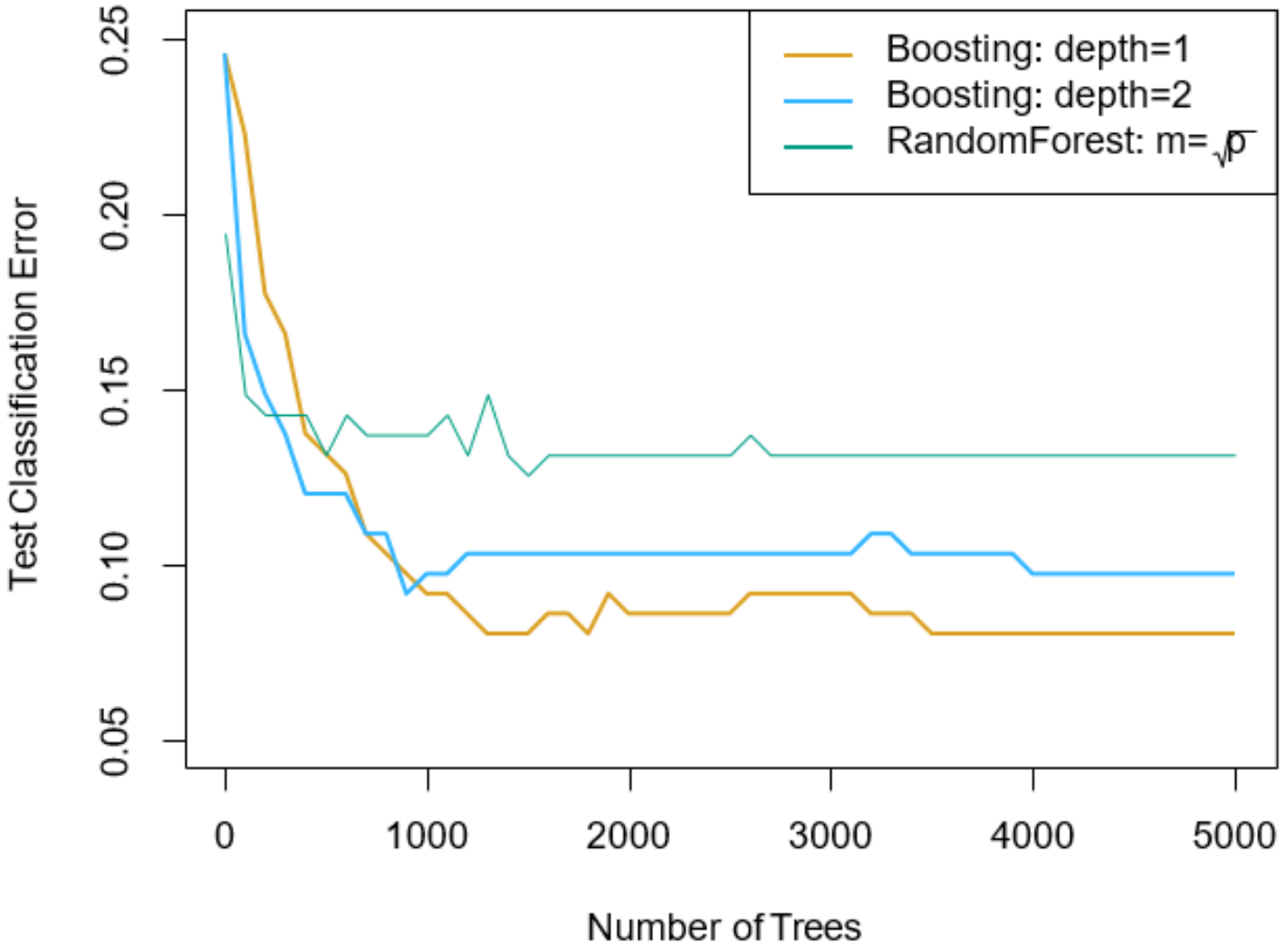
3. Output the boosted model,

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

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.

Gene expression data



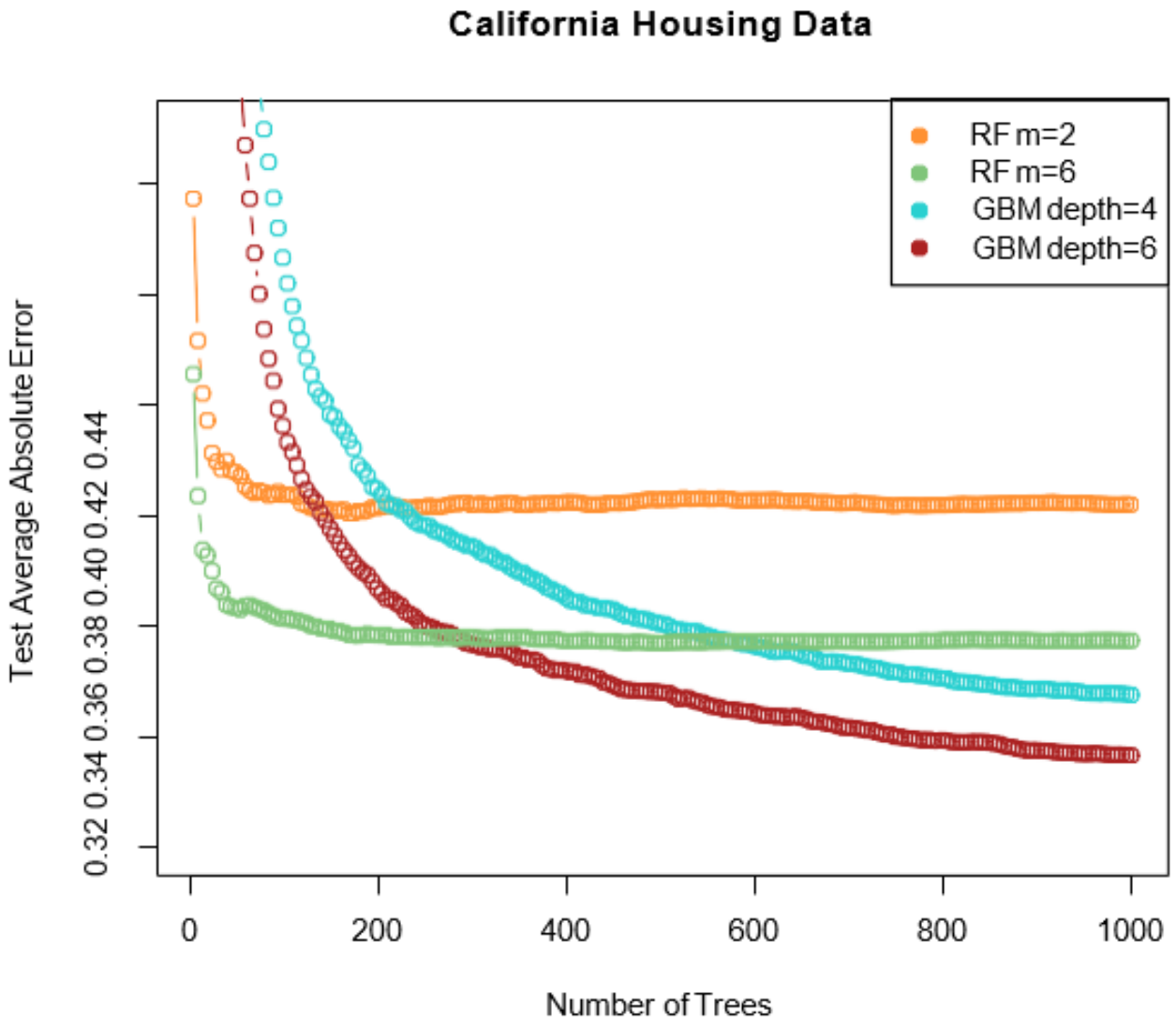
Details of previous figure

- 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

- 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, 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.
- 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 consists of a single split. 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.

Another regression example



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.

Reference

- Textbook: Introduction to Statistic Learning (Chapter 8)
- Tan, Pang-Ning et al. Introduction to Data Mining. 2018 (Section 4.10)