Bagging and Random Forests

Rebecca C. Steorts

Department of Statistical Science
Duke University

Predictive Modeling

November 2015

While bagging can improve predictions for many regression methods, it is very useful for regression trees. To apply bagging to regression trees we:

1. Construct B regression trees using B bootstrapped training sets.
2. We then average the predictions.
3. These trees are grown deep and are not pruned.
4. Each tree has a high variance with low bias. Averaging the B trees brings down the variance.
5. Bagging has been shown to give impressive improvements by combining hundreds or thousands of trees in a single procedures.

- We have just described bagging in a regression context to predict a quantitative outcome Y.
- What if our outcome is qualitative?
- For a given test observation, we can record the class predicted by each of the B trees.
- Now take a majority vote: the overall prediction is the most commonly occurring class among the B predictions.

Straightforward way to estimate the test error of a bagged model, without the need to perform CV or the validation test set approach.

- Key to bagging is that the trees are repeatedly fit to bootstrapped subsets of observations.
- Can show that on average, each bagged tree make use of around two-third of the observations.
  - The remaining one-third of the observations not used to fit a given bagged tree are called *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 approximately $B/3$ predictions for the ith observation.
- To obtain a single prediction for the ith observation, we can average these predicted responses (if regression is the goal).
- This yields a single OOB prediction for the ith observation.

- An OOB prediction can be found in this way for each of the n observations, from which the overall OOB MSE (for a regression problem) or a classification error (for a classification problem) can be computed.
- Check the details on your own!

- Bagging typically results in improved accuracy over prediction using a single tree.
- However, the resulting model can be difficult to interpret.
- One advantage of decision trees is ease of interpretation.
- When we bag a large number of trees, we can no longer represent the resulting statistical learning procedure using a single tree
- It is not clear which variables are the most important.
- Bagging improves prediction accuracy at the cost of interpretability.

- Collection of bagged trees is much more difficult to interpret that a single tree.
- But we can obtain a summary of the importance of each predictor us the RSS (for bagging regression trees) or the Gini index (for bagging classification trees).
- In the case of regression trees, we can record the total amount that the RSS decreases due to splits over a given predictor, averaged over all B trees.
  - A large value indicates an important predictor.
  - For classification trees, we can add up the total amount that the Gini index decreases by splits over a given predictor averaged over all B trees.

Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees. As in bagging, we build a number of decision trees on bootstrapped training samples.

- When building these decision trees, each time a split is considered, a random sample 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 sample of m predictors is taken at each split, and typically we take $m \approx \sqrt{(p)}$.
- That is, the number of predictors considered at each split is approximately equal to, the square root of the total number of predictors.

- Build a random forest
  - At each split:
  - The algorithm is not allowed to consider a majority of the available predictors.
  - What is the reasoning?
  - Suppose one very strong predictor along with many moderately strong predictors.
- In the collection of bagged trees, most or all of the trees will use this strong predictor in the top split.
- All of the bagged trees will look very similar.
- Predictions from the bagged trees highly correlated.
- Averaging highly correlated trees doesn't lead to as large of a reduction in variance as averaging many uncorrelated trees.
- This means that bagging will NOT lead to a substantial reduction in variance over a single tree for this example.

- How do random forests overcome this problem?
- Force each split to consider only a subset of the predictors.
- On average, $(p - m)/p$ of the splits won't even consider a strong predictor
- Means other predictors have more of a chance!
- This process de-correlates the trees.

- What is the main difference between bagging and random forests?
- It's the choice of the predictor subset size m.
- For example, if the random forest is built using $m = p$, then this is the same as bagging.
- Using a small value of m in building a random forest will typically be helpful when we have a large number of correlated predictors.

- We perform bagging on the Boston dataset using the randomForest package in R.
- The results from this example will depend on the version of R installed on your computer.[1]
- We can use the randomforest() function to perform both random forests and bagging.
- Note that `medv` is the median value of owner-occupied homes in $1000s.
- The covariates can be found at `http://cran.r-project.org/web/packages/MASS/MASS.pdf`

[1] Recall that bagging is a special case of a random forest with $m = p$.

```
library(ISLR)
library(MASS)
library(randomForest)
attach(Boston)
set.seed(1)
train <- sample(1:nrow(Boston), nrow(Boston)/2)
bag.boston <- randomForest(medv~., data=Boston,
     subset=train,  mtry=13, importance= TRUE)
```

The argument mtry=13 indicates that all 13 predictors should be considered for each split of the tree. Basically, bagging should be done. How well does the bagged model perform on the test set?

```
> bag.boston

Call:
 randomForest(formula = medv ~ ., data = Boston, mtry = 13,
 importantance = TRUE,       subset = train)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 13

          Mean of squared residuals: 11.13692
                    % Var explained: 86.52
```

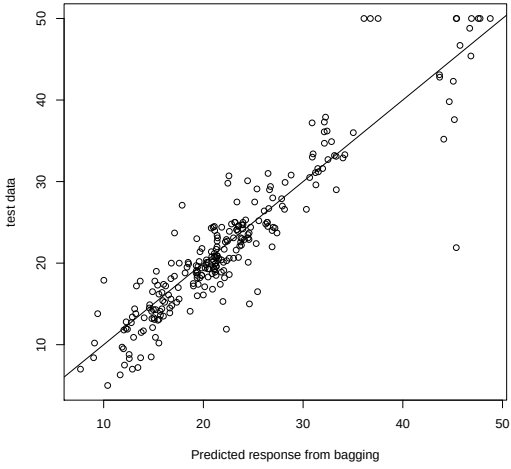

Figure 1

The code below shows that using tree=25 increases the test set MSE to from 11.14 to 14.28.

```
bag.boston <- randomForest(medv~., data=Boston,
     subset=train, mtry=13, importantance = TRUE, ntree = 25)
yhat.bag <- predict(bag.boston, newdata=Boston[-train,])
mean((yhat.bag - boston.test)^2)
```

- Growing a random forest proceeds in exactly the same way, except we use a smaller value of the mtry argument.
- By default, randomForest() uses $p/3$ variables when building a random forest of regression trees, and $\sqrt{(p)}$ variables when building a random forest of classification trees. Here we use a mtry=6.
- The test set MSE is 11.63 (compared to 14.28), indicating that random forests yield an improvement over bagging.

```
set.seed(1)
rf.boston <- randomForest(medv~., data=Boston, subset=train,
mtry=6, importantance = TRUE)
yhat.rf <- predict(rf.boston, newdata=Boston[-train,])
mean((yhat.rf - boston.test)^2)
```

Using the importance() function, we can view the importance of each variable.

```
> importance(rf.boston)
        IncNodePurity
crim       1127.35130
zn           52.68114
indus      1093.92191
chas         56.01344
nox        1061.66818
rm         6298.06890
age         556.56899
dis        1371.10322
rad         111.89502
tax         442.61144
ptratio     947.18872
black       370.15308
lstat      7019.97824
```

- Two measures of variable importance are reported.
- The former is based upon the mean decrease of accuracy in predictions in the out of bag samples when a given variable is excluded from the model.
- The latter is a measure of the total decrease in node impurity that results from splits over that variables, averaged over all trees.
- In the case of regression trees, the the node impurity is measured by the training RSS and for classification trees by the deviance.
- Plots of these importance measures can be produced using the varImpPlot() function.

Node Purity

- Splitting on those variables (rm and stat) has more of an effect on the conditional distribution of the response (y) compared to the other predictors.
- As the node purity increases, the conditional distribution of the response is more concentrated around particular points. (Recall we are doing regression and not classification).
- In a classification setting, the node purity means that the response (y) is increasingly of the same type.

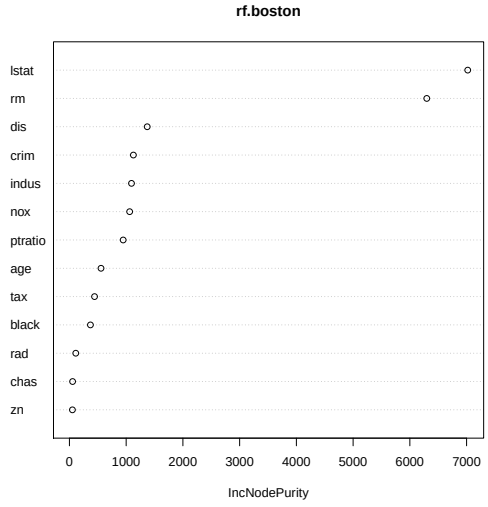
rf.boston
lstat
rm
dis
crim
indus
nox
ptratio
age
tax
black
rad
chas
zn
0
1000
2000
3000
4000
5000
6000
7000
IncNodePurity

Figure 2