Lex Analytics

Tool: Bagging and Random Forest

The Analytics Edge: Bagging and random forests use trees as building blocks to construct more powerful prediction models. Bagging is an ensemble algorithm that fits multiple models on different subsets (obtained by sampling with replacement) of a training dataset, then combines the predictions from all models. Random forest is an extension of bagging that also randomly selects subsets of features used in each data sample. Both bagging and random forests have proven effective on a wide range of different predictive modeling problems.

1 Bootstrapping

The Bootstrap is a resampling method used to quantify the uncertainty associated with a statistical learning method (or a given estimator). The term originates from the expression "to pull oneself up by one's bootstraps". The other resampling method introduced in this course is cross-validation.

1.1 A toy example

We illustrate the bootstrap on a toy example in which we wish to determine the best investment allocation under a simple model. Suppose that we want to invest a given amount of money in two financial assets that yield returns of X and Y, respectively, where X and Y are random quantities. We will invest a fraction α of our money in X, and will invest the remaining $1-\alpha$ in Y. Since there is variability associated with the returns on these two assets, we wish to choose the value of α that minimizes the total risk, measured with the variance of our investment, $\text{Var}(\alpha X + (1-\alpha)Y)$. One can show that the value of α that minimizes the risk is given by

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

where $\sigma_X^2 = \text{Var}(X)$, $\sigma_Y^2 = \text{Var}(Y)$, $\sigma_{XY} = \text{Cov}(X, Y)$.

In reality, the values of σ_X^2, σ_Y^2, and σ_{XY} are unknown. We can calculate estimates ($\hat{\sigma}_X^2$, $\hat{\sigma}_Y^2$, and $\hat{\sigma}_{XY}$) using a dataset with observations of X and Y. And then estimate $\hat{\alpha}$ by

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}. \quad (1)$$

The Figure 1 illustrates this approach for estimating α on a simulated data set. In each panel, we simulated 100 pairs of returns for the investments X and Y. We used these returns to estimate σ_X^2, σ_Y^2, and σ_{XY}, which we then substituted into (1) in order to obtain estimates for α. The value of $\hat{\alpha}$ resulting from each simulated data set ranges from 0.532 to 0.657.

It is natural to wish to quantify the accuracy of our estimate of α. To estimate the standard deviation of $\hat{\alpha}$, we repeated the process of simulating 100 paired observations of X and Y, and estimating α using (1), 1000 times. We thereby obtained 1000 estimates for α, which we can call $\hat{\alpha}_1, \hat{\alpha}_2, ..., \hat{\alpha}_{1000}$. The left-hand panel of Figure 2 displays a histogram of the resulting estimates. For these simulations the parameters were set to $\sigma_X^2 = 1$, $\sigma_Y^2 = 1.25$, and $\sigma_{XY} = 0.50$, so we know that the true value of α is 0.60. The mean over all 1000 estimates for α is

$$\bar{\alpha} = \frac{1}{1000}\sum_{r=1}^{1000}\hat{\alpha}_r = 0.5996,$$

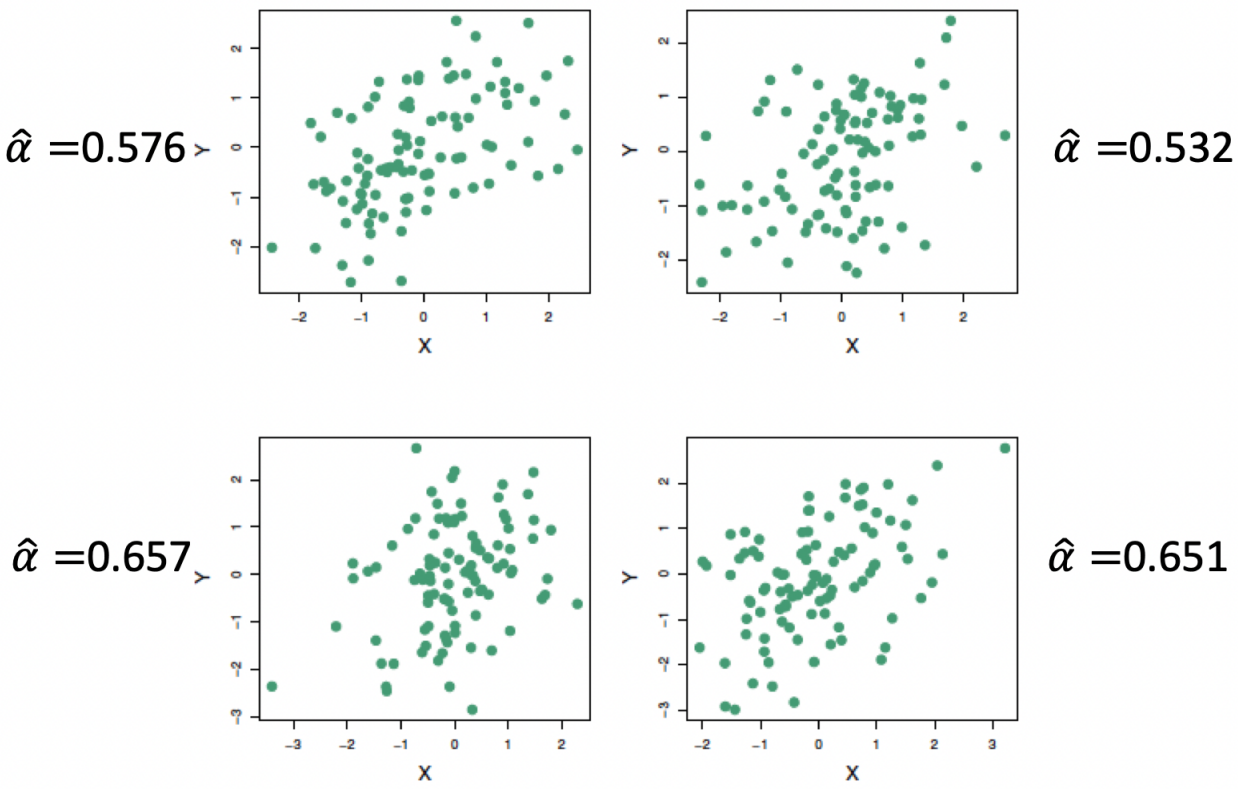

Figure 1: Each panel displays 100 simulated returns for investments X and Y. (Source: James et al., 2014).

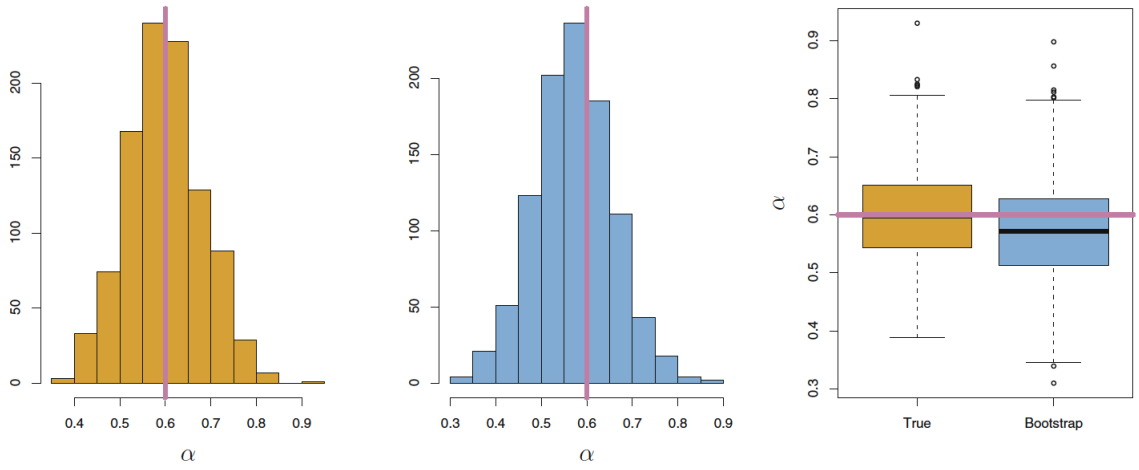

Figure 2: Left: A histogram of the estimates of α obtained by generating 1000 simulated data sets from the true population. Center: A histogram of the estimates of α obtained from 1000 bootstrap samples from a single data set. Right: The estimates of α displayed in the left and center panels are shown as boxplots. (Source: James et al., 2014).

which is close to 0.60. The standard deviation of the estimates is:

$$\sqrt{\frac{1}{1000-1}\sum_{r=1}^{1000}(\hat{\alpha}_r - \bar{\alpha})^2} = 0.083.$$

This gives us a very good idea of the accuracy of $\hat{\alpha}$: $\text{SE}(\hat{\alpha}) \approx 0.083$. So roughly speaking, for a random sample from the population, we would expect $\hat{\alpha}$ to differ from α by approximately 0.08, on average.

1.2 Bootstrapping

In practice, however, the example above cannot be used, because we cannot generate new samples from the original population. Bootstrapping mimics this process: we obtain distinct datasets by repeatedly sampling observations from the original data set with replacement. Each of these bootstrap datasets is created by sampling with replacement, and is the same size as our original dataset.

This approach is illustrated in Figure 3 on a simple data set, which we call Z, that contains only $n = 3$ observations.

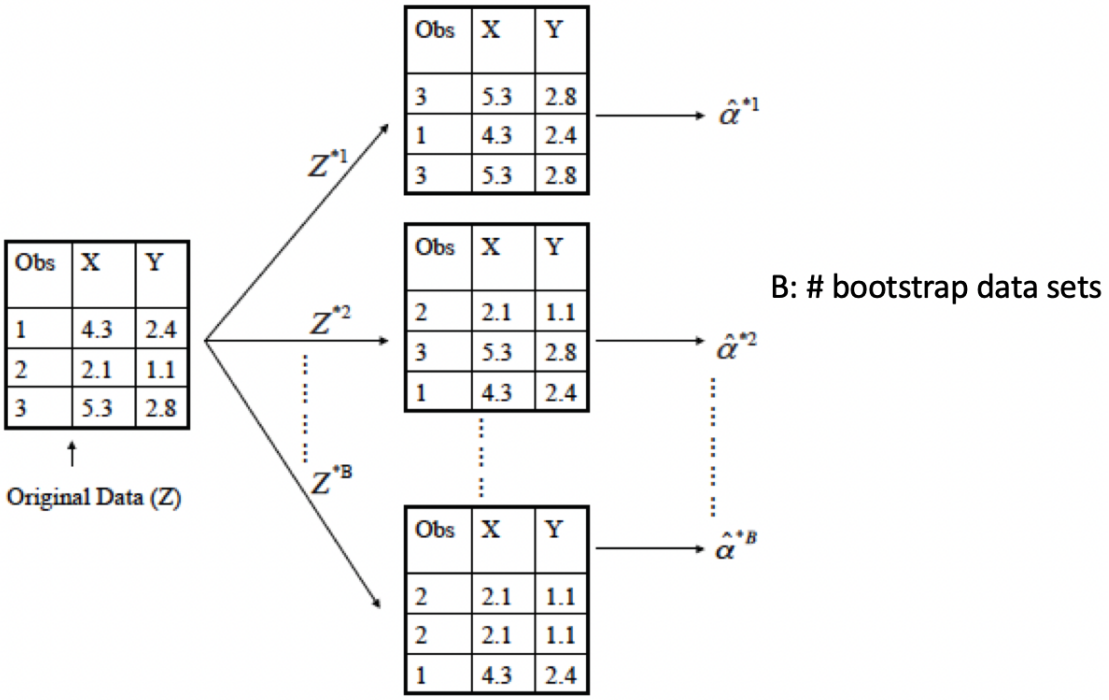

Figure 3: A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations. (Source: James et al., 2014).

We randomly select n observations from the data set in order to produce a bootstrap data set, Z^{*1}. The sampling is performed with replacement, which means that the same observation can occur more than once in the bootstrap data set. In this example, Z^{*1} contains the third observation twice, the first observation once, and no instances of the second observation. Note that if an observation is contained in Z^{*1}, then both its X and Y values are included. We can use Z^{*1} to produce a new bootstrap estimate for α which we

call $\hat{\alpha}^{*1}$. The procedure is repeated B times (for example, 100 or 1000), in order to produce B different bootstrap datasets (Z^{*1}, Z^{*2}, ..., Z^{*B}), and B corresponding estimates $\hat{\alpha}^{*1}$, $\hat{\alpha}^{*2}$, ..., $\hat{\alpha}^{*B}$. We estimate the standard error of these bootstrap estimates with the formula

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

where $\bar{\hat{\alpha}}^{*} = \frac{1}{B}\sum_{r=1}^{B}\hat{\alpha}^{*r}$. This serves as an estimate of the standard error of $\hat{\alpha}$ estimated from the original data set.

The bootstrap approach is illustrated in the center panel of Figure 2, which displays a histogram of 1000 bootstrap estimates of α, each computed using a distinct bootstrap data set. This panel was constructed on the basis of a single data set, and hence could be created using real data. Note that the histogram looks very similar to the left-hand panel which displays the idealized histogram of the estimates of α obtained by generating 1000 simulated data sets from the true population. In particular the bootstrap estimate $\mathrm{SE}_B(\hat{\alpha})$ is 0.087, very close to the estimate of 0.083 obtained using 1000 simulated data sets. The right-hand panel displays the information in the center and left panels in a different way, via boxplots of the estimates for α obtained by generating 1000 simulated data sets from the true population and using the bootstrap approach. Again, the boxplots are quite similar to each other, indicating that the bootstrap approach can be used to effectively estimate the variability associated with $\hat{\alpha}$.

1.3 Cross-Validation and the Bootstrap

First, we give some differences between cross-validation and bootstrapping. Bootstrap resamples with replacement, while cross-validation resamples without replacement. In addition, the main goal of cross-validation is to measure, or generalize, the performance of a model,, while bootstrapping is used to establish empirical distribution functions for a widespread range of statistics.

Next, can we use bootstrapping to estimate the prediction error? We could think about using each bootstrap dataset as our training sample, and the original sample as our validation sample. But each bootstrap sample has significant overlap with the original data, thus the bootstrap will underestimate the true prediction error. We can fix the problem by using the out-of-bag (out-of-bootstrap) estimate.

The bootstrap is an extremely powerful idea. It is used in many situations in which it is hard or even impossible to directly compute the standard deviation of a quantity of interest. We see here that the bootstrap can be used in a completely different context, in order to improve statistical learning methods such as decision trees.

2 Bagging

We know that decision trees suffer from high variance. This means that if we split the training data into two parts at random, and fit a decision tree to both halves, the results that we get could be quite different. Bagging (or Bootstrap aggregation), 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.

2.1 Bagging

Recall that given a set of n independent observations $Z^1, Z^2, \ldots, Z^n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of observations is σ^2/n. In other words, averaging a set of observations reduces variance. Hence a natural way to reduce the variance and increase the prediction accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions.

Let's write this (slightly) more formally for regression. We could take repeated samples from the training dataset (if that's everything we have), that is, generate B different bootstrapped training datasets. Then we train model (e.g. CARTs without pruning) $f^{*b}(x)$ on the b-th bootstrapped training dataset (repeat for all B datasets) to get B models in the ensemble. Finally, we average all the predictions as follows

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

This entire process is called Bagging. And for Classification Trees, we only change a little bit: we record the class predicted by each of the B trees and take a majority vote.

2.2 Out-of-Bag Error Estimation

It turns out that there is a very straightforward way to estimate the test error of a bagged model, without the need to perform cross-validation or the validation set approach. 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. In order to obtain a single prediction for the ith observation, we can average these predicted responses (if regression is the goal) or can take a majority vote (if classification is the goal). This leads to a single OOB prediction for the ith observation. An OOB prediction can be obtained in this way for each of the n observations, from which the overall OOB MSE (for a regression problem) or classification error (for a classification problem) can be computed. The resulting OOB error is a valid estimate of the test error for the bagged model, since the response for each observation is predicted using only the trees that were not fit using that observation. It can be shown that with B sufficiently large, OOB error is virtually equivalent to leave-one-out cross-validation error. The OOB approach for estimating the test error is particularly convenient when performing bagging on large data sets for which cross-validation would be computationally onerous.

2.3 Back to R!

How can we implement Bagging in R? Options:

- Write our own code
  1. Bootstrap the "train" dataset (Z^{*1}, Z^{*2}, ..., Z^{*B})
  2. `for i = 1:B`
     `CART(i) = rpart(y~., train = Z*i)`
  3. Averaging / Majority vote

- Use the function `bagging` (package `ipred`). But note this only works for Decision Trees.

  `model <- bagging(formula, data, coob=TRUE)` Fit a bagging model

  `print(model)` Short summary of model

  `pred <- predict(model, newdata, type)` Predict via bagging

3 Random Forests

Random forests provide an improvement over bagged trees by way of a random small tweak that decorrelates the trees. Suppose that there is one very strong predictor in the data set, along with a number of other moderately strong predictors. Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split. Consequently, all of the bagged trees will look quite similar to each other. Hence the predictions from the bagged trees will be highly correlated. Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities. In particular, this means that bagging will not lead to a substantial reduction in variance over a single tree in this setting.

3.1 Random forests

Random forests overcome this problem by forcing each split to consider only a subset of the predictors. 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 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. On average, $(p - m)$ predictors are not considered (in each model), so other predictors will have a chance.

The fundamental difference between Bagging and Random Forests stands in the subset of predictors m. If $m = p$, then there is no difference between the two methods. Here are some recommended values of m:

- Regression Tree: $m = p/3$
- Classification Tree: $m = \sqrt{p}$

Note that these values were found experimentally, so there is no theoretical guarantee they will provide the best performance on all datasets.

It is common practice to explore the effect of the hyperparameters value on the performance of Random Forests. To recap, we have the following parameters:

- Number of trees, B
- Number of predictors used at each split, $m \leq p$
- (number of points in each terminal leaf)

There are no optimization routines to find their values. We typically use grid search, or similar.

3.2 Back to R!

To learn a Random Forest, we will use the function `randomForest`, implemented in the package ... `randomForest`:

`forest <- randomForest(formula, data, ntree, mtry, ...)` Fit

`predictforest <- predict(forest,newdata,type)` Predict

`importance(forest)` or `varImpPlot(forest)` Variable importance

`varUsed(forest, by.tree=FALSE, count=TRUE)` Frequencies of variables

`forest$err.rate[ntree,1]` OOB error rate

3.3 Advantages and Disadvantages of Random Forests

Pros:

- Better bias-variance trade-off than CARTs
- Higher accuracy (on the test dataset)

Cons:

- Less interpretable
- Higher computational requirements

Reference

[1] James et al. (2014) *An Introduction to Statistical Learning with Applications in R*, Springer, 2014. Chapter 5.2 and 8.2.