Decision Trees, Bagging, Boosting, Random Forest and Extra-Trees

1 Bootstrap

The bootstrap is a general tool for assessing statistical accuracy. First we describe the bootstrap in general, and then show how it can be used to estimate extra-sample prediction error.

Suppose we have a model fit to a set of training data. We denote the training set by $\mathbf{Z} = (z_1, z_2, ..., z_N)$ where $z_i = (x_i, y_i)$. The basic idea is to randomly draw datasets with replacement from the training data, each sample the same size as the original training set. This is done B times (B = 100 say), producing B bootstrap datasets. Then we refit the model to each of the bootstrap datasets, and examine the behavior of the fits over the B replications.

2 Bootstrap Aggregation, i.e. Bagging

How to use the bootstrap to improve the estimate or prediction itself?

Consider first the regression problem. Suppose we fit a model to our training data $\mathbf{Z} = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$, obtaining the prediction $\hat{f}(x)$ at input x. Bootstrap aggregation or bagging averages this prediction over a collection of bootstrap samples, thereby reducing its variance. For each bootstrap sample Z^{*b}, $b = 1, 2, \cdots, B$, we fit our model, giving prediction $\hat{f}^{*b}(x)$. The bagging estimate is defined by

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

The bagged estimate above will differ from the original estimate $\hat{f}(x)$ only when the latter is a nonlinear or adaptive function of the data.

Bagging is a method for reducing the variance of an estimated prediction function. Bagging works well for high-variance, low-bias procedures, such as trees. (It is not specific o trees, it could be applied to regression models, B-Splines, etc.)

For regression tree we simply fit the same regression tree many times too bootstrap sampled versions of training data and average the result. For classification a committee of trees, each cast a vote for the predicted class.

3 Boosting

The motivation for boosting was a procedure that combines the outputs of many "weak" classifiers to produce a powerful "committee." From this perspective boosting bears a resemblance to bagging and other committee-based approaches. However we shall see that the connection is at best superficial and that boosting is fundamentally different.

Lets explain boosting with the famous model of AdaBoost.

Consider a two-class problem, with the output variable coded as $Y \in \{-1, 1\}$. Given a vector of predictor variables X, a classifier $G(X)$ produces a prediction taking one of the two values $\{-1, 1\}$. The error rate on the training sample is

$$\overline{err} = \frac{1}{N} \sum_{i=1}^{N} I(y_i \neq G(x_i))$$

A weak classifier is one whose error rate is only slightly better than random guessing. The purpose of boosting is to **sequentially** apply the weak classification algorithm to repeatedly **modified versions of the data**, thereby producing a sequence of weak classifiers $G_m(x)$, $m = 1, 2, ..., M$. The predictions from all of them are then combined through a weighted majority vote to produce the final prediction:

$$G(x) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

Here $\alpha_1, \alpha_2, ..., \alpha_M$ are computed by the boosting algorithm, and weight the contribution of each respective $G_m(x)$. Their effect is to give higher influence to the more accurate classifiers in the sequence.

The data modifications at each boosting step consist of applying weights $w_1, w_2, /cdots, w_N$ to each of the training observations (x_i, y_i), $i = 1, 2, ..., N$. Initially all of the weights are set to $w_i = 1/N$, so that the first step simply trains the classifier on the data in the usual manner. For each successive iteration $m = 2, 3, ..., M$ the observation weights are individually modified and the classification algorithm is reapplied to the weighted observations. At step m, those observations that were misclassified by the classifier $G_{m-1}(x)$ induced at the previous step have their weights increased, whereas the weights are decreased for those that were classified correctly. Thus as iterations proceed, observations that are difficult to classify correctly receive ever-increasing influence. Each successive classifier is thereby forced to concentrate on those training observations that are missed by previous ones in the sequence. (please take a look at reffig:AdaBoost)

4 RandomForest

Random forests is a substantial modification of bagging that builds a large collection of decorrelated trees, and then averages them. On many problems the performance of random forests is very similar to boosting, and they are simpler to train and tune.

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Figure 1: AdaBoost Algorithm

The essential idea in bagging is to average many noisy but approximately unbiased models, and hence reduce the variance. Trees are ideal candidates for bagging, since they can capture complex interaction structures in the data, and if grown sufficiently deep, have relatively low bias. Since trees are notoriously noisy, they benefit greatly from the averaging. Moreover, since each tree generated in bagging is identically distributed (i.d.), the expectation of an average of B such trees is the same as the expectation of any one of them. This means the bias of bagged trees is the same as that of the individual trees, and the only hope of improvement is through variance reduction. This is in contrast to boosting, where the trees are grown in an adaptive way to remove bias, and hence are not i.d.

An average of B i.i.d. random variables, each with variance σ^2, has variance $\frac{1}{B}\sigma^2$. If the variables are simply i.d. (identically distributed, but not necessarily independent) with positive pairwise correlation ρ, the variance of the average is

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{1}$$

As B increases, the second term disappears, but the first remains, and hence the size of the correlation of pairs of bagged trees limits the benefits of averaging. The idea in random forests is to improve the variance reduction of bagging by reducing the correlation between the trees, without increasing the variance too much. This is achieved in the tree-growing process through random selection of the input variables.

Specifically, when growing a tree on a bootstrapped dataset:

Before each split, select $m \leq p$ of the input variables at random as candidates for splitting.

Typically values for m are $\sqrt{p}$ or even as low as 1.

After B such trees $\{T(x;\Theta_b)\}_1^B$ are grown, the random forest (regression) predictor is

$$\hat{f}_{rb}^B = \frac{1}{B}\sum_{b=1}^{B} T(x;\Theta_b)$$

Θ_b characterizes the bth random forest tree in terms of split variables, cutpoints at each node, and terminal-node values. Intuitively, reducing m will reduce the correlation between any pair of trees in the ensemble, and hence by Eq. 1 reduce the variance of the average.

Not all estimators can be improved by shaking up the data like this. It seems that highly nonlinear estimators, such as trees, benefit the most.

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Figure 2: Random Forest Algorithm

5 ExtraTrees

[2]: The Extra-Trees algorithm builds an ensemble of unpruned decision or regression trees according to the classical top-down procedure. Its two main differences with other tree-based ensemble methods are that it splits nodes by choosing **cut-points** fully at random and that it uses **the whole learning sample** (rather than a bootstrap replica) to grow the trees.

References

[1] Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome The Elements of Statistical Learning.

[2] Geurts, Pierre and Ernst, Damien and Wehenkel, Louis Extremely randomized trees Published online: 2 March 2006 Springer Science + Business Media, Inc. 2006