Model Search and Inference by Bootstrap “Bumping”

By Robert Tibshirani and Keith Knight

A paper summary by

Hong X. Shi
&
Riley A. Metzger

March 2003

For

Dr. Hugh Chipman’s Stat 946 Course
A Summary of:
*Model Search and Inference by Bootstrap “Bumping”*

1. Notation and Assumptions – A Brief Description

As in most papers, there is a set of notation and assumptions that must be covered before we can get into the actual article. The *training sample* of size N will be denoted by the set of values: \( z = (z_1, z_2, \ldots, z_N) \). A particular bootstrap sample will be denoted by \( z^i \). We will assume that each of the \( z_i \) are iid from the same distribution \( F \) and in general that \( z_i = (x_i, y_i) \) where \( x_i \) denotes a vector of *explanatory variates*, \( \theta \), a vector of parameter(s) of our predicting model, and \( y_i \) a *response measurement*.

We also define the *target criterion*, denoted by \( R \), as the criterion that we wish to use to determine the class of our object. On the other hand, \( R_0 \) will be the *working criterion* which will be the criterion used to determine the class of our object. It may be the case that \( R = R_0 \). However, it may also be the case that \( R \neq R_0 \). For the remainder of this paper it is assumed that under a suitable assumption \( R \approx R_0 \).

**Example 1.1:** Suppose that we wish to minimize the target criterion \( R = \sum |y_i - x_i \theta| \). It may (and indeed is) easier to instead minimize working criterion \( R_0 = \sum (y_i - x_i \theta)^2 \).

2. Bagging – A Review

“Boot Strap Aggregation” or Bagging was a concept covered in class. It suggested the use of bootstrap samples to predict data classifiers.

Bagging uses \( B \) bootstrap samples denoted by \( z^i (i = 1, 2, \ldots, B) \) to obtain a prediction:

\[
\hat{f} = \frac{1}{B} \sum_{b=1}^{B} f^b(x) \quad \text{<- notation seen in class.}
\]

The result is a method which is useful in the case of unstable models (such as trees). However, there are some drawbacks. For example, just like the mean of a sample, features of the data have been destroyed. As Brieman is quoted in the paper, in the specific case of tree models, a bagged tree in not a tree. Bumping was suggested as a method which would retain the structure of the original model.

3. Bumping - “Bootstrapping Umbrella of Model Parameters”

We’ll start our description of bumping based on the simplest case and build complexity into the method as necessary. Throughout the discussion I’ll make various references to the least squares method, as it is likely the best known and simplest method with which to discuss bumping.

Suppose that we want to obtain the optimal values of a parameter \( \theta \) that minimizes some function \( R(z, \theta) \). In the case of least squares we have \( R = \sum (y_i - x_i \theta)^2 \). In general we want \( \theta^* = \arg\min_{\theta} R(z, \theta) \).

Now, suppose that we have generated \( B \) bootstrap samples \( z^{b1}, \ldots, z^{bB} \). For each sample \( z^{bi} \) we can obtain an estimate of our parameters:

\[
\hat{\theta}^{*b} = \arg\min_{\theta} R(z^{bi}, \theta) = \arg\min_{\theta} \sum (y_i^{bi} - x_i \theta)^2
\]

This will give us a vector of parameter estimates given by \( (\hat{\theta}^{*b1}, \ldots, \hat{\theta}^{*bB}) \). It makes sense in such a case to select \( \hat{\theta} = \hat{\theta}^* \) (our parameter estimate via bootstraping) from the minimum of these bootstrap estimates.
Now, suppose that instead of \( R = \sum_{i} (y_i - x \theta)^2 \) we have defined \( R = \sum_{i} \left| y_i - x \theta \right| \). In other words, we want to obtain value of \( \theta \) denoted by \( \theta^* \) that minimizes \( R \). We know that although we can minimize \( R \), it may be easier to minimize our earlier \( R \) which we now call:

\[
R_0 = \sum_{i} (y_i - x \theta)^2.
\]

Due to this change, the process has changed minimally. First, use our bootstrap values and \( R_0 \) to obtain a vector of minimum \( \theta^*_{b} = \text{argmin}_{\theta} R_0(z^b, \theta) \) parameter estimates. The final twist is to select as \( \theta^* = \text{argmin}_{b} R(z, \theta^{*b}) \) the smallest of the \( (\theta^{*1}, \ldots, \theta^{*B}) \) estimates in the function \( R \).

To review,

1) We want \( \theta^* = \text{argmin}_{\theta} R(z, \theta) \)
2) Estimate \( \theta^* \) via \( \theta^{*b} = \text{argmin}_{\theta} R_0(z^b, \theta) \)
3) Then take \( \theta^* = \text{argmin}_{b} R(z, \theta^{*b}) \)

This paper investigated Bumping as it related to regression and classification. In such cases, as mentioned above, we define \( z_i = (x_i, y_i) \). Using the notation from class we can then define \( L(y_i, f(x_i, \theta)) \) as a loss function and naturally set \( R_0 = \sum_{i} L(y_i, f(x_i, \theta)) \).

Notes:
- It has been found that as many as 20-30 bootstrap samples are necessary in unconstrained problems but that constrained problems require about 100-200 bootstrap samples.
- The original data \( B' \) is always included in the set of bootstrap samples.
- If \( R = R_0 \) and the procedure always finds its global minimum (Least Squares) then bumping will give the global minimum (recall that the original sample is included in all bootstrap sets and that \( \theta^{*b} \) is chosen from \( R \) using the original data).
- In the case of smooth functions with many local minima it is hoped that bumping will find the ‘best’ local minima.
- When \( R \) and \( R_0 \) are similar but \( R \) is difficult to minimize, the fact that we can use \( R_0 \) instead makes things easier.

4. Model Complexity

One of the issues related to models such as trees is model complexity. In general we attempt to penalize the complexity of our models by adding a complexity parameter \( \lambda \). Since highly complex models tend to have less error they are very likely to be favoured using the bumping method as defined above. One of the ways in which we can counter this situation is by defining the complexity parameter before we bump. Thus we will only compare bumped estimates if they come from models with the same complexity parameter \( \lambda \). Therefore, in models with complexity parameters, we must first determine the optimal complexity, and then bump.

Our new bumping algorithm becomes,

1) Estimate the optimal complexity parameter \( \lambda \) using cross validation.
2) We want \( \theta^* = \text{argmin}_{\theta} R(z, \theta) \)
3) Estimate \( \theta^* \) via \( \theta^{*b} = \text{argmin}_{\theta} R_0(z^b, \theta) \)
4) Then take \( \theta^* = \text{argmin}_{b} R(z, \theta^{*b}) \)

Example 4.1: Suppose we have a tree \( T \) with cost-complexity parameter \( \lambda \) and size \(|T|\). Then from class we saw that \( C_\lambda(z, T) = C(z, T) + \lambda |T| \). The paper suggests that we first, using cross validation, obtain \( \lambda^* \) and they compute \( T_\lambda(z^*) \) from each bootstrap sample that minimizes our criterion \( R(z, T) \).
5. Bumping is Robust

One of the benefits of bumping is that, due to its relationship with bootstrapping, it is fairly robust. Suppose for example that you have a set of data such that \( x = (0, 1, 2, 3, 4, 5, 6, 7, 60, 63) \) and \( y = (5, 6, 7, 8, 9, 10, 11, 12, 1, 1) \) to which you wanted to fit a least squares model. Since we are running bootstrap samples, there is a possibility that these two points will not be in any of our samples.

\[
\Pr (\text{a point } i \text{ of } N \text{ is in the sample}) = \frac{1}{N} \\
\Pr (\text{point } i \text{ is not in the sample}) = 1 - \frac{1}{N} \\
\Pr (\text{point } i \text{ is not in any of the samples}) = \left(1 - \frac{1}{N}\right)^N \\
\Pr (\text{point } i \text{ is in all samples}) = 1 - \left(1 - \frac{1}{N}\right)^N = 1 - \exp(-1) = 63.2\%
\]

Therefore there is a positive probability that outliers won’t be in the sample.

In general, for \( k \) outliers we write:

\[
\Pr (\text{our } k \text{ points are in all samples}) = 1 - \left(1 - \frac{k}{N}\right)^N \\
= 1 - \exp(-k) \\
\geq 63.2\%
\]

Thus, this method is robust for small \( k \). However, for a large number of outliers, the probability that all of the outliers will be omitted from all samples tends to zero.

**Example 5.1:** Using the above plot, we have \( \Pr (x=60, 63 \text{ are not in any of the samples}) = 1 - \exp(-2) = 86\% \)

6. Bumping Simplifies Constrained Optimization

In addition to being more robust bumping makes constrained optimization easier to handle. Suppose for example that you have a set of constraints \( K \) which \( \theta \) must satisfy. These constraints can be of any form (examples include inequality, equality etc). In general then we would define our working criterion in a constrained problem as:

\[
R = R' + \alpha I(\theta \notin K)
\]

Thus we penalize our target criterion based on whether or not \( \theta \) satisfies the \( K \) constraints. This type of problem is generally fairly difficult to solve. However, bumping suggests a method by which we can avoid this issue.

Our new bumping algorithm with a slight modification becomes,

1) Estimate the optimal complexity parameter \( \lambda \) using cross validation.
2) We want \( \theta^* = \arg\min_\theta R(z, \theta) \)
3) Define \( R_0 = R' \)
4) Estimate \( \theta^* \) via \( \theta^{\star h} = \arg\min_\theta R_0(z^{\star h}, \theta) \)
5) Then take \( \theta^{\star h} = \arg\min_\theta R(z, \theta^{\star h}) \)
Bumping has therefore avoided the issues related to minimizing a constrained problem by minimizing a surrogate problem and simply ensuring the end result satisfies the K constraints.

7. Bumping Confidence Sets

Inference can be carried out via bumping parameter estimates. The following sections will attempt to give methods and reasons.

Let \( (\theta^*(1), \ldots, \theta^*(B)) \) be an ordered set of parameter estimates based on the B bootstrap samples. If \( \alpha \) is a significance level then we define our confidence set as the first \( \alpha B \) parameter estimates. For example the 1-\( \alpha \) confidence set is:

\[
C^* = \{ \theta^*(1), \ldots, \theta^*(\alpha B) \}
\]

Example 7.1: Suppose we obtained \( (\theta^*(1), \ldots, \theta^*(B)) = (1,2,3,\ldots,100) \). For \( \alpha = 0.05 \) we have \( C^* = \{ \theta^*(1), \ldots, \theta^*(0.05B) \} = \{1,2,3,4,5\} \).

In general we suppose \( \text{Prob}^* (\text{R}(z, \theta^* o) \geq r) = 1-\alpha \) where \( \text{Prob}^* \) denotes a bootstrap sample with training set \( z \) fixed. We define \( C^* = \{ \theta^* : \text{R}(z, \theta^*) \geq r \} \). If we say \( g(.) \) is a monotone increasing function then the order will not change and we have:

\[
g(-\text{R}(z, \theta^*)) - g(-\text{R}(z, \theta^* o)) \sim F
\]

And this looks very much like the likelihood ratio statistic. The paper says that if \( R=R_0 = -2 \) times the log likelihood then it is chi-squared in the limit.

8. Bumping and Asymptotics

Define \( \theta^* n = \theta^* \) based on \( R \) and \( \theta^{**} n = \theta^{**} \) based on \( R_0 \). \( \theta^* \) is of course the true parameter value. Since \( \theta^* \) is meant to estimate \( \theta \) this section goes over when it is appropriate to use the bumped estimate \( \theta^{**} \) instead of the normal estimated parameters \( \theta^* \).

Let \( \{ a_n \} \) be a set of constants such that \( U_n = a_n (\theta^* n - \theta^* o) \rightarrow_d U \) and we want \( U^{**}_n = a_n (\theta^{**} n - \theta^* o) \rightarrow_d U \). Thus, it is hoped that \( U_n - U^{**}_n = a_n (\theta^{**} n - \theta^* n) = 0 \) for some bootstrap estimate.

The direct theorem that specifies when exactly this occurs involves measure theory and compact sets as well as a generalized stochastic process \( Z \). To save space this theorem has been omitted. However, the general idea is given in the previous paragraph.

9. An Example

9.1 The Problem

As the samples in the article are not clear, and, as we already have a complete description of the spam data we will also use the spam data as one of our examples. Our second example, given in class, was chosen from the article so it will be omitted here.

9.2 The R Code

```r
spam.test.y <- spam.test$spam
spam.train$spam <- as.factor(spam.train$spam)
spam.test$spam <- as.factor(spam.test$spam)
```
```r
cp1 <- 0.00001
n1 <- 100
min1 <- 1.0
p2 <- matrix(0,n1,3)

for (i in 1:n1){
    print(i)
    which <- sample(nrow(spam.train),rep=T)
    tempmodel <- rpart(spam~.,data=spam.train[which,,cp=cp1,
        method='class', pars=list(split='information') )
    tempprun <- prune(tempmodel, tempmodel$cptable[tempmodel$cptable[,4]
    == min(tempmodel$cptable[,4],1)[1])
    if (min1 > min(tempprun$cptable[,4])) {
        min1 <- min(tempprun$cptable[,4])
        best <- tempprun
    }
    preds1 <- predict(tempprun,newdata=spam.test,type='matrix')
    preds2 <- predict(tempprun,newdata=spam.test,type='class')
    t1 <- table(predicted=preds2,actual=spam.test$spam)
    p2[i,1] <- gain1(spam.test.y, preds1)
    p2[i,2] <- bindev(spam.test.y, preds1)
    p2[i,3] <- mis <- t1[1,2] + t1[2,1]
}
write.table(p2, "p1_p2_200.csv", sep=",")
```
- Within class

<table>
<thead>
<tr>
<th></th>
<th>precision</th>
<th>deviance</th>
<th>misclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single rpart</td>
<td>520.2</td>
<td>729.7</td>
<td>115</td>
</tr>
<tr>
<td>Bag small trees</td>
<td>545.7</td>
<td>678.7</td>
<td>117</td>
</tr>
<tr>
<td>Bag big trees</td>
<td>554.9</td>
<td>621.6</td>
<td>100</td>
</tr>
</tbody>
</table>

**BUMPING:**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>By misclass.</td>
<td>506.2086</td>
<td>785.7541</td>
<td>114</td>
</tr>
<tr>
<td>By deviance</td>
<td>520.5463</td>
<td>763.1782</td>
<td>122</td>
</tr>
<tr>
<td>By precision</td>
<td>535.0994</td>
<td>797.0884</td>
<td>120</td>
</tr>
</tbody>
</table>