Paper 1300-2014

Model Variable Selection Using Bootstrap Decision Tree

David J. Corliss, Magnify Analytic Solutions

ABSTRACT

Bootstrapped Decision Tree is a variable selection method used to identify and eliminate unintelligent variables from a large number of initial candidate variables. Candidates for subsequent modeling are identified by selecting variables consistently appearing at the top of decision trees created using a random sample of all possible modeling variables. The technique is best used to reduce hundreds of potential fields to a "short list" of 30–50 fields to be used in developing a model. This method for variable selection has recently become available in JMP® under the name BootstrapForest; this paper presents an implementation in Base SAS®9. The method does accept but does not require a specific outcome to be modeled and will therefore work for nearly any type of model, including segmentation, MCMC, multiple discrete choice, in addition to standard logistic regression.

Keywords: Bootstrapped Decision Tree, Variable Selection

INTRODUCTION

A Bootstrapped Decision Tree is an ensemble learning technique for selecting variables to model development. The mathematical framework of this technique was developed by Leo Breiman and Adele Cutler in 2001, who trademarked a name for their implementation. The random selection of candidate model factors uses Tin Kam Ho's Random Subspace method (1995) as a means of stochastic discrimination (E. Kleinberg, 1996).

Since the first implementations of bootstrapped decision trees in the early 1990's, applications have been developed for several platforms, including JMP Pro under the name "Bootstrap Forests". This paper introduces bootstrapped decision tree functionality in Base SAS.

The centerpiece of the process is a decision tree halted after only a single step. Of all the possible variables available for the development of a model, only a handful are used in the decision tree. PROC FASTCLUS divides the dataset into two subsets that are the most different, using a single variable as the basis of the distinction. In this way, the variables used in the decision tree effectively compete against one another for top placement in the tree. Poor candidates for modeling often will have limited ability to inform or distinguish between records and therefore rarely will be found at the top of the decision tree. The frequency at which given variable is selected from among a small number of randomly selected candidates is indicative of its value in modeling. This is relative indication, not an absolute measure. As a result, the method is effective only for the preliminary step of identifying a short list of top candidates for subsequent modeling, not a final list of the variable to be used.

This implementation of the bootstrapped decision tree process in Base SAS with macro code uses the following steps:

1. The PROC FASTCLUS in this process only operates on numeric variables. Accordingly, any categorical variables to be included must first be put in a numeric format such as 0 / 1 instead of Y / N.
2. PROC CONTENTS to create a list of the variables available for modeling
3. A random subset of variable names is selected
4. PROC FASTCLUS is used to create a decision tree with the selected variables
5. The process captures the name of the variable selected for the first split – this variable gets one "vote"
6. The random selection – Decision Tree – First Variable steps are bootstrapped: these steps are repeated many times (e.g., 10,000), with the variable at the top of the decision tree getting one vote each time
7. Candidate variables are ranked by the number of votes received by each

MACRO SET-UP

Once any character variables of interest have been converted to a numerical format, the bootstrap macro is executed using these parameters:

```
%macro BDT(lib,file,varnum,rep);
```

where **lib** and **file** are the SAS library and dataset to be used and **rep** is the number of repetitions of the bootstrap process. Typical values of **rep** range from 5,000 to 25,000.

varnum is the sample size for the random variable selection, which should be chosen with some care. If too large a number is taken, a small number of the very best variables will be selected frequently, diminishing the ability of the process to accurately indicate the performance of middle-ranking variables. If **varnum** is too small, lower ranking variables will seldom be compared to high ranking ones, inflating the performance of weaker candidates. In practice, preferable values of **varnum** may be assessed by brute force modeling using all possible candidates, with an optimal value of **varnum** producing a distribution similar to variable contributions to the final model. Clearly, this should not be done frequently, with the results of such a test used as guidance is the development of future models. Performance testing of this macro leads to the recommendation of a **varnum** between 10% and 25% of the total number of variables, with larger values preferable for datasets with fewer variables (less than 200). The process has been successfully tested in datasets with as few as 60 variables and as many as 1,300.

VARIABLE LIST USING PROC CONTENTS

The PROC CONTENTS with an OUT statement writes the procedure output to a SAS dataset. The KEEP statement retains only the name of each variable along with the variable type – 1 for numeric and 2 for character fields.

```
proc contents data=&lib..&file.
   out=work.vars(keep=name type) noprint;
run;
```

"VOTE" COUNTER

A variable receives one "vote" for each time it is selected by the PROC FASTCLUS in the first split in the decision tree. This data step creates a SAS dataset for tracking the number of votes with one record for each numeric variable, initializing the number of "votes" to zero.

```
data work.vars;
   set work.vars;
   rename name = variable_name;
   if type = 1;   /* Numeric variables only */;
   vote_count = 0;
run;


data work.var_votes;
   set work.vars;
   if _n_ < 1;
   keep variable_name;
run;
```

CREATION OF VARIABLE LIST FOR THE DECISION TREE

The bootstrap portion of the process begins with selection of a random sample of the variables. The same number of variables is selected each time, as set by the macro parameter **varnum**. This is accomplished by assigning a random number to each variable (that is, each record in the variable list) and sorting by the random number. A data step is used to capture the number of rows given by **varnum**. Since the order of the variables is random, the select of the first **varnum** rows constitutes a random sample of the variables. The names of the variables selected are concatenated into a string and converted into a macro variable named **varlist**. This list of randomly selected variables is written to the log.

```
%do i = 1 %to &rep.;
   data work.var_list;
      set work.vars;
      random = ranuni(0);
   run;
   proc sort data=work.var_list;
      by random;
   run;

   data work.test_list;
      set work.var_list (obs=&varnum. keep=variable_name);
   run;

   data _null_;
      length allvars $1000;
      retain allvars ' ';
      set work.test_list end=eof;
      allvars = trim(left(allvars))||' '||left(variable_name);
      if eof then call symput('varlist', allvars);
   run;
   %put &varlist;
```

ONE-STEP DECISION TREE

PROC FASTCLUS is used to create a decision tree using the randomly selected list of variables. The statement maxclusters=2 results in only two output clusters and consequently only one split.

```
proc fastclus data=&lib..&file. maxclusters=2
      outstat=work.cluster_stat noprint;
      var &varlist;
run;
```

The summary statistics from the PROC FASTCLUS are written to a SAS dataset. This produces statistics for each variable – not just the one selected for the first split. The dataset output by PROC FASTCLUS contains one column for each variable. The identity of the variable selected for the first split is extracted by transposing the data, resulting on one row for each candidate variable.

```
   data work.rsq;
         set work.cluster_stat;
         if _type_ = 'RSQ';
```

```
        drop _type_ cluster over_all;
     run;

    proc transpose data=work.rsq out=work.rsq2;
    run;

    data work.rsq2;
       set work.rsq2;
       length variable_name $32.;
       variable_name = _name_;
    run;
```

COUNTING "VOTES"

The output from the PROC TRANSPOSE has one record for each variable, with the variable selected at the top of the decision tree having the largest value of COL1. The TRANSPOSE output is sorted by descending COL1 and the name of the variable at the top of the decision tree appended to an output file. Each repetition of the bootstrap process results in a new random sample of variables and a new decision tree, with the name of first variable selected by the PROC FASTCLUS added to the end of the vote text file. The frequency distribution of the variable names gives a ranked list of the estimated discriminatory value of the variables.

```
    proc sort data=work.rsq2;
       by descending col1;
    run;

    data work.var_votes;
       set work.var_votes work.rsq2(obs=1);
    run;
%end;

%mend BDT;

 proc freq data=work.var_votes order=freq;
   table _name_;
run;
```

```
pop_count        15,805
hh_income         8,126
pop_change_pct    4,805
unemploy_pct      3,287
.
.
.
```

Output 1. An example of system output

The PROC FREQ produces a list of variables ranked by the number of time each was selected by the FASTCLUS

COMPLETE MACRO CODE

```
%macro BDT(lib,file,varnum,rep);

proc contents data=&lib..&file.
   out=work.vars(keep=name type) noprint;
run;

data work.vars;
   set work.vars;
   rename name = variable_name;
   if type = 1;   /* Numeric variables only */;
   vote_count = 0;
run;

data work.var_votes;
   set work.vars;
   if _n_ < 1;
   keep variable_name;
run;

%do i = 1 %to &rep.;
   data work.var_list;
      set work.vars;
      random = ranuni(0);
   run;
   proc sort data=work.var_list;
      by random;
   run;

   data work.test_list;
      set work.var_list (obs=&varnum. keep=variable_name);
   run;

   data _null_;
      length allvars $1000;
      retain allvars ' ';
      set work.test_list end=eof;
      allvars = trim(left(allvars))||' '||left(variable_name);
      if eof then call symput('varlist', allvars);
   run;
```

```
   %put &varlist;

   proc fastclus data=&lib..&file. maxclusters=2
      outstat=work.cluster_stat noprint;
      var &varlist;
   run;

   data work.rsq;
         set work.cluster_stat;
         if _type_ = 'RSQ';
         drop _type_ cluster over_all;
    run;

   proc transpose data=work.rsq out=work.rsq2;
   run;

   data work.rsq2;
      set work.rsq2;
      length variable_name $32.;
      variable_name = _name_;
   run;

   proc sort data=work.rsq2;
      by descending col1;
   run;

   data work.var_votes;
      set work.var_votes work.rsq2(obs=1);
   run;
%end;
%mend BDT;
```

A TYPICAL IMPLEMENTATION OF THE BOOTSTRAP DECISION TREE PROCESS

Here is a typical implementation of the process, with operational recommendations:

```
* Run a sample with just a few repetitions as a test (in this example, 10);

%bdt(projlib,customers,12,10);

* Write the log to a file – needed to avoid filling the log buffer during
bootstrapping;
```

```
PROC PRINTTO LOG='V:\Whirlpool\Models\Segmentation\bootstrap_log.log' NEW;
RUN;

* Final run using a large number of repetitions (in this example, 10,000);

%bdt(projlib,customers,12,10000);

proc printo; /* Turn off PROC PRINTTO, restoring the log to usual location */;
run;
```

CONCLUSION

The Bootstrapped Decision Tree process selects candidates for subsequent modeling by identifying variables consistently appearing at the top of decision trees. As only numeric variables are supported by the FASTCLUS procedure used in this macro, character variables must be converted to a numeric format before running the process. This process identifies only identifies candidates for subsequent modeling and should not be used to as a final step in variable selection. The technique is most effective when used to reduce a large number of possible fields – hundreds or more – to a few dozen candidate variables.

This non-parametric unsupervised learning process supports identification of preferred candidate variables for many different types of models. This decision tree process addresses highly correlated variables by testing them directly against each other. The process is supported in both the Base SAS® and the JMP Pro® environments.

REFERENCES

- Breiman, Leo (2001), "Random Forests", Machine Learning 45, 1, 5
- Ho, Tin Kam (1995), "Random Decision Forest", Proc. 3rd International Conference on Document Analysis and Recognition, Montreal, QC, pp. 278–282
- Kleinberg, Eugene (1996), "An Overtraining-Resistant Stochastic Modeling Method for Pattern Recognition", Annals of Statistics 24,6, 2319

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

David J Corliss
Wayne State University / Physics and Astronomy
666 Hancock St.
Detroit, MI 48202
Phone: 734.837.9323
Email: david.corliss@wayne.edu, davidjcorlis@gmail.com
sascommunity.org: dcorliss