2015

Case-Specific Random Forests for Big Data Prediction

Joshua Zimmerman
Iowa State University, joshuaz@iastate.edu

Dan Nettleton
Iowa State University, dnett@iastate.edu

Case-Specific Random Forests for Big Data Prediction

Abstract
Some training datasets may be too large for storage on a single computer. Such datasets may be partitioned and stored on separate computers connected in a parallel computing environment. To predict the response associated with a specific target case when training data are partitioned, we propose a method for finding the training cases within each partition that are most relevant for predicting the response of a target case of interest. These most relevant training cases from each partition can be combined into a single dataset, which can be a subset of the entire training dataset that is small enough for storage and analysis in memory on a single computer. To generate a prediction from this selected subset, we use Case-Specific Random Forests, a variation of random forests that replaces the uniform bootstrap sampling used to build a tree in a random forest with unequal weighted bootstrap sampling, where training cases more similar to the target case are given greater weight. We demonstrate our method with an example concrete dataset. Our results show that predictions generated from a small selected subset of a partitioned training dataset can be as accurate as predictions generated in a traditional manner from the entire training dataset.

Keywords
bootstrap, machine learning, parallel computing

Disciplines
Categorical Data Analysis | Computer Sciences | Statistical Methodology | Statistical Models

Comments
This proceeding is published as Zimmerman, J., Nettleton, D. (2015). Case-specific random forests for big data prediction. In *JSM Proceedings,* General Methodology. Alexandria, VA: American Statistical Association, pp. 2537–2543. Posted with permission.

Case-Specific Random Forests for Big Data Prediction

Joshua Zimmerman* Dan Nettleton†

Abstract

Some training datasets may be too large for storage on a single computer. Such datasets may be partitioned and stored on separate computers connected in a parallel computing environment. To predict the response associated with a specific target case when training data are partitioned, we propose a method for finding the training cases within each partition that are most relevant for predicting the response of a target case of interest. These most relevant training cases from each partition can be combined into a single dataset, which can be a subset of the entire training dataset that is small enough for storage and analysis in memory on a single computer. To generate a prediction from this selected subset, we use Case-Specific Random Forests, a variation of random forests that replaces the uniform bootstrap sampling used to build a tree in a random forest with unequal weighted bootstrap sampling, where training cases more similar to the target case are given greater weight. We demonstrate our method with an example concrete dataset. Our results show that predictions generated from a small selected subset of a partitioned training dataset can be as accurate as predictions generated in a traditional manner from the entire training dataset.

Key Words: bootstrap, machine learning, parallel computing

1. Introduction

As recent advances in technology have allowed larger and larger datasets to be collected and stored, it has become important to be able to handle and process such large datasets so that valid conclusions, inferences, or predictions can be made from them. In situations where large training datasets cannot be stored on a single computer, new methods for prediction are needed. In this paper, we assume a training dataset is partitioned in multiple disjoint subsets. Each subset is assumed to be small enough that it may be stored and processed on a single computer within a parallel computing environment. We address the problem of predicting the response of a target case in this scenario.

We propose a method that focuses on selecting a subset of the partitioned training dataset that is most relevant for predicting the response of the target case. In many situations, this selected subset may be substantially smaller than the entire training dataset without compromising prediction accuracy. Our subset selection is accomplished by searching each partition of the training dataset in parallel. Thus, large training datasets can be scanned quickly without the need for operating on any dataset larger than the largest subset in the partitioned training datatset. We limit the size of our selected subset to be no larger than a size that can be handled easily by a single computer. Thus, sophisticated parallel computing techniques are not needed to generate the prediction of the target response once the relevant training data subset has been identified.

*Iowa State University, Department of Statistics, Ames, IA 50011
†Iowa State University, Department of Statistics, Ames, IA 50011

2. Proposed Method

Let $\boldsymbol{\mathcal{C}} = \{(\boldsymbol{x}_i, y_i) : i = 1, \ldots, N\}$ be a training dataset consisting of N observations of a predictor vector $\boldsymbol{x}$ and a corresponding response y. Using the training dataset $\boldsymbol{\mathcal{C}}$, we want to predict the response y_0 corresponding to an observed predictor vector $\boldsymbol{x}_0$. Suppose $\boldsymbol{\mathcal{C}}$ is partitioned into K disjoint subsets $\boldsymbol{\mathcal{C}}^{(1)}, \ldots, \boldsymbol{\mathcal{C}}^{(K)}$ for storage and parallel computation. One solution to this problem would be to predict y_0 separately using each of $\boldsymbol{\mathcal{C}}^{(1)}, \ldots, \boldsymbol{\mathcal{C}}^{(K)}$ and then average the K predictions. However, some subsets of $\boldsymbol{\mathcal{C}}$ may contain few cases relevant for prediction of y_0, which could lead to inaccurate predictions of the response for some subsets and, thus, an inaccurate average prediction. To address this problem, we first collect the most relevant cases from each subset and then employ a method of prediction that further concentrates on the most relevant cases in the collection. In our first step, we identify the training caseses in $\boldsymbol{\mathcal{C}}$ whose $\boldsymbol{x}$ vectors are "close" to $\boldsymbol{x}_0$ and collect those cases in a relatively small subset of training cases denoted as $\boldsymbol{\mathcal{C}}(\boldsymbol{x}_0)$.

2.1 Finding $\boldsymbol{\mathcal{C}}(\boldsymbol{x}_0)$

To determine $\boldsymbol{\mathcal{C}}(\boldsymbol{x}_0)$, the subset of training cases whose $\boldsymbol{x}$ vectors are "close" to $\boldsymbol{x}_0$, we implement the following steps:

1. For $k = 1, ..., K$, use $\boldsymbol{\mathcal{C}}^{(k)}$ to build a Random Forest RF$^{(k)}$.
2. Let $n^{(k)}(\boldsymbol{x}_0, \boldsymbol{x}_i)$ be the number of trees in RF$^{(k)}$ where $\boldsymbol{x}_0$ ends up in a terminal node containing $\boldsymbol{x}_i$.
3. Include $\boldsymbol{\mathcal{C}}_i = (\boldsymbol{x}_i, y_i)$ in $\boldsymbol{\mathcal{C}}(\boldsymbol{x}_0)$ if and only if $n^{(k)}(\boldsymbol{x}_0, \boldsymbol{x}_i)$ is among the highest h of the values $n^{(k)}(\boldsymbol{x}_0, \boldsymbol{x}_1), ..., n^{(k)}(\boldsymbol{x}_0, \boldsymbol{x}_N)$ for some $k = 1, ..., K$.

After employing steps 1–3, $\boldsymbol{\mathcal{C}}(\boldsymbol{x}_0)$ consists of the hK training cases obtained by collecting the h cases with values of $\boldsymbol{x}$ in closest proximity to $\boldsymbol{x}_0$ from each of the K training data subsets. The number of observations within $\boldsymbol{\mathcal{C}}(\boldsymbol{x}_0)$ is determined by h, which can be set to any manageable value. For example, a user may select h so that hK is small enough to allow the step of predicting y_0 with $\boldsymbol{\mathcal{C}}(\boldsymbol{x}_0)$ to be executed on a single computer.

2.2 Prediction of y_0 with $\boldsymbol{\mathcal{C}}(\boldsymbol{x}_0)$

We use Case-Specific Random Forests (Xu et al., 2015) to carry out the prediction step. Case-Specific Random Forests (CSRFs) are very similar to random forests (Breiman, 2001), with the only difference being that we replace the uniform equal probability bootstrap resampling used to grow trees in a random forest (RF) with unequal probability bootstrap resampling. Training cases whose $\boldsymbol{x}$ vectors are in closer proximity to the target vector $\boldsymbol{x}_0$ are resampled with higher probability. Xu et al. (2015) show that a CSRF can yield better prediction performance for specific cases than an RF.

To determine the bootstrap resampling probabilities for a CSRF, we first grow a forest from the selected training dataset $\boldsymbol{\mathcal{C}}(\boldsymbol{x}_0)$. Let $n(\boldsymbol{x}_0, \boldsymbol{x}_i)$ denote the number of trees containing both $\boldsymbol{x}_0$ and $\boldsymbol{x}_i$ in the same terminal node. For $i = 1, ..., N$, let

$$p_i(\boldsymbol{x}_0) = \frac{n(\boldsymbol{x}_0, \boldsymbol{x}_i)}{\sum_{j=1}^{N} n(\boldsymbol{x}_0, \boldsymbol{x}_j)}.$$

Then we proceed to construct a random forest as usual except, when drawing the bootstrap sample of the training cases prior to the construction of each tree, we sample case i with

probability $p_i(\boldsymbol{x}_0)$ rather than using uniform equal probability bootstrap resampling for all of the cases. This means that training cases with closest proximity to the test case are more likely to be selected in the bootstrap sample, and training cases with low proximity to the test case are unlikely to be selected in the bootstrap sample. This allows us to grow trees in the forest using training cases that should be more similar to the target case for which a prediction of the response is desired. Then we predict y_0 from the resulting forest by averaging over all of the predictions generated by the trees in the forest. Because the trees in the forest consist of mostly training cases with close proximity to the test case, we can see a reduction in the prediction error compared to a regular random forest.

3. An Example Analysis of a Concrete Dataset

To demonstrate our methods, we use a concrete dataset (Yeh, 1998) as an example. This concrete dataset is commonly used for evaluation of machine learning algorithms and is available on the web from the UC Irvine Machine Learning Repository. The purpose of this dataset is to predict the compressive strength of a concrete sample given various factors that are known to affect its compressive strength, such as the age of the concrete and the amount of cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate that is present in the concrete sample. Although this dataset is relatively small and can easily be stored in memory on a single computer, we use this dataset to demonstrate the same ideas and methods that we would use given a dataset so large that it must be partitioned and distributed across multiple computers. The concrete dataset consists of $N = 1030$ observations on $p = 8$ predictor variables and a quantitative response (compressive strength with distribution shown in Figure 1). We randomly selected 30 cases to use as test cases, and the remaining 1000 cases served as the training dataset $\boldsymbol{\mathcal{C}}$. For each of 500 random partitions of $\boldsymbol{\mathcal{C}}$ into $K = 10$ subsets of 100 observations, we predicted each of the 30 test cases using the following methods:

1. RF using all 1000 training cases (RF).
2. CSRF using all 1000 training cases (CSRF).
3. CSRF using the $h = 10$ training cases with closest proximity to each test case from each of the $K = 10$ training data subsets (CSRFh10).
4. Average prediction from RFs applied separately to each of the $K = 10$ training data subsets (RFa).

To evaluate prediction accuracy for each method, we calculated the mean squared prediction error (MSPE) across the 500 partitions of the training dataset, separately for each test case. The average of MSPE values across the 30 test cases is provided in Table 1. Of the four methods, CSRF had the smallest average MSPE, but our proposed method CSRFh10 had only slightly larger average MSPE. Considering that CSRFh10 never uses more than a tenth of the entire training dataset on any one computer, it seems to perform pretty well compared to the other methods. Also, for some of the test cases, CSRFh10 produced more accurate predictions than RF or CSRF using all 1000 training cases. Figure 2 shows a boxplot of the log ratio MSPE for RF relative to MSPE for each of the other prediction methods. The distribution of log ratios indicates that CSRF improved on RF predictions for slightly more than half the cases. Similary, CSRFh10 had an advantage over RF for half the cases, and sometimes that advantage was substantial. For example, the boxplots of predictions for Case 24 in Figure 3 shows that the predictions using CSRFh10 were not that different from the predictions produced by CSRF on the entire dataset, while RF and

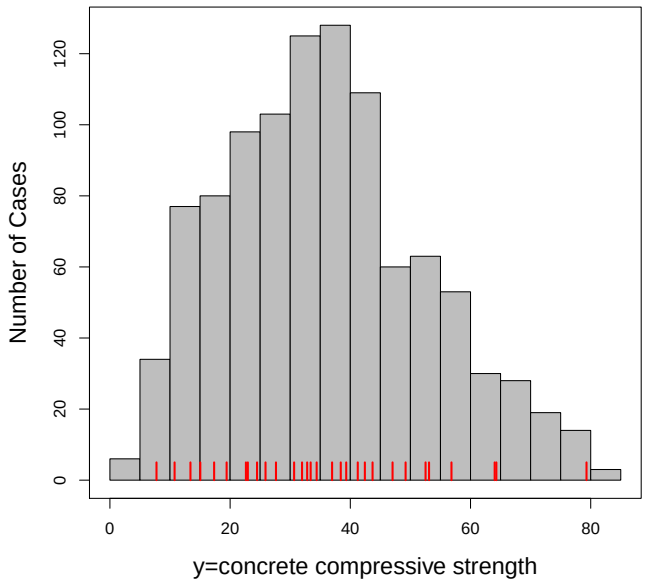

Figure 1: Training Response Distribution with Test Case Responses

RFa predictions were pulled downwards towards the overall average response in the entire training dataset.

4. Discussion

The analysis of the concrete dataset shows that our proposed method of selecting the most relevant cases from each subset of a partitioned training dataset, and then using those cases to predict a target response with CSRF, can produce predictions comparable to those produced by either RF or CSRF applied the entire training dataset. Thus, our proposed approach is useful, particularly when the size of a training dataset makes partitioning necessary. Our results also show that averaging predictions from RFs applied separately to each subset of the partitioned dataset may perform poorly. Part of the reason for the poor

Method	RF	CSRF	CSRFh10	RFa
Average MSPE	27.8	26.0	27.7	71.5

Table 1: Table of Average Mean Squared Prediction Error for each method

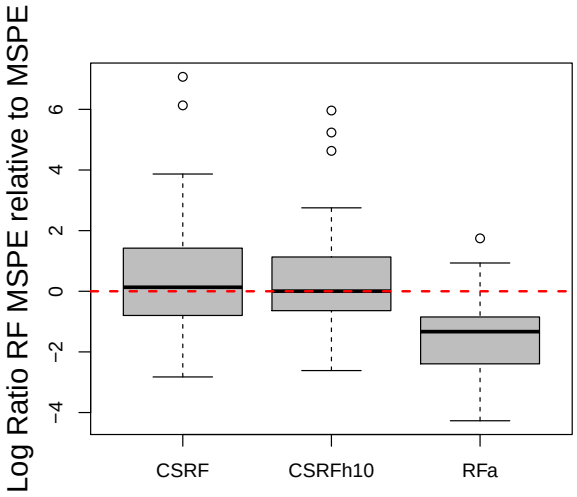

Figure 2: Comparison of MSPE for Competing Methods

performance of the averaging approach may be the relatively small number of observations (100) in each subset of the partitioned training dataset in our example. We would expect the simple averaging approach to improve as the size of the smallest subset grows, but we would not expect the approach to outperform our proposal when signal to noise ratio is sufficiently high.

Each of the methods we have considered generates a prediction that is simply a weighted average of training case responses. Averaging separate RF predictions tends to produce weights that are distributed more broadly across training cases than the proposed method,

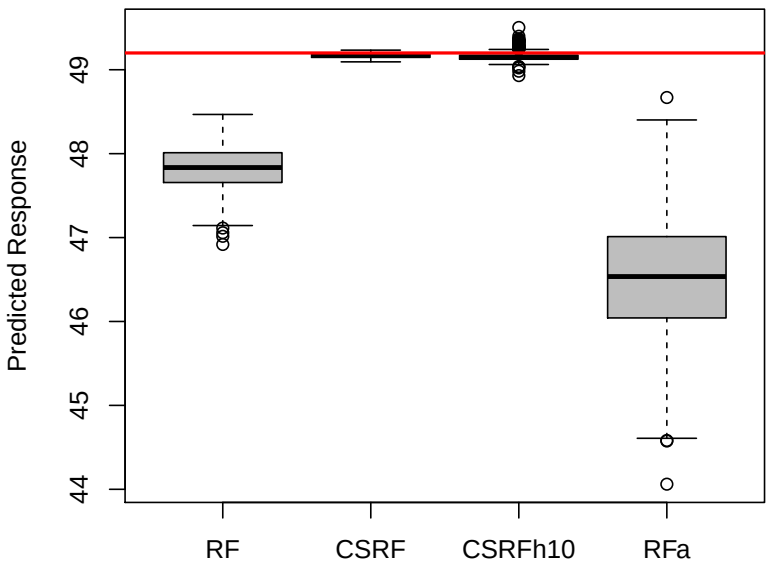

Figure 3: Boxplot of Predictions for Case 24

which attempts to concentrate weights on the most relevant training cases and, by design, completely ignores at least $N - hK$ of the training cases. The concentration of weights can be especially useful for prediction of y_0 when $\boldsymbol{x}_0$ is in close proximity to only a small subset of training cases. In such situations, reductions in prediction bias by the proposed approach can more than compensate for increases in prediction variance that result from concentrating weights on fewer training cases.

To simplify our presentation above, we suppressed details on the number of trees per

forest, the number of predictor variables considered when performing node splits, and the rule used to decide when to cease node splitting during tree construction. For the sake of completeness, we provide those details here. All forests in this paper were constructed from 500 trees. When generating a forest for prediction, we used fully grown trees and considered two randomly selected predictor variables to determine each node split. When growing forests for determining proximities between the target case and training cases, all eight predictor variables were scanned to find the best possible split at each node, and only nodes with more than 10 cases were split. Reasons for these choices are hinted at by the work of Ruo et al. (2015). However, additional research is needed to determine how to most effectively set these tuning parameters.

REFERENCES

Breiman, L. (2001). Random forests. *Machine Learning*. 45, 5–32.

Xu, R., Nettleton, D., Nordman, D. J. (2015). Case-specific random forests. *Journal of Computational and Graphical Statistics*. Accepted.

Yeh, I. (1998). Modeling strength of high performance concrete using artificial neural networks. *Cement and Concrete Research*. 28, 1979–1808.