Journal of the Korean Data & Information Science Society
2010, **21**(2), 371–377

한국데이터정보과학회지

Simple hypotheses testing for the number of trees in a random forest

Cheolyong Park[1]

[1]Department of Statistics, Keimyung University

Received 12 February 2010, revised 11 March 2010, accepted 17 March 2010

Abstract

In this study, we propose two informal hypothesis tests which may be useful in determining the number of trees in a random forest for use in classification. The first test declares that a case is 'easy' if the hypothesis of the equality of probabilities of two most popular classes is rejected. The second test declares that a case is 'hard' if the hypothesis that the relative difference or the margin of victory between the probabilities of two most popular classes is greater than or equal to some small number, say 0.05, is rejected. We propose to continue generating trees until all (or all but a small fraction) of the training cases are declared easy or hard. The advantage of combining the second test along with the first test is that the number of trees required to stop becomes much smaller than the first test only, where all (or all but a small fraction) of the training cases should be declared easy.

Keywords: Hypotheses testing, random forest.

1. Introduction

Ensemble methods are techniques for improving classification accuracy by aggregating the predictions of multiple classifiers. Bagging (Breiman, 1996), boosting (Shapire *et al.*, 1998), and random forest (Breiman, 2001) are among the well-known ensemble methods. Bagging and boosting are two examples of ensemble methods that manipulate their training sets whereas random forest is an ensemble method that manipulates its input features.

When decision trees are used as their base classifiers, bagging, boosting, and random forest can be explained as follows. In bagging, each tree is independently constructed using a bootstrap sample of the data set. In boosting, successive trees give extra weight to points incorrectly predicted by earlier predictors and a weighted vote is taken for prediction in the end. Random forest adds an additional layer of randomness to bagging. In addition to constructing each tree using a different bootstrap sample of the data, each node is split using the best among a subset of input variables randomly chosen at that node. This somewhat counter-intuitive strategy turns out to perform quite well (Hamza and Larocque, 2005), and is robust against over-fitting (Breiman, 2001).

[1] Professor, Department of Statistics, Keimyung University, Daegu 704-701, Korea.
E-mail: cypark1@kmu.ac.kr

Random forest is considered particularly well suited to situations characterized by a large number of features. In such situations, classical classification approaches tend to become overwhelmed by the large number of features and fail. However random forest continues to work well. For example, Dudoit *et al.* (2002) and Lee *et al.* (2005) showed that random forest outperforms most of the other classification techniques in DNA microarray data. Random forest does not perform well when the percentage of truly informative features is small and Amaratunga *et al.* (2008) proposed so called enriched random forest that performs well even in that case.

In this study, we propose two informal hypothesis tests which may be useful in determining the number of trees in a random forest for use in classification. The first test, which is proposed by Park (2007), declares that a case is 'easy' if the hypothesis of the equality of probabilities of two most popular classes is rejected. In the easy case, we are pretty sure that the most popular class will be the same even if we continue generating much more trees. The second test declares that a case is 'hard' if the hypothesis that the relative difference or the margin of victory between the probabilities of two most popular classes is greater than or equal to some small number, say 0.05, is rejected. In the hard case, the probabilities of two most popular classes are so close that the case can hardly be declared easy even in much larger number of trees. We propose to continue generating trees until all (or all but a small fraction) of the training cases are declared easy or hard. The advantage of combining the second test along with the first test is that the number of trees required to stop becomes much smaller than the first test only, where all (or all but a small fraction) of the training cases should be declared easy.

This paper is organized as follows. In Section 2, we summarize the random forest algorithm and present two informal hypothesis tests with a sketch of proofs. In Section 3, some examples are provided where the number of trees required to stop by the two tests becomes much smaller than the first test only, and concluding remarks are given in Section 4.

2. Random forest algorithm and main results

Let $(x_1, y_1), \ldots, (x_m, y_m)$ be a training sample of m cases where x_i's are input variables of k -vectors and y_i's are the target variable of r classes. If we wish to construct a random forest to use in classifying a new case x, the random forest algorithm for generating n trees can be summarized as follows:

Step 1. Draw n bootstrap samples from the original data.
Step 2. For each of the bootstrap samples, grow an un-pruned classification tree, with the following modification. At each node, rather than choosing the best split among all input variables, randomly sample $l(l < k)$ of the input variables and choose the best split among the chosen l variables.
Step 3. Predict the new case x by aggregating the predictions of the trees.

Let M_n be the votes for the most popular class and let N_n be the votes for the second most popular class. If $M_n - N_n$ is large enough, we will declare that the case x is 'easy' because the most popular class will stay the same even if we generate more trees. On the other hand, if $M_n - N_n$ is small enough, we will declare that the case x is 'hard' because the most popular class will change easily even if we generate more trees. Let $S_n = M_n + N_n$,

then more precise forms of these informal tests are given as follows: we declare that the case x is 'easy' if

$$\frac{(M_n - N_n)}{\sqrt{S_n}} > c \tag{2.1}$$

for some large value of $c > 0$ and we say that the case x is 'hard' if

$$M_n - N_n \leq \epsilon S_n - d\sqrt{(1-\epsilon^2)S_n} \tag{2.2}$$

for some large value of $d > 0$ and some small value of $0 < \epsilon < 1$. We propose to stop generating more trees if the case x is easy or hard.

There are three tuning parameters c, d, ϵ in these tests. The parameters c, d are chosen such that the rejection regions (2.1) and (2.2) attain the significance level α. The parameters equal to z_α, the upper α -th quantile of the standard normal distribution, if M_n, N_n are the votes for two fixed classes in a multinomial distribution respectively. Our informal tests just set c, d to be large because M_n, N_n are not fixed classes. Formal discussions on this matter can be found in Alam (1991), Ramsey and Alam (1979), and Bhandari and Ali (1994) among others. The parameter ϵ is a fixed small positive number chosen such that if $|(p-q)/(p+q)| < \epsilon$, then we consider that the probabilities p, q of two classes are not easy to discriminate even in large trees.

We now extend our study from a single new case to each case in the training data. For each case in the training data, we need to calculate so called OOB (out-of bag) votes for r classes as follows:

Step 1. At each bootstrap iteration, predict the case not in the bootstrap sample (so called OOB data) using the tree grown with the bootstrap sample.

Step 2. Aggregate the OOB predictions for each case to get the OOB votes for the classes.

For each training case (x_i, y_i), let $M_n(i)$ be the OOB votes for the correct class y_i and let $N_n(i)$ be the OOB votes for the most popular wrong class. Here we define $M_n(i)$ and $N_n(i)$ a little differently from the earlier M_n and N_n. When we know the correct class y_i, it makes sense to use this in the definitions because we do not care much if the classification procedure cannot pick the correct class.

Define $S_n(i) = M_n(i) + N_n(i)$, then the informal tests in (2.1) and (2.2) become (2.3) and (2.4) respectively: we declare that the i -th case is easy if

$$\left| \frac{M_n(i) - N_n(i)}{\sqrt{S_n(i)}} \right| > c \tag{2.3}$$

for some large value of $c > 0$ and we say that the i -th case is hard if

$$|M_n(i) - N_n(i)| \leq \epsilon S_n(i) - d\sqrt{(1-\epsilon^2)S_n(i)} \tag{2.4}$$

for some large value of $d > 0$ and some small value of $0 < \epsilon < 1$. We propose to stop generating more trees if all (or all but a small fraction) of the training data are declared easy or hard. The first informal test (2.1) or (2.3) are proposed by Park (2001), and we

propose to add the second test (2.2) or (2.4) to the first test because the number of trees required to stop becomes much smaller than the first test only. Some examples will be given in Section 3.

Here we give some mathematical justification of the rules in (2.1) through (2.4). Let $(U_1, \ldots, U_r)$ be a multinomial random vector with sample size n^* and class probabilities $p_1, \ldots, p_r$. Note that $n^* = n$ for the rules in (2.1) and (2.2) and that $n^* \approx n/e$ for the rules in (2.3) and (2.4). Consider two fixed classes j and k. It is well known that

$$U_j|(U_j + U_k = s) \sim B\left(s, p_j/(p_j + p_k)\right)$$

from which it follows that

$$E(U_j - U_k|U_j + U_k = s) = s(p_j - p_k)/(p_j + p_k) = s\beta$$

and

$$Var(U_j - U_k|U_j + U_k = s) = s4p_jp_k/(p_j + p_k)^2 = s(1 - \beta^2)$$

where $\beta = (p_j - p_k)/(p_j + p_k)$. By the central limit theorem, we have

$$U_j - U_k|(U_j + U_k = s) \approx N\left(s\beta, s(1 - \beta^2)\right)$$

for large values of s, and

$$(U_j - U_k - s\beta)/\sqrt{s(1 - \beta^2)} \approx N(0, 1)$$

for large values of s.

Therefore, when $\beta = 0$ and s is large, $(U_j - U_k)/\sqrt{s} \approx N(0, 1)$ and the test with rejection region

$$|U_j - U_k|/\sqrt{s} > z_{\alpha/2} \tag{2.5}$$

is a test of $H_0 : \beta = 0$ with an approximate significance level α. This provides a rationale for the tests in (2.1) and (2.3).

If $|U_j - U_k|/\sqrt{s} > z_\alpha/2$, then it corresponds to an easy case and we stop generating more trees. If the test (2.5) is not rejected, we next go on to test the null hypothesis $H_0^{'} : |\beta| \geq \epsilon$, where ϵ is a fixed small positive number. If this test is rejected, it corresponds to a hard case and we stop generating more trees.

For the test of $H_0^{'} : |\beta| \geq \epsilon$, it is clear that the boundary cases $\beta = \pm\epsilon$ are the critical ones. We will show that the test with rejection region

$$|U_j - U_k| \leq s\epsilon - z_\alpha\sqrt{s(1 - \epsilon^2)} \tag{2.6}$$

is a test of $H_0^{'} : |\beta| \geq \epsilon$ with an approximate significance level α. Assume $\beta = \epsilon$ (we can handle the other case $\beta = -\epsilon$ exactly in the same way). From (2.5), when $\beta = \epsilon$ and s is large, we have

$$(U_j - U_k - s\epsilon)/\sqrt{s(1 - \epsilon^2)} \approx N(0, 1)$$

and

$$P\left(U_j - U_k \leq s\epsilon - z_\alpha\sqrt{s(1-\epsilon^2)}\right) = \alpha.$$

It is clear that

$$P\left(|U_j - U_k| \leq s\epsilon - z_\alpha\sqrt{s(1-\epsilon^2)}\right) \leq P\left(U_j - U_k \leq s\epsilon - z_\alpha\sqrt{s(1-\epsilon^2)}\right)$$

and two probabilities will be close when $\beta = \epsilon > 0$ and s is large because $|U_j - U_k|$ and $U_j - U_k$ are equal with high probability. This provides a rationale for the rules in (2.2) and (2.4).

3. Examples

We now consider two examples where the second test (2.4) is a useful addition to the first test (2.1). We will calculate the minimum numbers of trees required to stop for the rules with (2.3) only and with either (2.3) or (2.4) respectively, and show that adding (2.4) will reduce significantly the number of trees required to stop.

Example 1. Consider the case where there are only two classes and that, for a particular training case, the probabilities of the two classes are 0.505 and 0.495, respectively.

First we consider the case where we use (2.3) only. If we set $c = 3$, about 90,000 OOB votes are needed to achieve a probability 0.5 of satisfying (2.3). This means that we need to generate about $n = 244,645$ trees if all the cases in the training sample satisfy (2.3) with probability 0.5. This is because, with n trees, the number of OOB votes for any training case is approximately n/e. With $c = 2$, we need about 40,000 OOB votes and 108,731 trees.

Next, we consider the case where we use both (2.3) and (2.4). Let us take $\epsilon = 0.05$, $c = 3$, and $d = 2.782$. We take $d = 2.782$, so that it is comparable to $c = 3$ in the sense that $P(|Z| > c) = P(Z > d)$ where $Z \sim N(0,1)$. Elementary numerical calculations show that 5,245 OOB votes are necessary to satisfy (2.3) or (2.4) with probability 0.5. This is much less than the 90,000 OOB votes. If we take $c = 2$ and $d = 1.69$ with the same $\epsilon = 0.05$, we need about 2,155 OOB votes to satisfy (2.3) or (2.4) with probability 0.5.

Example 2. In this example, we consider the case where there are 15 classes and the probability of choosing the correct class is small. Suppose that, for a particular training case, the two most popular classes have probabilities 0.1 and 0.095. We also assume the other 13 classes have probabilities less than 0.07, so that they are well out of running for the first place. The absolute difference in the probabilities is 0.005, which is smaller than Example 1, but the relative difference or the margin of victory β is $0.005/0.195 = 0.026$, which is larger than Example 1.

For the case where we use (2.3) only, if we set $c = 3$, about 70,200 OOB votes and 190,823 trees are needed. With $c = 2$ we need about 31,200 OOB votes and 84,810 trees.

We now consider the case where we use both (2.3) and (2.4). Let us take $\epsilon = 0.05$, $c = 3$, and $d = 2.782$, then about 40,234 OOB votes are needed to satisfy (2.3) or (2.4) with probability 0.5. This is a modest improvement on the earlier value of 70,200. With $c = 2$ and $d = 1.69$ and the same $\epsilon = 0.05$, we need about 11,882 OOB votes, another modest improvement compared with the earlier 31,200.

From Examples 1 and 2, we note that the improvement is dependent more on the margin of victory β than on the absolute difference. Indeed we note that the improvement is smaller when β is somewhat bigger.

4. Concluding remarks

In this study, we proposed two informal hypothesis tests which may be useful in determining the number of trees in a random forest for use in classification. The first test declares that a case is easy if the hypothesis of the equality of probabilities of two most popular classes is rejected. In the easy case, we are pretty sure that the most popular class will be the same even if we continue generating much more trees. The second test declares that a case is hard if the hypothesis that the relative difference or the margin of victory between the probabilities of two most popular classes is greater than or equal to some small number, say 0.05, is rejected. In the hard case, the probabilities of two most popular classes are so close that the case can hardly be declared easy even in much larger number of trees. We propose to continue generating trees until all (or all but a small fraction) of the training cases are declared easy or hard. The advantage of combining the second test along with the first test is that the number of trees required to stop becomes much smaller than the first test only.

This advantage is illustrated with two examples. These examples show that the improvement, in the numbers of trees needed to stop, is dependent more on the margin of victory β than on the absolute difference. Indeed we has found that the improvement is smaller when β is somewhat bigger.

The rules in (2.3) and (2.4) say that we stop generating more trees if we can reject the null hypothesis $H_0 : \beta = 0$ or if we can reject the null hypothesis $H_0^{'} : |\beta| \geq \epsilon$ for a fixed small positive number ϵ. The procedures in (2.3) and (2.4) must be regarded as merely informal. The procedures in (2.5) and (2.6) are valid for comparing two fixed class j and k, but in (2.1) and (2.2) we are comparing two random classes, and in (2.3) and (2.4) we are comparing a fixed class with a random class. This raises the standard issues of multiple comparisons which become especially important when the number of classes is large.

References

Alam, K. (1971). On selecting the most probable category. *Technometrics,* **13**, 843-850.

Amaratunga, D., Cabrera, J. and Lee, Y. S. (2008). Enriched random forests. *Bioinformatics*, **24**, 2010-2014.

Bhandari, S. K. and Ali, M. M. (1994). An asymptotically minimax procedure for selecting the t -best multinomial cells. *Journal of Statistical Planning & Inference,* **38**, 65-74.

Breiman, L. (1996). Bagging predictors. *Machine Learning,* **24**, 123-140.

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

Dudoit, S., Fridlyand, J. and Speed, T. P. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Society*, **97**, 77-87.

Hamza, M. and Larocque, D. (2005). An empirical comparison of ensemble methods based on classification trees. *Journal of Statistical Computation & Simulation,* **75**, 629-643.

Lee, J. W., Lee, J. B., Park, M. and Song, S. H. (2005). An extensive evaluation of recent classification tools applied to microarray data. *Computational Statistics & Data Analysis*, **48**, 869-885.

Park, C. (2007). A stopping rule for the number of generating trees in a random forest. *Journal of the Institute of Natural Sciences*, **27**, 7-10.

Ramey, J. T. and Alam, K. (1979). A sequential procedure for selecting the most probable multinomial event. *Biometrika,* **55**, 171-173.

Shapire, R., Freund, Y., Bartlett, P. and Lee, W. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics,* **26**, 1651-1686.