Hindawi
Security and Communication Networks
Volume 2019, Article ID 3984031, 12 pages
https://doi.org/10.1155/2019/3984031

Research Article

Dynamic Nonparametric Random Forest Using Covariance

Seok-Hwan Choi, Jin-Myeong Shin, and Yoon-Ho Choi

School of Computer Science and Engineering, Pusan National University, Busan, 26241, Republic of Korea

Correspondence should be addressed to Yoon-Ho Choi; yhchoi@pusan.ac.kr

Received 2 November 2018; Accepted 6 March 2019; Published 27 March 2019

Academic Editor: Mamoun Alazab

As the representative ensemble machine learning method, the Random Forest (RF) algorithm has widely been used in diverse applications on behalf of the fast learning speed and the high classification accuracy. Research on RF can be classified into two categories: (1) improving the classification accuracy and (2) decreasing the number of trees in a forest. However, most of papers related to the performance improvement of RF have focused on improving the classification accuracy. Only some papers have focused on reducing the number of trees in a forest. In this paper, we propose a new Covariance-Based Dynamic RF algorithm, called C-DRF. Compared to the previous works, while ensuring the good-enough classification accuracy, the proposed C-DRF algorithm reduces the number of trees. Specifically, by computing the covariance between the number of trees in a forest and F-measure at each iteration, the proposed algorithm determines whether to increase the number of trees composing a forest. To evaluate the performance of the proposed C-DRF algorithm, we compared the learning time, the test time, and the memory usage with the original RF algorithm under the different areas of datasets. Under the same or higher classification accuracy, it is shown that the proposed C-DRF algorithm improves the performance of the original RF algorithm by as much as 58.68% at learning time, 47.91% at test time, and 68.06% in memory usage on average. As a practical application area, we also show that the proposed C-DRF algorithm is more efficient than the state-of-the-art RF algorithms in Network Intrusion Detection (NID) area.

1. Introduction

As one of the classification modeling approaches, decision tree learning has widely been used in various learning fields such as statistics, data mining, and machine learning. When classifying the optimal value of an output variable based on input variables, decision tree learning uses a single decision tree or multiple decision trees. By using the Hill Climbing method [1–3] and the Greedy method [4, 5], the single decision tree method has obtained optimal solutions. However, since the classified output value is sensitive to the size of the learning sample, the single decision tree method can be applied to find only a local optimum.

To solve such a limitation, there has been a lot of interest in an ensemble machine learning method, which generates multiple classifiers and aggregates their results, since the late 1990s [6–8]. The ensemble machine learning method has been designed by using various methods such as bagging, boosting, and stacking [9–11]. As a representative ensemble machine learning method, the Random Forest (RF) algorithm was proposed by Breiman [12]. The RF algorithm grows many decision trees for classification and regression analysis [13]. RF grows by combining randomized node optimization (RNO) and bootstrap aggregating (bagging). Each tree is independently constructed using a bootstrap sample of the dataset and added into a forest.

Based on Breiman's paper [12], let us overview the overall operation of the RF algorithm. To classify the output value using multiple decision trees, the RF algorithm consists of two operational phases: (1) training and (2) test. In the training phase, a randomly sampled dataset, called the inBag dataset, from the training dataset is selected. Note that the remaining training dataset different from the inBag dataset is called the out-of-bag (OoB) dataset. By using the inBag dataset, a decision tree is grown. This process is repeated to generate $k(\geq 1)$ number of decision trees and, then, an RF is grown. When growing an RF, the OoB dataset can be used to evaluate the classification accuracy of a decision tree. In the test phase,

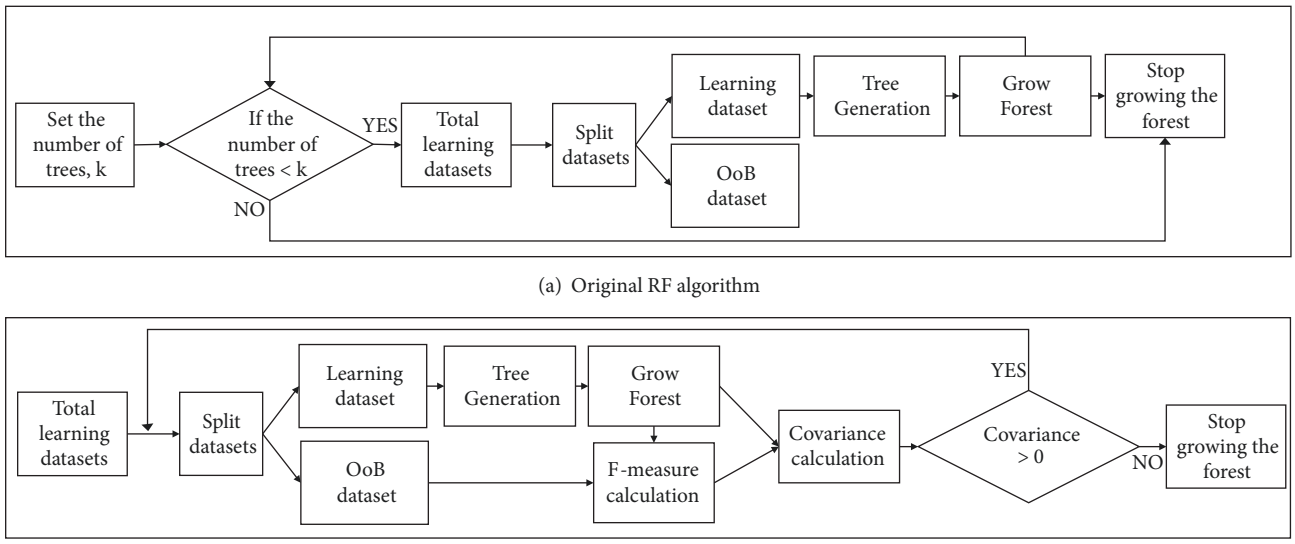

Figure 1: Overall learning operation of the original RF algorithm and the proposed C-DRF algorithm.

the RF algorithm classifies the input data into an output value by voting the result of all decision trees.

On behalf of the fast learning speed and the high classification accuracy, the RF algorithm has been widely used in diverse applications such as image recognition and information security [20–22]. For example, as a practical application, the RF algorithm has frequently been used as the core engine of intrusion detection system because of the fast learning speed and the high detection accuracy [23, 24]. Also, because the classification accuracy and learning speed of RF can mainly vary according to the operational details in the learning phase, multiple variances have been proposed. To improve the classification accuracy of RF, Cutler et al. proposed an algorithm which modified how to create a decision tree [14]. Rodriguez et al. proposed the RF induction algorithm based on the readjustment of the learning data [15]. Also, in [16, 17], new algorithms that assign weight to each tree were proposed. Even though these algorithms focused on improving the classification accuracy of RF, the classification accuracy can be decreased according to the number of trees, denoted into k, composing the forest.

To find the optimal value of k, Cuzzocrea et al. proposed an algorithm that gradually increased the number of trees [18] and found the best classifiers than the others. Even though Cuzzocrea et al.'s algorithm minimizes the number of trees composing the forest, it requires much learning time and much memory usage. P. Latinne et al. also proposed an algorithm where the forest was grown based on a direct nonparametric test of comparison, that is, the McNemar test [19, 25]. By determining the minimum number of weakened classifiers, P. Latinne et al.'s algorithm reduced the number of trees composing the forest. It was shown that a gain for time and memory could be obtained.

In this paper, we propose a new Covariance-Based Dynamic RF algorithm, called C-DRF. While keeping the best classification accuracy close to that of the original RF algorithm [12], the proposed algorithm reduces the number of trees composing the forest. Compared to Cuzzocrea et al.'s algorithm [18], the proposed algorithm does not need to know the predefined number of trees k and a certain a priori information. Also, compared to P. Latinne et al.'s algorithm [19], the proposed algorithm uses the covariance between the number of trees and F-measure for a forest at each iteration. Here, F-measure is a measure of the test accuracy and the covariance is a measure of the joint variability of two random variables or vectors.

As shown in Figure 1(a), the original RF algorithm iteratively splits the learning dataset into inBag and OoB datasets and, then, generates k trees using inBag dataset. On the other hand, the proposed algorithm determines whether to increase the number of trees based on a direct nonparametric test of comparison dynamically. That is, as shown in Figure 1(b), according to the sign of the computed covariance between the number of trees in a forest and F-measure for the OoB dataset at each iteration, the proposed algorithm determines whether to increase the number of trees composing a forest.

The main contributions of this paper can be summarized as follows. (1) To the best of our knowledge, we propose the first RF learning algorithm that generates the minimum number of trees using covariance, while keeping the best classification accuracy close to the original RF algorithm. (2) By analyzing the computational time complexity of the proposed C-DRF algorithm, we show that C-DRF can reduce the computational time complexity of the original RF algorithm in practice. (3) We evaluate the proposed C-DRF algorithm with datasets collected from diverse applications. From the experimental results, we show that the proposed algorithm reduces the number of trees in a forest while keeping the best accuracy close to the original RF algorithm

[12, 19]. Also, we show that the proposed C-DRF algorithm reduces the learning time, the memory usage, and the test time compared to the other RF algorithms under various applications such as network intrusion detection.

The rest of the paper is organized as follows. In Section 2, we overview research works related to the RF algorithm. In Section 3, after showing the overall operation of the proposed algorithm, we describe the operation in detail. We also show the computational complexity for learning in Section 4. After showing the experimental results for evaluating the performance of C-DRF in Section 5, we discuss the limitation of C-DRF in Section 6. Finally, we summarize this paper in Section 7.

Require: $isBreak = FALSE$
Require: $i = 0$
(1) **procedure** C-DRF(D)
(2) **while** $isBreak == FALSE$ **do**
(3) $i = i + 1$;
(4) $Forest_i$ = SubForestGeneration(D, i);
(5) C_i = CovCalculation($Forest_i$);
(6) $isBreak$ = LearnTermination(C_i)
(7) **end while**
(8) **end procedure**

Algorithm 1: Overall Operation.

2. Related Works

In this section, we overview the characteristics of the RF algorithms according to their goals: (1) improving the classification accuracy and (2) optimizing the number of trees. We summarize the characteristics of the various RF algorithms in Table 1.

To improve the classification accuracy of the original RF algorithm, Cutler et al. proposed a new tree creation algorithm that sets the cut point between randomly selected two learning instances, called PERT [14]. Even though the PERT algorithm has shortened learning time and improved the classification accuracy, the performance of the PERT algorithm can be degraded for certain datasets such as DNA [26]. In Rodriquez et al.'s paper [15], the Rotation Forest algorithm was proposed. The Rotation Forest algorithm grows a forest after rotating the data axis while applying principal component analysis (PCA) to the learning data. Even though the Rotation Forest algorithm improved the classification accuracy by constructing an oblique decision boundary, the classification accuracy for dataset with unclear direction of dispersion is strictly degraded. Robnik also proposed an algorithm that assigns weight to each tree composing the forest [16]. Robnik's algorithm showed higher classification accuracy than the original RF algorithm. However, Robnik's algorithm has a limitation that the classification accuracy varies depending on a well-formed tree and the accuracy variation is very large. In [17], Bernard et al. proposed a dynamic RF algorithm that dynamically derived the forest while adding the most appropriate tree to the already configured forest. The weight values are assigned to the training instance based on the OoB error rate. Based on the weight values, a new tree is generated using randomly selected training data. Bernard et al.'s algorithm showed the good classification accuracy in the pattern recognition area even though a specific dataset contains a lot of noise. However, Bernard et al.'s algorithm has a disadvantage that the tree generation time increases when reflecting the weight value of the learning instance. Also, the greater the number of trees composing the forest is, the higher the possibility of overfitting to the learning data.

To optimize the number of trees while keeping the classification accuracy close to or higher than that of the original RF algorithm, Cuzzocrea et al. proposed a new algorithm [18]. Based on the relationship between the predictive power which means the percentage of positively classified cases of instances of that dataset and the number of trees in a forest, they proposed how to optimize the number of trees in RF using an information-theoretic approach. However, the performance of Cuzzocrea et al.'s method was limited because, while estimating the predictive power for the large dataset, a large amount of memory and much learning time were required. As an alternative to reduce memory usage and learning time, P. Latinne et al. proposed a new algorithm that limits the number of trees by using the McNemar test [19]. P. Latinne et al.'s algorithm [19] grows a forest with fewer number of trees than the maximum number of trees of the original RF algorithm. However, since the maximum number of trees could not guarantee an optimal result, the performance of P. Latinne et al.'s algorithm varies depending on the predefined maximum number of trees.

To overcome the performance degradation of the previous RF algorithms due to the dependency on learning data and the number of trees, we propose a new learning algorithm that dynamically derives the optimal number of trees by using the covariance between the number of trees in a forest and F-measure for a forest from each iteration.

3. C-DRF Algorithm

In this section, we overview the operation of the proposed C-DRF algorithm in detail.

3.1. Overall Operation. To determine whether to increase the number of trees based on a direct nonparametric test of comparison dynamically, the proposed C-DRF algorithm consists of three functional modules: (1) subforest generation; (2) covariance calculation; and (3) learning termination. We show the overall operation of C-DRF in Algorithm 1. In Table 2, we also summarize the terms and notation used in this paper.

3.1.1. Subforest Generation. By using D_L for each iteration, a decision tree is generated and, then, included in a forest. Here, we call an intermediate forest before learning termination into "subforest" because the complete forest satisfying the termination condition is not driven.

Table 1: Characteristics of various RF algorithms.

	Goal	Learning Characteristics	Drawback
PERT [14]	Accuracy Improvement	To set the cut point between randomly selected two learning instances	Accuracy varies depending on learning data
Rodriguez et al.'s algorithm [15]	Accuracy Improvement	To find key attributes using PCA before learning	Accuracy varies depending on learning data
Robnik-Sikonja's algorithm [16]	Accuracy Improvement	To get the target value using weighted voting	Accuracy varies depending on a well-formed tree
Bernard et al.'s algorithm [17]	Accuracy Improvement	To assign weights to learning instances	Much calculation time and the possibility of overfitting
A. Cuzzocrea et al.'s algorithm [18]	Optimal Number of Trees	To select the number of trees through many tests	Waste of memory and high computation costs
P. Latinne et al.'s algorithm [19]	Optimal Number of Trees	To use k as an input parameter	The number of trees varies depending on k

Table 2: Terms and Notation.

Terms	Notation
D	Total dataset for training
D_L	Learning dataset consisting of p% of random samples from D
D_{OoB}	OoB dataset, i.e., $D_{OoB} = D - D_L$
$Tree_i$	A decision tree generated by using D_L at the i^{th} iteration
$Forest_i$	A forest consisting of trees generated by the i^{th} iteration
x_L	A single data instance in D_L
x_{OoB}	A single data instance in D_{OoB}
I_b	Information-theoretic criteria for finding best attributes
A_b	Best attributes based on I_b
D_b	A subset of D_L induced by A_b
x_b	A single data instance in D_L
$Node_b$	A decision node for testing A_b
N	A row vector consisting of numbers of decision trees by the i^{th} iteration, i.e., $N = [1, \ldots, i]$
N_{max}	Maximum number of trees composing a forest
TP	Number of true positives for D_{OoB} from each iteration
FP	Number of false positives for D_{OoB} from each iteration
FN	Number of false negatives for D_{OoB} from each iteration
R_{Cp}	Precision for D_{OoB} from each iteration
R_{Cr}	Recall for D_{OoB} from each iteration
F_i	F-measure for D_{OoB} at the i^{th} iteration
F	A row vector consisting of F_js by the i^{th} iteration, i.e., $F = [F_1, \ldots, F_i]$.
C_i	Covariance between N and F at the i^{th} iteration

```
(1) procedure SubForestGeneration(D, i)
(2)     D_L = RandomSelection(D);
(3)     D_OoB = D - D_L;
(4)     Tree_i = CreateTree(D_L);
(5)     Forest_i = Forest_i + Tree_i;
(6)     return Forest_i;
(7) end procedure
```

Algorithm 2: Sub-Forest Generation.

3.1.2. Covariance Calculation. For each iteration, C-DRF calculates the F-measure value of the current subforest for D_{OoB}. Given the F-measure value, covariance between the number of trees and the F-measure value for the current subforest is computed.

3.1.3. Learning Termination. By inspecting C_i, C-DRF determines whether to generate an additional tree for RF or not.

3.2. Generation of Subforest. To generate a new subforest, a new decision tree is generated from D_L and added to the existing subforest.

As shown in Algorithm 2, subforest generation works as follows. After selecting an inBag dataset D_L for learning from D (Line (2)), the subforest generation module determines an OoB dataset D_{OoB} for evaluating the covariance (Line (3)). For the given D_L, a new decision tree $Tree_i$ is generated and, then, the existing forest is combined with $Tree_i$ (Line (4) to (5)). Finally, the subforest generation module returns the updated forest (Line (6)).

```
(1) procedure CreateTree(D_L)
(2)     for all x_L ∈ D_L do
(3)         I_b = ComputeCriteria(x_L);
(4)     end for
(5)     A_b = CalBestAttribute(I_b);
(6)     Tree_i = CreateDecisionNode(A_b);
(7)     D_b = SplitData(D_L, A_b);
(8)     for all x_b ∈ D_b do
(9)         Node_b = CreateTree(x_b);
(10)        Tree_i = Tree_i + Node_b;
(11)    end for
(12)    return Tree_i;
(13) end procedure
```

Algorithm 3: Creation of a Decision Tree Using C4.5.

As the function CreateTree (D_L) in Algorithm 2, the C4.5 algorithm [27] in Algorithm 3 is used. The C4.5 algorithm works as follows. The information gains for all attributes in D are computed (Lines (2) to (4)). After the best attribute with the highest information gain is chosen (Line (5)), a decision

```
(1) procedure CovCalculation(Forest_i)
(2)   if i == 1 then
(3)     for all x_OoB ∈ D_OoB do
(4)         Count either TP, FP or FN from the return
(5)         value of Test4Classification(x_OoB, Tree_i);
(6)     end for
(7)     R_Cp = TP/(TP+FP);
(8)     R_Cr = TP/(TP+FN);
(9)     F_i = F_Measure(R_Cp, R_Cr);
(10)    F ⟵ F_i;
(11)  else
(12)    for all x_OoB ∈ D_OoB do
(13)      for all 1 ≤ j ≤ i do
(14)        Count either TP, FP or FN from the return
(15)        value of Test4Classification(x_OoB, Tree_j);
(16)      end for
(17)    end for
(18)    R_Cp = TP/(TP+FP);
(19)    R_Cr = TP/(TP+FN);
(20)    F_i = F_Measure(R_Cp, R_Cr);
(21)    F ⟵ F_i;
(22)    N ⟵ i;
(23)    C_i = cov(N, F);
(24)  end if
(25)  return C_i
(26) end procedure
```

Algorithm 4: Covariance Calculation.

node based on the best attribute is included into $Tree_i$ (Line (6)). Based on the best attribute, C4.5 splits D_L and, thus, generates D_b. From every x_b, a decision tree is generated to get a subtree (Lines (9) to (11)). Finally, C4.5 returns a new decision tree $Tree_i$ (Line (12)).

3.3. Covariance Calculation. In this section, we describe how to compute the covariance between the number of trees N_i and a set of Fl-measures F at the i^{th} iteration in detail. As shown in Algorithm 4, the covariance can be computed into two cases: (1) for the 1*st* iteration and (2) for the other iterations.

After a decision tree $Tree_1$ is generated at the 1*st* iteration, each x_{OoB} is tested with $Tree_1$ and, thus, TP, FP, and FN are computed (Lines (3) to (6)). By using TP, FP, and FN, the precision R_{Cp} and the recall R_{Cr} for $Forest_1$ are computed (Lines (7) and (8)). Since the precision and the recall are commonly used to evaluate machine learning algorithms, we compute F_1, which composes F, with the parameters R_{Cp} and R_{Cr} (Lines (9) and (10)), and continues to grow the forest (Line (11)). In the following equation, we show how to compute F-measure (Line (9)):

$$F\_Measure\left(R_{Cp}, R_{Cr}\right) = \frac{1}{\alpha\left(1/R_{Cp}\right) + (1-\alpha)\left(1/R_{Cr}\right)}, \tag{1}$$

where α and $1-\alpha$ are the relative weights for precision and recall, respectively. Here, when α is 0.5, F-measure is especially called $F1$-measure, which is the harmonic mean of precision and recall. Since $F1$-measure is the most frequently used for evaluating the accuracy of machine learning algorithms, we set α into 0.5 in the proposed C-DRF algorithm. Thus, F-measure indicates $F1$-measure in this paper.

Different from the 1*st* iteration, $Forest_i$ consists of multiple numbers of tress. Thus, each x_{OoB} is tested with i number of decision trees and, then, TP, FP, and FN are computed from every $Tree_j$ (Lines (13) to (18)). By using TP, FP, and FN, the precision R_{Cp} and the recall R_{Cr} for $Forest_i$ are computed (Lines (19) and (20)). From Equation (1), F_i, which composes F, is computed with the parameters R_{Cp} and R_{Cr} (Lines (21) and (22)). After including i into N (Line (23)), the covariance C_i is computed by using N and F (Line (24)). To understand why the covariance is used to determine whether to terminate growing the forest or not, let us consider the covariance between two random vectors X and Y as follows:

$$\mathrm{cov}\,(X,Y) = \frac{1}{N'}\sum_{j=1}^{N'}\left(x_j - \overline{x}\right)\left(y_j - \overline{y}\right), \tag{2}$$

where N' is the number of elements composing a random vector and $\overline{x}$ and $\overline{y}$ are the average values of the random vectors X and Y, respectively. Note that if the covariance is larger than zero, it implies that Y increases as the random variable X increases. If the covariance is less than zero, Y decreases as X increases. Also, if the covariance is zero, it implies that the two random vectors are independent from each other [28]. In this regard, many studies have

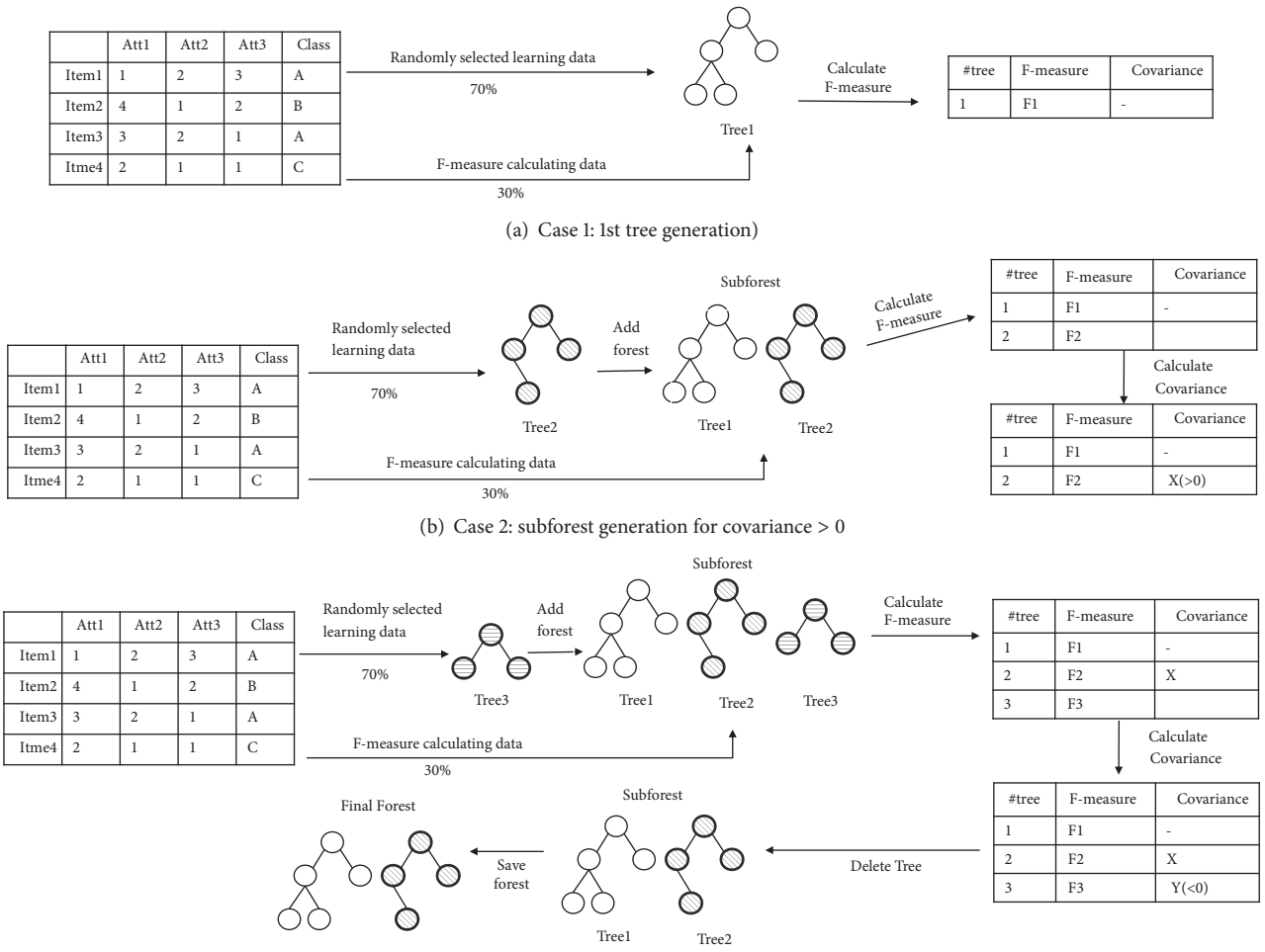

Figure 2: Operational example of the proposed C-DRF algorithm.

already proven that these characteristics of the covariance are effective for data analysis [29, 30]. From these characteristics of the covariance, we use the covariance to determine whether to terminate growing the forest or not.

The proposed C-DRF algorithm defines two row vectors N and F as follows:

$$N = [1, 2, 3, \ldots, i-1, i]$$
$$\text{and } F = \left[F_1, F_2, \ldots, F_{i-1}, F_i\right], \tag{3}$$

where $1 \leq i \leq N_{max}$.

By following the characteristics of the covariance, if the covariance between N and F is positive, the F-measure trends to increase as the forest grows at the i^{th} iteration. It implies that the forest can grow with a new decision tree. However, if the covariance is negative or equal to zero, the F-measure trends to decrease as the forest grows at the i^{th} iteration. Since the classification accuracy trends to decrease after adding a new tree, it implies that the forest should not grow further.

3.4. Learning Termination. Note that after the covariance has been computed, the learning termination module decides whether to grow the forest or not. As shown in Algorithm 5, if C_i is less than or equal to zero, the forest does not grow with a decision tree generated at the i^{th} iteration (Lines (2) to (6)). That is, the learning phase stops growing the forest and, thus, the variable *isBreak* at Algorithm 1 is set into *TRUE*. Otherwise, the forest continues to grow. Also, in order to generate a new decision tree from the $i + 1_{th}$ iteration, the learning termination modules return the value *FALSE* (Line (7)).

3.5. Example. In Figure 2, we show an example of how the proposed C-DRF algorithm dynamically generates a subforest. Let us assume that D consists of four numbers of x_L, that is, $\{x_{L1}, x_{L2}, x_{L3}, x_{L4}\}$. Here, each x_L consists of four numbers of attributes. We also assume that p is set into 70. Thus, D_L consists of the randomly selected 70% datasets in D and, then, D_{OoB} consists of the remaining 30% datasets in D.

As shown in Figure 2(a) for the 1st iteration, the proposed C-DRF algorithm generates the 1st decision tree $Tree_1$ for

```
(1) procedure LearnTermination(C_i)
(2)   if C_i ≤ 0 then
(3)     Forest_i = Forest_i - Tree_i;
(4)     SaveForest(Forest_i);
(5)     return TRUE
(6)   end if
(7)   return FALSE
(8) end procedure
```

Algorithm 5: Learning Termination.

a new random sample dataset D_L. At the 1st iteration for a decision tree, F_1 is set into $F1$ but the covariance is not computed because there is no comparative F_i. Here, a subforest consists of only $Tree_1$. After the 1st tree is generated, the C-DRF algorithm iteratively selects D_L to generate the i^{th} decision tree.

In Figure 2(b), we show how to generate the 2nd decision tree and add it to a subforest. As the 1st decision tree is generated, the 2nd decision tree, $Tree_2$, is generated for a new random sample dataset D_L and is ensembled with $Tree_1$. After F_2 is updated with the $F1$-measure for a subforest consisting of $Tree_1$ and $Tree_2$, F_1 and F_2 are set into $F1$ and $F2$, respectively. Next, from the given F_i, the covariance X between the number of trees, that is, 2, and F_i is computed. Here, if X is larger than zero, a subforest consists of an ensemble of $Tree_1$ and $Tree_2$. However, if the covariance Y for the 3rd iteration is less than zero as shown in Figure 2(c), the proposed C-DRF algorithm does not grow a subforest. After removing a new decision tree $Tree_3$ from subforest, the previous subforest is set into a final forest.

4. Complexity Analysis

Note that while keeping the same classification accuracy as the original RF algorithm, the proposed C-DRF algorithm shows faster learning time than the original RF algorithm. In this section, we analyze the time complexity of the proposed C-DRF algorithm and, then, compare it with the original RF algorithm.

From Louppe et al.'s paper [34], the time complexity for learning of the original RF algorithm is given into

$$\Theta\left(MN_L^2 \log^2 N_L\right), \tag{4}$$

where M is the number of trees composing the forest and N_L is the number of data in D_L. Compared to the original RF algorithm, the proposed C-DRF algorithm generates m number of trees while computing covariance of each subforest using p% dataset of D. Thus, the time complexity of the proposed C-DRF algorithm is expressed as follows:

$$\Theta\left(mN_L^2 \log^2 N_L + \frac{m(m+1)}{2} \log\left(\frac{N_L(1-p)}{(p)}\right)\right), \tag{5}$$

where $mN_L^2 \log^2 N_L$ represents the time complexity for constructing m numbers of trees. The time complexity for computing covariance of each subforest is given as the multiplication of the ratio of D_{OoB} to D_L (: $(1-p)/(p)$) and the time complexity for covariance computation while a forest consists of m numbers of trees (: $m(m+1)/2$). If m is less than M and N_L is large enough, the proposed C-DRF algorithm is faster than the original RF algorithm because the ratio of Equation (4) over Equation (5) approximates as follows:

$$\lim_{N_L \longrightarrow \infty} \frac{MN_L^2 \log^2 N_L}{mN_L^2 \log^2 N_L + (m(m+1)/2) \log N_L (1-p)/p} \approx \frac{M}{m}. \tag{6}$$

If M is equal to m, the proposed C-DRF algorithm is slower than the original RF algorithm by as much as $\Theta((m(m+1)/2)\log(N(1-p)/(p)))$, which is the additional time complexity caused by the covariance calculation. However, if N_L is larger than or equal to 2500, which is achieved by the numerical analysis, the proposed C-DRF algorithm shows the same time complexity as the original RF algorithm because M/m approximates to 0.99. Note that N_L is larger than 2500 in general. Since most learning data consist of more than 2500 number of data, the proposed C-DRF algorithm takes less learning time than the original RF algorithm in practice.

5. Experimental Evaluation

In this section, we show the experimental results of the proposed C-DRF algorithm by comparing with the previous RF algorithms. To evaluate the performance of the proposed C-DRF algorithm, we compared accuracy, memory usage, learning time, and test time of the proposed C-DRF algorithm with the original RF algorithm and P. Latinne e al.'s algorithm [19] by using different areas of datasets. Since P. Latinne et al.'s algorithm is the best algorithm to reduce the number of trees while maintaining the same classification accuracy, we used it as a comparison algorithm for the performance evaluation of the proposed C-DRF algorithm. Also, since the 100 trees showed the best result in our experiment for the original RF algorithm, the number of trees composing the forest of the original RF algorithm is fixed to 100.

5.1. Experimental Environment. We evaluated the performance of the proposed C-DRF algorithm by using three areas of datasets. The first dataset is the MNIST dataset [31], which is commonly used to verify the performance of machine learning algorithms related to image processing. The second dataset is the data for the building property, which is used to evaluate the value of the building. This real estate dataset is generated by preprocessing some real estate information at Korea MOLIT [32]. As the third dataset, we used the UNSW-NB15 dataset [33], which complements the probability distribution problem of the KDDCup99 [35] dataset. The UNSW-NB15 dataset includes the recently reported attack traces and, thus, is used primarily to verify the performance of machine learning algorithms in the field of Network Intrusion Detection. The details of the datasets are

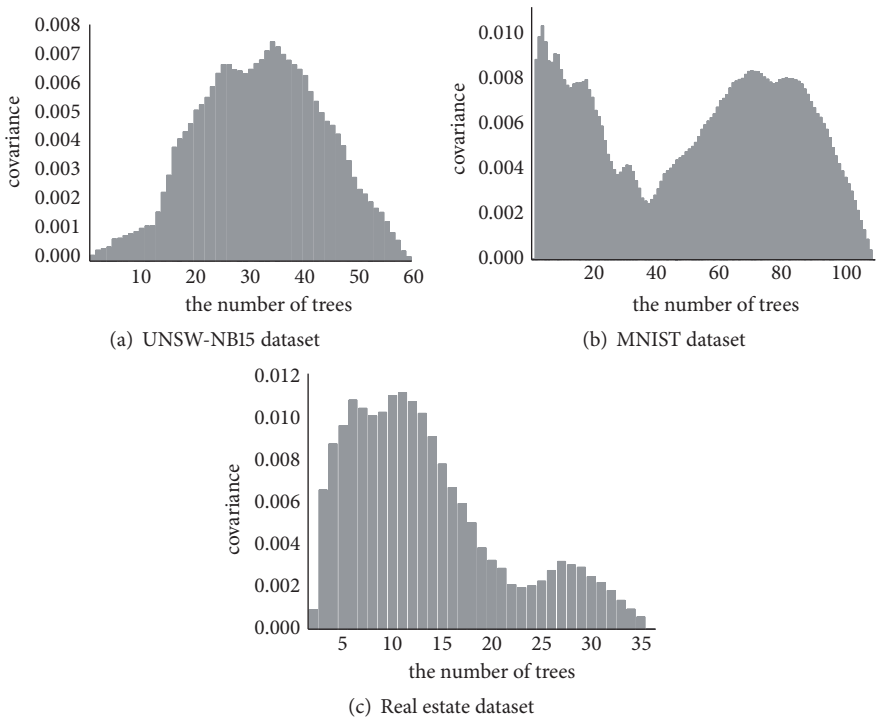

Figure 3: Covariance change for different numbers of trees for various datasets.

Table 3: Experimental Datasets.

Dataset	#Training data	#Feature	#Class
MNIST [31]	60000	784	10
real estate [32]	41700	29	9
UNSW-NB15 [33]	97278	42	10

described in Table 3. Also, we measured the performance of the proposed C-DRF algorithms on the Ubuntu 14.04.5 LTS machine with kernel version 4.2.0-27-generic, 2.40GHz CPU clock(Intel Xeon CPU E5-2630 v3), and 32GB memory.

5.1.1. Comparison of Classification Accuracy and Number of Trees. In Figure 3, we show how the number of trees for each dataset was determined based on the covariance in the proposed C-DRF algorithm. For the UNSW-NB15 dataset, as shown in Figure 3(a), the covariance is less than zero after the 61_{th} decision tree is generated. Thus, the forest consists of 60 trees for the UNSW-NB15 dataset. For the real estate dataset, the covariance is less than zero after the 37_{th} decision tree is generated in Figure 3(c). Thus, the forest consists of 36 trees for the real estate dataset. In Table 4, we summarize the number of trees for each dataset.

In Table 4, we also show the accuracy of each algorithm with the different number of trees under different datasets. For the UNSW-NB15 dataset, while the forest consists of 60 trees in the proposed C-DRF algorithm, the forest consists of 100 trees and 70 trees in the original RF algorithm and P. Latinne et al.'s algorithm, respectively. From Table 4, we can observe that, even though the proposed C-DRF algorithm composes the forest with the smallest number of trees, the proposed C-DRF algorithm shows the classification accuracy of 76.322% close to 76.314% of the original RF algorithm and 76.293% of P. Latinne et al.'s algorithm.

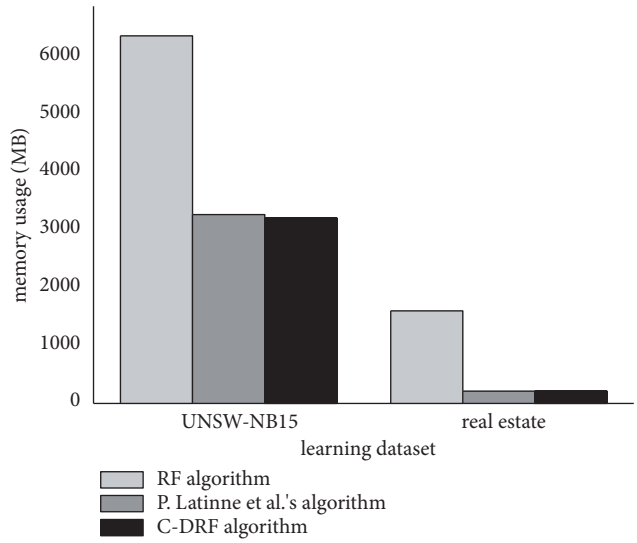

Figure 4: Comparison of memory usage under different datasets.

5.1.2. Memory Usage. In Figure 4, we show compared memory usage of each algorithm under different datasets. For the UNSW-NB15 dataset, the proposed C-DRF algorithm used the memory space of 3,212.65MB, which is smaller than

Table 4: Experimental Results.

	RF algorithm	C-DRF algorithm	P. Latinne et al.'s algorithm [19]
# of Trees (UNSW-NB15)	100	60	70
Accuracy (UNSW-NB15)	76.314%	76.322%	76.293%
# of Trees (real estate dataset)	100	36	30
Accuracy (real estate dataset)	90.72%	90.73%	90.71%

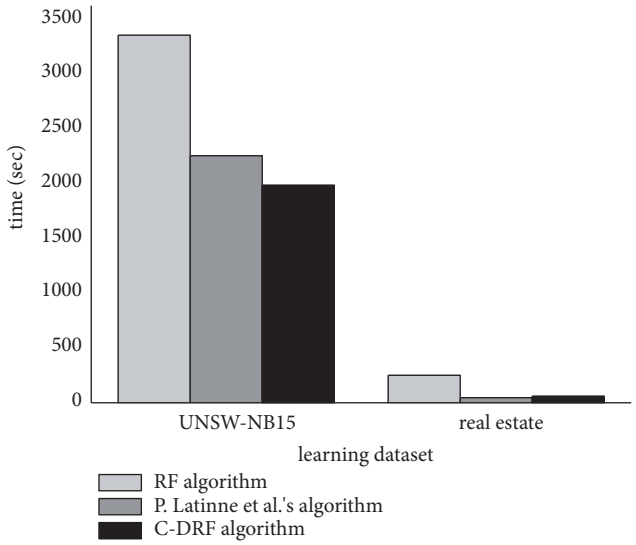

Figure 5: Comparison of learning time under different datasets.

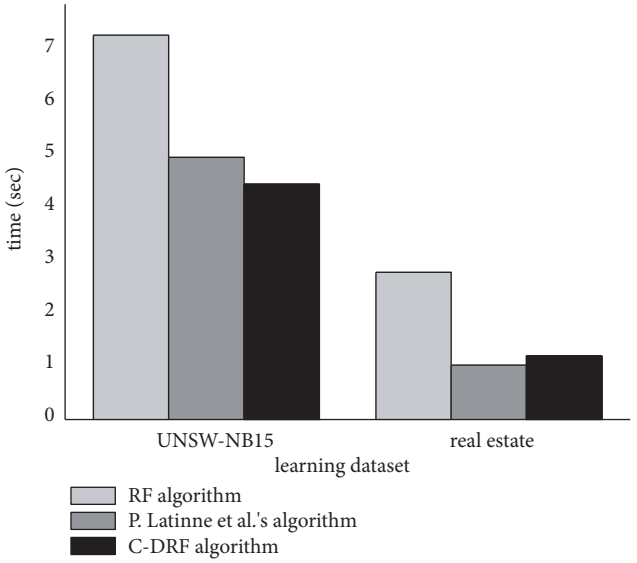

Figure 6: Comparison of test time under different datasets.

6,368.08MB of the RF algorithm by as much as 49.55%. Also, the proposed C-DRF algorithm used less memory space than P. Latinne et al.'s algorithm by as much as 1.84%. Next, compared to 1,601.15 MB of the original RF algorithm and 214.48 MB of P. Latinne et al.'s algorithm, the proposed C-DRF algorithm used 215.104 MB of memory space for the real estate dataset.

5.1.3. Learning Time. In Figure 5, we show the learning time of each algorithm under the UNSW-NB15 dataset. For the UNSW-NB15 dataset, the learning time of the proposed C-DRF algorithm was 1,987.48 seconds on average, which was faster than 3,363.02 seconds and 2,258.81 seconds of the original RF algorithm and P. Latinne et al.'s algorithm, respectively.

In Figure 5, we show the learning time of the proposed C-DRF algorithm, RF algorithm, and P. Latinne et al.'s algorithm for the real estate dataset. For the real estate dataset, the learning time of P. Latinne et al.'s algorithm was 47.137 seconds on average faster than the other two algorithms.

5.1.4. Test Time. To evaluate the test time of the proposed C-DRF algorithm, we measured the time spending from loading of the forest into the memory to the output file creation. In Figure 6, we show the test time of the proposed C-DRF algorithm, the RF algorithm, and P. Latinne et al.'s algorithm under different datasets. For the UNSW-NB15 dataset, the test time of the proposed C-DRF algorithm was 4.45 seconds, which is faster than the other two algorithms by as much as 38.85% and 10.28%, respectively. For the real estate dataset, the proposed C-DRF algorithm showed the test time of 1.2 seconds, which is faster than 17.66 seconds of the RF algorithm by 56.98%.

6. Discussion

6.1. A MNIST Dataset. Note that, by using the covariance, the proposed C-DRF algorithm dynamically determines the number of trees composing a forest under different datasets. However, as shown in Table 5, the number of trees of the proposed C-DRF algorithm under the MNIST dataset can be larger than the original RF algorithm and P. Latinne et al.'s algorithm under the same classification accuracy. Also, we can observe that the proposed C-DRF algorithm uses 12,854.79MB of memory more than 6,542.73MB and 8,773.15MB of the original RF algorithm and P. Latinne et al.'s algorithm, respectively. That is, the proposed C-DRF algorithm requires more memory space than the original RF algorithm by as much as 49.11% and P. Latinne et al.'s algorithm by as much as 31.75%. For the learning time and test time, the proposed C-DRF algorithm also showed the learning time of 10,736.8 seconds and the test time of 7.84 seconds on average. The learning time was larger than 8,664.95 seconds and 7,322.52 seconds of the original RF algorithm and P. Latinne et al.'s algorithm, respectively. The

Table 5: Experimental results for the MNIST dataset.

	RF algorithm	C-DRF algorithm	P. Latinne et al.'s algorithm [19]
# of Trees	100	115	90
Accuracy	94%	94.35%	94.33%
Memory usage	6542.736 MB	12854.79 MB	8773.15 MB
Learning Time	8664.952 sec	10736.8 sec	7322.52 sec
Test Time	6.91 sec	7.84 sec	6.09 sec

test time was larger than 6.91 seconds and 6.09 seconds of the original RF algorithm and P. Latinne et al.'s algorithm, respectively.

This is because the MNIST dataset consists of many correlated data, each of which is represented with the combination of 0 and 1. Since the proposed algorithm is designed by using the covariance as a metric for determining the classification accuracy while increasing the number of trees, this observation implies that the performance of the proposed C-DRF algorithm has a limitation when analyzing the dataset including the highly correlated data from each other.

6.2. A Distributed Environment. Compared to other machine learning algorithms, there is another advantage of the RF algorithm that many trees can be derived independently in distributed computing or multicore environment. This advantage has enhanced the computational efficiency of the RF algorithm. In this paper, it seems that the proposed C-DRF algorithm can not be applied to a distributed computing or multicore environment because it creates a forest in a sequential manner. Note that this sequential manner maximizes the memory efficiency of the proposed C-DRF algorithm by not deriving unused trees. In fact, the covariance calculation result does not affect the tree generation algorithm. That is, the C-DRF algorithm can derive the tree independently for a certain unit such as the number of cores and then perform the covariance calculation at once to obtain the same result. However, since the trees are derived by a specific unit, unused trees may occur in the last iteration, which may cause some memory waste. Therefore, if the user focuses on learning time efficiency, the C-DRF algorithm can be applied in the abovementioned method. On the contrary, if the user focuses on memory efficiency, the C-DRF algorithm can be applied in a sequential manner.

7. Conclusion

As a representative ensemble machine learning algorithm, the RF algorithm has been widely used in various applications. In this paper, to decrease the number of trees in the RF algorithm, we proposed a new dynamic RF algorithm which reduces the number of trees composing the forest. By analyzing the covariance between the number of trees in a forest and the $F1$-measure at each iteration, the proposed C-DRF algorithm composed a forest with the minimum number of trees while ensuring the good-enough classification accuracy. While generating a decision tree at each iteration, the proposed C-DRF algorithm determines whether to increase the number of trees based on a direct nonparametric test for OoB dataset. We evaluated the performance of the proposed C-DRF algorithm in theory and in practice. That is, from the mathematical analysis of the learning time complexity of the proposed C-DRF algorithm, we showed that the learning time of the proposed C-DRF algorithm is faster than the original RF algorithm in practice. From the experimental results under different areas of datasets, we observed that, compared to the original RF algorithm, the proposed C-DRF algorithm showed the fast learning time by as much as 58.68% on average and the fast test time by as much as 47.91% on average. We also showed that, compared to the original RF algorithm, the proposed C-DRF algorithm significantly reduced the memory usage by as much as 68.06% on average while keeping the same classification accuracy. Compared to the best algorithm for decreasing the number of trees, that is, P. Latinne et al.'s algorithm, the proposed C-DRF algorithm showed the fast learning time by as much as 1.55% on average and the fast test time by as much as 0.45% on average. Also, it was shown that, compared to P. Latinne et al.'s algorithm, the proposed C-DRF algorithm reduced the memory usage by as much as 0.35% on average and reduced the number of trees 2.00% while keeping the same classification accuracy. From these observations, we believe that the proposed C-DRF algorithm can be used as an alternative to the original RF algorithm in various fields.

Data Availability

The data used to support the findings of this study have been deposited in https://github.com/S3lab-pnu/Real-estate-dataset and [33].

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Basic Science Research Program (NRF-2018R1D1A3B07043392) and Capacity Enhancement Program for Scientific and Cultural Exhibition Services (NRF-2018X1A3A1069642) through National Research Foundation Korea (NRF) funded by the Ministry of Science, ICT, and Future Planning. This work was also supported by BK21PLUS, Creative Human Resource Development Program for IT Convergence.

References

[1] X. Sun, S. Y. Chiu, and L. A. Cox, "A hill-climbing approach for optimizing classification trees," in *Learning from Data*, vol. 112 of *Lecture Notes in Statistics*, pp. 109–117, Springer, New York, NY, USA, 1996.

[2] J. Oliver, *Decision Graphs: An Extension of Decision Trees*, Monash University, Department of Computer Science, 1992.

[3] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[4] A. Alkhalid, I. Chikalov, and M. Moshkov, "Decision tree construction using greedy algorithms and dynamic programming comparative study," in *Proceedings of the 20th International Workshop on Concurrency, Specification and Programming*, Bialystok University of Technology, Pultusk, Poland, 2011.

[5] S. K. Murthy and S. Salzberg, *Decision Tree Induction: How Effective Is the Greedy Heuristic?* KDD, 1995.

[6] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 3, pp. 418–435, 1992.

[7] W. Fan, S. Stolfo, and P. Chan, "Using conflicts among multiple base classifiers to measure the performance of stacking," in *Proceedings of the ICML-99 Workshop on Recent Advances in Meta-Learning and Future Work*, pp. 10–17, 1999.

[8] K. Woods, W. Philip Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

[9] L. Breiman, "Bagging classifiers," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[10] E. Bauer and R. Kohavi, "Empirical comparison of voting classification algorithms: bagging, boosting, and variants," *Machine Learning*, vol. 36, no. 1, pp. 105–139, 1999.

[11] G. Tsoumakas and I. Vlahavas, "Effective stacking of distributed classifiers," in *Proceedings of the 15th European Conference on Artificial Intelligence*, pp. 340–344, 2002.

[12] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[13] A. Liaw and M. Wiener, "Classification and regression by random forest," *The R Journal*, vol. 2, no. 3, pp. 18–22, 2002.

[14] A. Cutler and G. Zhao, "Pert-perfect random tree ensembles," *Computing Sciences and Statistics*, vol. 33, pp. 490–497, 2001.

[15] J. J. Rodríguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: a new classifier ensemble method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619–1630, 2006.

[16] M. Robnik-Sikonja, "Improving random forests," in *Proceedings of the European Conference on Machine Learning*, pp. 359–370, Springer, Berlin, Germany, 2004.

[17] S. Bernard, S. Adam, and L. Heutte, "Dynamic random forests," *Pattern Recognition Letters*, vol. 33, no. 12, pp. 1580–1586, 2012.

[18] A. Cuzzocrea, S. L. Francis, and M. M. Gaber, "An Information-theoretic approach for setting the optimal number of decision trees in random forests," in *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, pp. 1013–1019, UK, October 2013.

[19] P. Latinne, O. Debeir, and C. Decaestecker, "Limiting the number of trees in random forests," in *Multiple Classifier Systems (Cambridge, 2001)*, vol. 2096 of *Lecture Notes in Computer Science*, pp. 178–187, Springer, Berlin, Germany, 2001.

[20] J. Shotton, T. Sharp, A. Kipman et al., "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[21] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*, Cengage Learning, 2014.

[22] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 5, pp. 649–659, 2008.

[23] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, pp. 213–217, 2016.

[24] S. R. Johnson and A. Jain, "An improved intrusion detection system using random forest and random projection," *International Journal of Scientific and Engineering Research*, vol. 7, 2016, Probe, 2, U2R.

[25] S. Siegel and N. J. Castellan, *Nonparametric Statistics for the Behavioral Sciences*, McGraw-Hill, 2nd edition, 1988.

[26] B. Lakshminarayanan, D. M. Roy, and Y. W. Teh, "Mondrian forests: efficient online random forests," in *Advances in Neural Information Processing Systems*, pp. 3140–3148, 2014.

[27] J. R. Quinlan, *C4. 5: Programs for Machine Learning*, Elsevier, 2014.

[28] R. J. A. Little, "Robust estimation of the mean and covariance matrix from data with missing values," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 37, no. 1, pp. 23–38, 1988.

[29] R. E. Skelton and M. Ikeda, "Covariance controllers for linear continuous-time systems," *International Journal of Control*, vol. 49, no. 5, pp. 1773–1785, 1989.

[30] E. A. Mäntysaari, R. L. Quaas, and Y. T. Gröhn, "Simulation study on covariance component estimation for two binary traits in an underlying continuous scale," *Journal of Dairy Science*, vol. 74, no. 2, pp. 580–591, 1991.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[32] "real estate dataset," 2017, https://github.com/S3lab-pnu/Real-estate-dataset.

[33] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proceedings of the Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6, IEEE, 2015.

[34] G. Louppe, "Understanding random forests: from theory to practice," 2014, https://arxiv.org/abs/1407.7502.

[35] "KDD Cup 1999," 2007, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.