An Algorithm for Reducing the Number of Distinct Branching Conditions in a Decision Forest

Atsuyoshi Nakamura[✉][1][0000−0001−7078−8655] and Kento Sakurada[1]

Hokkaido University, Kita 14, Nishi 9 Kita-ku Sapporo 060-0814, Japan
{atsu,k_sakurada}@ist.hokudai.ac.jp

Abstract. Given a decision forest, we study a problem of reducing the number of its distinct branching conditions without changing each tree's structure while keeping classification performance. A decision forest with a smaller number of distinct branching conditions can not only have a smaller description length but also be implemented by hardware more efficiently. To force the modified decision forest to keep classification performance, we consider a condition that the decision paths at each branching node do not change for $100\sigma\%$ of the given feature vectors passing through the node for a given $0 \leq \sigma < 1$. Under this condition, we propose an algorithm that minimizes the number of distinct branching conditions by sharing the same condition among multiple branching nodes. According to our experimental results using 13 datasets in UCI machine learning repository, our algorithm succeeded more than 90% reduction on the number of distinct branching conditions for random forests learned from 3 datasets without degrading classification performance. 90% condition reduction was also observed for 7 other datasets within 0.17 degradation of prediction accuracy from the original prediction accuracy at least 0.673.

Keywords: decision forest · algorithm · simplification

1 Introduction

A decision tree is a popular classifier not only for its classification performance but also for its high interpretability. As base classifiers of an ensemble classifier, decision trees are also preferred due to its usability such as being able to calculate feature importance. Therefore, several decision-tree-based ensemble classifiers such as random forests [3], extremely randomized trees [8] and gradient boosted regression trees [7] have been developed so far.

Various researches have been done to obtain more useful decision trees and forests, and one of them is their simplification such as pruning [9, 10]. Simplification of decision trees and forests is important not only as a countermeasure for overfitting but also as enhancement measures of interpretability and prediction time-and-space complexities.

Simplification methods developed so far for a decision forest, reduce the number of branching nodes or trees. Here, however, we proposed an algorithm that reduces the number of distinct branching conditions without changing the structure of each tree in a given forest. So, the number of branching nodes does not change but multiple nodes become to share the same branching condition by applying our algorithm.

This research is motivated by the recent development of hardware implementation of a random forest for fast inference. Implementation using FPGA has been successful in accelerating the inference process of a random forest [12, 1]. In the system proposed in [1], all the branching conditions are processed by fast comparators in parallel, then their binary outputs are used to parallelly evaluate which leaf is reached through a boolean net. The decrease of the number of distinct branching conditions reduces the number of comparators needed for this implementation.

In this paper, we first formalize our simplification problem as the problem of minimizing the number of distinct branching conditions in a decision forest by sharing conditions under the restriction that, given a set of feature vectors D_{L} and $0 \leq \sigma < 1$, at each branching node in each component tree, paths of at most $100\sigma\%$ of the vectors $\mathbf{x} \in D_{\mathrm{L}}$ passing through the node, can be changed. Assume that all the features are numerical and all the branching conditions are expressed as $x_i \leq \theta_i$ for ith component x_i of a feature vector $\mathbf{x}$ and a fixed threshold θ_i. Under the above restriction, the range in which θ_i can take a value becomes some interval $[\ell_i, u_i)$, and the above problem can be reduced to the problem of obtaining a minimum set that intersects all the given intervals, which are defined for each feature. We propose Algorithm Min_IntSet for this reduced problem and prove its correctness. We also develop Algorithm Min_DBN for our original problem using Min_IntSet to solve the reduced problem for each feature.

Effectiveness of our algorithm Min_DBN is demonstrated for the random forests that are learned from 13 datasets in UCI machine learning repository [6]. Without prediction performance degradation, Min_DBN with $\sigma = 0$ succeeds to reduce the number of distinct branching conditions at least 48.9% for all the datasets but RNA-Seq PANCAN dataset, which has more than 30 times larger number of features than the number of its train instances. For hapmass and magic datasets, which have the two largest number of instances, more than 90% condition reduction is achieved by running Min_DBN with $\sigma = 0$. All the datasets except RNA-Seq PANCAN, iris and blood datasets, whose vectors of the last two datasets are composed of only four features, achieves more than 90% condition reduction by allowing larger rate of path change ($\sigma = 0.1, 0.2, 0.3$) within about 0.17 prediction accuracy decrease from the original prediction accuracy, which is at least 0.673.

2 Problem Setting

Consider a d-dimensional real feature space $X = \mathbb{R}^d$ and a finite class-label space $C = \{1, \ldots, \ell\}$. A *classifier* $f : X \to C$ is a function that assigns some label

$c \in C$ to an arbitrary input feature vector $\mathbf{x} = (x_1, ..., x_d) \in \mathbb{R}^d$. A *decision tree* T is a kind of classifier that decides the class label assignment c of an input $\mathbf{x}$ by starting from the root node, repeatedly choosing a child node at each internal node depending on the branching condition attached to the node, and assigning label c that is labeled at the reached leaf node. Here, we assume that each internal node has just two child nodes and the attached branching condition is in the form of $x_i \leq \theta_i$. For a given feature vector $\mathbf{x}$, the left child node is chosen at a node if its branching condition $x_i \leq \theta_i$ is satisfied in the class label assignment process using a decision tree. Otherwise, the right child node is chosen. The *$\mathbf{x}$'s path in a decision tree T* is the path from the root node to the reached leaf node in the class label assignment process. We let a pair (i, θ_i) of a feature id i and a threshold θ_i denote the branching condition $x_i \leq \theta_i$. A set of decision trees is called a *decision forest.*

We consider the following problem.

Problem 1 (Problem of minimizing the number of distinct branching conditions in a decision forest). For a given decision forest $\{T_1, ...T_m\}$, a given set of feature vectors $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ and a given path-changeable rate $0 \leq \sigma < 1$, minimize the number of distinct branching conditions (i, θ_i) by changing the values of some θ_i without changing more than $100\sigma\%$ of feature vectors' paths passing through each node of each decision tree T_i $(i = 1, \ldots, m)$.

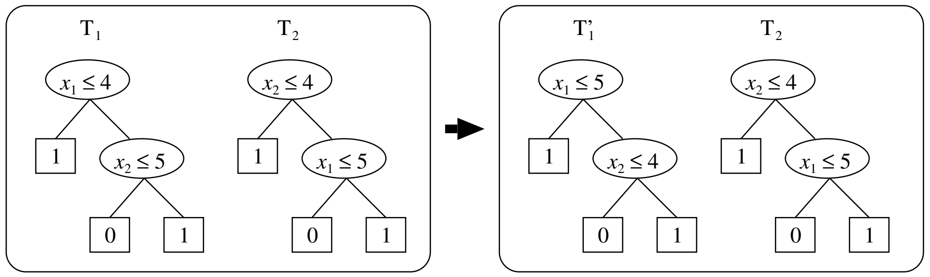

Fig. 1. The number of distinct branching conditions $(1, 4), (1, 5), (2, 4), (2, 5)$ in decision forest $\{T_1, T_2\}$ can be reduced to $(1, 5), (2, 4)$ by changing conditions $(1, 4)$ and $(2, 5)$ in T_1 to $(1, 5)$ and $(2, 4)$, respectively, without changing the path of any feature vector in $\{(1, 1), (2, 7), (7, 2), (8, 8)\}$.

Example 1. Consider Problem 1 for a decision forest $\{T_1, T_2\}$ in Fig. 1, a feature vector set $\{(1, 1), (2, 7), (7, 2), (8, 8)\}$, and a path-changeable rate $\sigma = 0$. The distinct branching conditions (i, θ_i) in decision forest $\{T_1, T_2\}$ are the following four:

$$(i, \theta_i) = (1, 4), (1, 5), (2, 4), (2, 5).$$

The branching conditions $(1, 4)$ and $(2, 5)$ in T_1 can be changed to $(1, 5)$ and $(2, 4)$, respectively, without changing the path of any feature vector in the given set $\{(1,1),(2,7),(7,2),(8,8)\}$. Decision tree T_1' in Fig. 1 is the one that is made from T_1 by this branching-condition change. Decision forest $\{T_1', T_2\}$ has two distinct conditions $(1,5),(2,4)$, which is a solution of Problem 1.

Remark 1. Decision forest $\{T_1, T_2\}$ in Fig. 1 can be outputted by a decision forest learner with training samples $((x_1,x_2),y) = ((1,1),1),((2,7),0),((7,2),0),((8,8),1)$. Assume that two sets of bootstrap samples are $D_1 = \{((1,1),1),((7,2),0),((8,8),1)\}$ and $D_2 = \{((1,1),1),((2,7),0),((8,8),1)\}$, and all the features are sampled for both the sets. In the implementation that the middle points of adjacent feature values are used as threshold candidates for branching conditions, CART algorithm can output T_1 for D_1 and T_2 for D_2. Decision tree T_1' in Fig. 1 has the same Gini Impurity as T_1 at each corresponding branching node for the set of samples D_1.

3 Problem of Minimum Set Intersecting All the Given Intervals

For a given path-changeable rate $0 \leq \sigma < 1$, at each branching node with condition (i, θ_i), the range in which θ_i can take a value without changing more than $100\sigma\%$ of given feature vectors' paths passing through the node, becomes interval $[\ell_i, u_i)$. So, the problem of minimizing the number of distinct branching conditions in a decision forest can be solved by finding clusters of conditions (i, θ_i) whose changeable intervals have a common value for each feature i. Thus, solving Problem 1 can be reduced to solving the following problem for each feature i.

Problem 2 (Problem of Minimum Set Intersecting All the Given Intervals). For a given set of intervals $\{[\ell_1, u_1), \ldots, [\ell_n, u_n)\}$, find a minimum set that intersects all the intervals $[\ell_j, u_j)$ $(j = 1, \ldots, n)$.

We propose Min_IntSet (Algorithm 1) as an algorithm for Problem 2. The algorithm is very simple. First, it sorts the given set of intervals $\{[\ell_1, u_1), \ldots, [\ell_n, u_n)\}$ by lower bound ℓ_i in ascending order (Line 1). For the obtained sorted list $([\ell_{i_1}, u_{i_1}), \ldots, [\ell_{i_n}, u_{i_n}))$, starting from $k = 1$ and $b_1 = 1$, the algorithm finds the kth point s_k by calculating the maximal prefix $([\ell_{i_{b_k}}, u_{i_{b_k}}), \ldots, [\ell_{i_{j-1}}, u_{i_{j-1}}))$ of the list $([\ell_{i_{b_k}}, u_{i_{b_k}}), \ldots, [\ell_{i_n}, u_{i_n}))$ that contain non-empty intersection

$$\bigcap_{h=b_k}^{j-1} [\ell_{i_h}, u_{i_h}) = [\ell_{i_{j-1}}, \min_{h=b_k,\ldots,j-1} u_{i_h}),$$

and t_k is updated such that $t_k = \min_{h=b_k,\ldots,j-1} u_{i_h}$ holds (Line 9). The algorithm can know the maximality of the prefix $([\ell_{i_{b_k}}, u_{i_{b_k}}), \ldots, [\ell_{i_{j-1}}, u_{i_{j-1}}))$ by checking the condition $\ell_{i_j} \geq t_k$ which means that the intersection $[\ell_{i_j}, t_k)$ is empty (Line 4). After finding the maximal prefix with non-empty intersection $[\ell_{i_{j-1}}, t_k)$, the middle point of the interval is set to s_k (Line 5) and repeat the same procedure for the updated k and b_k (Line 8).

The following theorem holds for Algorithm Min_IntSet.

Algorithm 1 Min_IntSet

```
Input: ${[\ell_i, u_i)|i \in I}$ : Non-empty set of intervals
Output: ${s_1, ..., s_k}$ : Minimum set satisfying ${s_1, \ldots, s_k} \cap [\ell_i, u_i) \neq \emptyset$ $(i = 1, \ldots, n)$
        ${I_1, ..I_k}$ : $I_j = {i \in I|s_j \in [\ell_i, u_i)}$ $(j = 1, ..., k)$
 1: $[\ell_{i_1}, u_{i_1}), ..., [\ell_{i_n}, u_{i_n}) \leftarrow$ list sorted by the values of $\ell_i$ in ascending order.
 2: $k \leftarrow 1, t_1 \leftarrow u_{i_1}, b_1 \leftarrow 1$
 3: for $j = 2$ to $n$ do
 4:   if $\ell_{i_j} \geq t_k$ then
 5:     $s_k \leftarrow \frac{\ell_{i_{j-1}} + t_k}{2}$
 6:     $I_k \leftarrow \{i_{b_k}, ..., i_{j-1}\}$
 7:     $k \leftarrow k + 1, t_k \leftarrow u_{i_j}, b_k \leftarrow j$
 8:   else if $u_{i_j} < t_k$ then
 9:     $t_k \leftarrow u_{i_j}$
10:   end if
11: end for
12: $s_k \leftarrow \frac{\ell_{i_n} + t_k}{2}, I_k \leftarrow \{i_{b_k}, ..., i_n\}$
13: return $\{s_1, ..., s_k\}, \{I_1, .., I_k\}$
```

Theorem 1. *For a given set of intervals $\{[\ell_1, u_1), ..., [\ell_n, u_n)\}$, the set $\{s_1, ..., s_k\}$ outputted by Algorithm Min_IntSet is a minimum set that intersects all the intervals $[\ell_j, u_j)$ $(j = 1, \ldots, n)$.*

Proof. We prove the theorem by mathematical induction in the number of intervals n. For $n = 1$, for-sentence between Line 3 and 11 is not executed. At Line 12, s_1 is set as

$$s_1 = \frac{\ell_{i_1} + u_{i_1}}{2}$$

because $t_1 = u_{i_1}$, and at Line 13 Min_IntSet outputs $\{s_1\}$, which is trivially a minimum set that intersects all the interval in the given set $\{[\ell_1, u_1)\}$.

Consider the case with $n = k + 1$. When if-sentence at Line 4 never holds, $\ell_{i_j} \leq \ell_{i_n} < t_1$ holds for all $j = 1, \ldots, n$ and Line 8-9 ensures $t_1 \leq u_{i_j}$ for all $j = 1, \ldots, n$. Thus, $[\ell_{i_n}, t_1)$ is contained by all the intervals and the set that is composed of its middle point s_1 only is trivially a minimum set that intersects all the intervals in the given set $\{[\ell_1, u_1), \ldots, [\ell_n, u_n)\}$.

When if-sentence at Line 4 holds at least once, s_1 is set as

$$s_1 = \frac{\ell_{i_{j-1}} + t_1}{2},$$

and the rest for-loop is executed for j from $j + 1$ to n given $k = 2$, $t_2 = u_{i_j}$, and $b_2 = j$. It is easy to check that $s_2, \ldots, s_k$ calculated in the rest part are the same as those outputted by

$$\text{Min\_IntSet}(\{[\ell_{i_j}, u_{i_j}), ..., [\ell_{i_n}, u_{i_n})\}).$$

The condition of if-sentence at Line 4 ensures that the intersection $[\ell_{i_{j-1}}, t_1)$ of $[\ell_{i_1}, u_{i_1}), \ldots, [\ell_{i_{j-1}}, u_{i_{j-1}})$ does not intersect the rest intervals $[\ell_{i_j}, u_{i_j}), \ldots, [\ell_{i_n}, u_{i_n})$.

So, any minimum set that intersects all the intervals, must contain at least one value that is at most t_1. Any value s_1 in $[\ell_{i_{j-1}}, t_1)$ can minimize the set of the rest intervals that does not contain s_1. In fact, s_1 is set to the middle point of $[\ell_{i_{j-1}}, t_1)$, so the rest set of intervals is minimized. The set of the rest points $s_2, \ldots, s_k$ calculated by Min_IntSet is the same as the set outputted by $Min\_IntSet(\{[\ell_{i_j}, u_{i_j}), ..., [\ell_{i_n}, u_{i_n})\})$, so the minimum set intersecting all the intervals in $\{[\ell_{i_j}, u_{i_j}), ..., [\ell_{i_n}, u_{i_n})\}$ can be obtained using inductive assumption. Thus, Min_IntSet outputs a minimum set that intersects all the given intervals in the case with $n = k + 1$. $\square$

The time complexity of Algorithm Min_IntSet is $O(n \log n)$ for the number of intervals n due to the bottleneck of sorting. Its space complexity is trivially $O(n)$.

4 Algorithm for Minimizing the Number of Distinct Branching Conditions

Algorithm 2 Min_DBN

```
Input: $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ : Set of feature vectors
       $\{T_1, ..., T_m\}$ : decision forest
       $\sigma$ : path-changeable rate ($0 \leq \sigma < 1$)
 1: $L_i \leftarrow \emptyset$ for $i = 1, \ldots, d$
 2: for $j = 1$ to $m$ do
 3:     for each branching node $N_{j,h}$ in $T_j$ do
 4:         $(i, \theta_i) \leftarrow$ branching condition attached to $N_{j,h}$
 5:         $[\ell_{j,h}, u_{j,h}) \leftarrow$ the range of values that $\theta_i$ can take without changing
                    more than $100\sigma\%$ of paths of $\mathbf{x}_1, .., \mathbf{x}_n$ passing through $N_{j,h}$ in $T_j$
 6:         $L_i \leftarrow L_i \cup \{[\ell_{j,h}, u_{j,h})\}$
 7:     end for
 8: end for
 9: for $i = 1$ to $d$ do
10:     $\{s_1, .., s_k\}, \{I_1, ..., I_k\} \leftarrow$ Min_IntSet($L_i$)
11:     for $g = 1$ to $k$ do
12:         for each $(j, h) \in I_g$ do
13:             Replace the branching condition $(i, \theta_i)$ attached to node $N_{j,h}$ with $(i, s_g)$.
14:         end for
15:     end for
16: end for
```

Min_DBN (Algorithm 2) is an algorithm for the problem of minimizing the number of distinct branching conditions in a decision forest. The algorithm uses Algorithm Min_IntSet for each feature $i = 1, \ldots, d$ to find a minimum set of branching thresholds that can share the same value without changing $100\sigma\%$ of paths of given feature vectors passing through each node of each tree in a given decision forest.

For the branching condition (i, θ_i) attached to each branching node $N_{j,h}$ of decision tree T_j in a given decision forest $\{T_1, \ldots, T_m\}$, Algorithm Min_DBN calculates the range of values $[\ell_{j,h}, u_{j,h})$ that θ_i can take without changing $100\sigma\%$ of paths of a given feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ passing through $N_{j,h}$ in T_j (Line 2-8), and adds the range (interval) to L_i, which is initially set to $\emptyset$ (Line 1). Then, by running Min_IntSet for each L_i $(i = 1, \ldots, d)$, Min_DBN obtains its output $\{s_1, ..., s_k\}$ (Line 10), and the branching condition (i, θ_i) of node $N_{j,h}$ with $s_g \in [\ell_{j,h}, u_{j,h})$ is replaced with (i, s_g) (Line 11-15).

Note that, for node $N_{j,h}$ with branching condition (i, θ_i) in decision tree T_j, the interval $[\ell_{j,h}, u_{j,h})$ in which threshold θ_i can take a value without changing more than $100\sigma\%$ of paths of feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ passing through $N_{j,h}$ in T_j, is expressed as

$$\ell_{j,h} = \inf_{\ell}\{\ell \mid |\{\mathbf{x}_f \in X_{j,h} \mid \ell < x_{f,i} \leq \theta_i\}| \leq \sigma|X_{j,h}|\} \text{ and}$$
$$u_{j,h} = \sup_{u}\{u \mid |\{\mathbf{x}_f \in X_{j,h} \mid \theta_i < x_{f,i} \leq u\}| \leq \sigma|X_{j,h}|\},$$

where

$$X_{j,h} = \{\mathbf{x}_f | \text{The path of } \mathbf{x}_f \text{ in } T_j \text{ passes through node } N_{j,h}\},$$

and $|S|$ for set S denotes the number of elements in S.

Let us analyze time and space complexities of Min_DBN. Let N denote the number of nodes in a given decision forest. For each branching node $N_{j,h}$, Min_DBN needs $O(|X_{j,h}| \log(\sigma|X_{j,h}| + 1)) \leq O(n \log(\sigma n + 1))$ time for calculating $\ell_{j,h}$ and $u_{j,h}$ using size-$(\sigma|X_{j,h}| + 1)$ heap. Min_IntSet(L_i) for all $i = 1, \ldots, d$ totally consumes at most $O(N \log N)$ time. Considering that $O(d)$ time is needed additionally, time complexity of Min_DBN is $O(N(n \log(\sigma n + 1) + \log N) + d)$. Space complexity of Min_DBN is $O(dn + N)$ because space linear in the sizes of given feature vectors and decision forest are enough to run Min_DBN.

5 Experiments

We show the results of the experiments that demonstrate the effectiveness of Algorithm Min_DBN.

5.1 Settings

We used 13 numerical-feature datasets registered in UCI machine learning repository [6], whose numbers of instances, features and distinct class labels are shown in Table 1. In the table, datasets are sorted in the order of the number of instances. The largest one is hepmass dataset that has 7 million instances. The dataset with the largest number of features is RNA-Seq PANCAN whose number of features is more than 20 thousands. Note that the number of features is larger than the number of instances only for this dataset. The number of distinct class

Table 1. Dataset used in our experiments

dataset	#instance	#feature	#class	reference
iris	150	4	3	Iris [6]
parkinsons	195	22	2	Parkinsons [11]
breast cancer	569	30	1	Breast Cancer Wisconsin (Diagnostic) [6]
blood	748	4	2	Blood Transfusion Service Center [13]
RNA-Seq PANCAN	801	20531	5	gene expression cancer RNA-Seq [4]
winequality red	1599	11	11	Wine Quality [5]
winequality white	4898	11	11	Wine Quality [5]
waveform	5000	40	3	Waveform Database Generator (Version 2) [6]
robot	5456	24	4	Wall-Following Robot Navigation [6]
musk	6598	166	2	Musk (Version 2) [6]
epileptic seizure	11500	178	5	Epileptic Seizure Recognition [2]
magic	19020	10	2	MAGIC Gamma Telescope [6]
hepmass	7000000	28	2	HEPMASS (train) [6]

labels are not so large for all the dataset we used, and winequality datasets have the largest number of distinct labels (11 distinct labels).

Decision forests used in the experiments are random forest classifiers [3] which are outputted by the fit method of the sklearn.ensemble.RandomForestClassifier class[1] for the input of each dataset. The parameters of the classifier were set to defaults except the number of trees (n_estimators), the number of jobs to run in parallel (n_jobs) and and the seed used by the random number generator (random_state): n_estimators = 100, n_jobs = -1 (which means the same as the number of processors) and random_state = 1. Note that parameter random_state was fixed in order to ensure that the same decision forest is generated for the same training dataset. Also note that the number of randomly selected features used for branching conditions of each decision tree was set to $\sqrt{d}$ as default value. Each dataset was split into training and test datasets using function sklearn.model_selection.train_test_split, and the training dataset only is fed to the fit method of the classifier. The non-default option parameters for train_test_split are the proportions of the dataset to include in the test (test_size) and the train (train_size) splits, and the seed used the random number generator (random_state): test_size = 0.2, train_size = 0.8 and random_state = $0, \ldots, 9$. Note that 10 different pairs of train and test datasets were generated for each dataset by setting different values to random_state parameter. For each pair of train and test datasets $(D_{\mathrm{L}}, D_{\mathrm{P}})$, a random forest RF was learned using D_{L}, and Min_DBN with parameter σ was run for the RF to obtain RF_σ in which the number of distinct branching conditions was minimized. Accuracies of RF and RF_σ for D_{P} were checked for the labels predicted by the predict method of the classifier. We conducted this procedure for 10 train-test splits of each dataset and obtained the number of distinct branching conditions and

[1] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

the accuracy averaged over 10 runs for each dataset and each random forest in $\{\mathrm{RF}\} \cup \{\mathrm{RF}_\sigma \mid \sigma = 0.0, 0.1, 0.2, 0.3\}$.

5.2 Results

The number of distinct branching conditions and prediction accuracy for each original decision forest and those with reduced distinct branching conditions are shown in Table 2. In the table, *reduction rate of the number of distinct branching conditions*, which is defined as

$$1 - \frac{\#(\text{distinct branching conditions in } \mathrm{RF}_\sigma)}{\#(\text{distinct branching conditions in } \mathrm{RF})},$$

and *prediction accuracy decrease*, which is defined as

$$(\text{accuracy of RF}) - (\text{accuracy of } \mathrm{RF}_\sigma) \quad \text{for the test dataset}$$

are also shown for original decision forest RF and decision forests RF_σ that are outputted by Algorithm Min_DBN for the input decision forest RF and path-changeable-rate σ.

Under the condition that the path of any given feature vector in each decision tree must be the same as that in the original tree ($\sigma = 0$), the reduction rate on the number of distinct branching conditions is at least 48.9% for all the datasets but RNA-Seq PANCAN dataset. The number of features is more than 30 times lager than the number of training instances in RNA-Seq PANCAN dataset, and number of distinct features in the original decision tree is less than 1/10 of the number of features, so the number of appearing branching conditions for each feature might be small, which makes condition-sharing difficult. The reduction rate exceeds 90% for magic and hepmass datasets, and especially it reaches 99% for hepmass dataset. Note that the prediction accuracy decrease is between -0.004 and 0.007 for all the datasets, so no degradation of prediction performance was observed.

There is a tradeoff between reduction rate and prediction accuracy decrease, that is, larger reduction rate causes larger prediction accuracy decrease, and it can be to some extent controllable by path-changeable rate σ. By using larger σ, reduction rate can be increased but prediction accuracy decrease is also increased. 90% reduction rate is achieved by waveform dataset with 0.3% prediction accuracy decrease (PAD), by parkinsons and breast cancer datasets with 5-7% PAD, by musk and epileptic seizure datasets with about 11% PAD, by winequality and robot datasets with 14-17% PAD. Note that the minimum prediction accuracy of the original random forests among the 13 datasets is 67.3%. The reduction rate cannot reach 90% for iris, blood and RNA-Seq PANCAN datasets even using $\sigma = 0.3$. Iris and blood datasets have only four features, which causes a small number of distinct branching conditions even for original decision forest: 168.3 and 154.2 distinct branching conditions for iris and blood datasets, respectively, in 100 decision trees. Considering comparison to the number of trees, such relatively small number of distinct branching conditions seems to be difficult to reduce.

6 Conclusions and Future Work

We formalized a novel simplification problem of a decision forest, proposed an algorithm for the problem and demonstrated its effectiveness on reduction rate of the number of distinct branching conditions for the random forests that were trained using 13 datasets in UCI machine learning repository. Hardware implementation for checking effectiveness of the proposed algorithm on inference efficiency is planned for our next step.

Practically better problem formalization may exist. It might be better to restrict the rate of path changes not at each branching node but at each whole tree. Furthermore, important thing is not the rate of path changes but the rate of reached leaf label changes, so its control might be more preferred. There are a tradeoff between the number of distinct branching conditions and prediction accuracy. So, formalization with restriction directly on one of them might be more useful. As our future work, we would like to pursue better formalizations of the problem, develop their algorithms and analyze their complexity.

Acknowledgments

We thank Assoc. Prof. Ichigaku Takigawa and Assoc. Prof. Shinya Takamaeda-Yamazaki of Hokkaido University for helpful comments to improve this research. We also thank Prof. Hiroki Arimura of Hokkaido University and Prof. Masato Motomura of Tokyo Institute of Technology for their support and encouragement. This work was supported by JST CREST Grant Number JPMJCR18K3, Japan.

References

1. Amato, F., Barbareschi, M., Casola, V., Mazzeo, A.: An fpga-based smart classifier for decision support systems. In: Intelligent Distributed Computing VII. pp. 289–299. Springer International Publishing (2014)
2. Andrzejak, R., Lehnertz, K., Rieke, C., Mormann, F., David, P., CE, E.: Indications of nonlinear deterministic and finite dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state. Phys. Rev. E **64**(061907) (2001)
3. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
4. Cancer Genome Atlas Research Network, Weinstein, J.N., Collisson, E.A., Mills, G.B., Shaw, K.R., Ozenberger, B.A., Ellrott, K., Shmulevich, I., Sander, C., Stuart, J.M.: The cancer genome atlas pan-cancer analysis project. Nature genetics **45**(10), 1113–1120 (2013)
5. Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J.: Modeling wine preferences by data mining from physicochemical properties. Decision Support Systems **47**(4), 547–553 (2009)
6. Dua, D., Karra Taniskidou, E.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
7. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics **29**, 1189–1232 (2000)

8. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning **63**(1), 3–42 (2006)
9. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer Series in Statistics, Springer New York Inc. (2001)
10. Kulkarni, V.Y., Sinha, P.K.: Pruning of random forest classifiers: A survey and future directions. In: 2012 International Conference on Data Science Engineering (ICDSE). pp. 64–68 (2012)
11. Little, M.A., McSharry, P.E., Roberts, S.J., Costello, D.A., Moroz, I.M.: Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. BioMedical Engineering OnLine **6**(23) (2007)
12. Van Essen, B., Macaraeg, C., Gokhale, M., Prenger, R.: Accelerating a random forest classifier: Multi-core, gp-gpu, or fpga? In: Proceedings of the 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines. pp. 232–239 (2012)
13. Yeh, I.C., Yang, K.J., Ting, T.M.: Knowledge discovery on rfm model using bernoulli sequence. Expert Systems with Applications **36**(3), 5866–5871 (2009)

Table 2. Number of distinct branching conditions and prediction accuracy of original random forest classifier and those outputted by Min_DBN

	original	outputted by Min_DBN			
		$\sigma = 0.0$	$\sigma = 0.1$	$\sigma = 0.2$	$\sigma = 0.3$
dataset	#(distinct conditions)(95% confidence interval) reduction rate of #(distinct conditions) prediction accuracy(95% confidence interval) prediction accuracy decrease				

iris	168.3(±6.5)
	0
	0.943(±0.044)
	0
		53.1(±2.1)
		0.684
		0.937(±0.043)
		0.007
			46.0(±2.3)
			0.727
			0.917(±0.038)
			0.027
				37.8(±2.4)
				0.775
				0.857(±0.058)
				0.087
					31.1(±2.1)
					0.815
					0.833(±0.059)
					0.11

parkinsons	1005.5(±33.9)
	0
	0.89(±0.031)
	0
		398.0(±13.4)
		0.604
		0.885(±0.029)
		0.005
			176.9(±6.5)
			0.824
			0.854(±0.036)
			0.036
				105.9(±1.8)
				0.895
				0.846(±0.032)
				0.044
					77.7(±2.5)
					0.923
					0.841(±0.031)
					0.049

breast cancer	1389.8(±54.7)
	0
	0.961(±0.015)
	0
		572.0(±15.2)
		0.588
		0.958(±0.015)
		0.003
			210.7(±7.3)
			0.848
			0.911(±0.028)
			0.05
				130.8(±4.7)
				0.906
				0.891(±0.03)
				0.069
					98.5(±2.9)
					0.929
					0.86(±0.026)
					0.101

blood	154.2(±5.8)
	0
	0.761(±0.018)
	0
		78.5(±2.7)
		0.491
		0.765(±0.015)
		-0.004
			71.0(±3.4)
			0.54
			0.755(±0.014)
			0.006
				62.7(±3.0)
				0.593
				0.75(±0.012)
				0.011
					58.5(±2.6)
					0.621
					0.743(±0.014)
					0.018

RNA-Seq PANCAN	1938.5(±24.1)
	0
	0.998(±0.003)
	0
		1851.5(±21.7)
		0.045
		0.998(±0.003)
		0.0
			1664.2(±23.9)
			0.142
			0.996(±0.004)
			0.001
				1641.0(±23.7)
				0.153
				0.994(±0.004)
				0.003
					1633.8(±22.9)
					0.157
					0.99(±0.003)
					0.007

winequality red	4066.2(±25.4)
	0
	0.68(±0.01)
	0
		873.8(±8.7)
		0.785
		0.684(±0.008)
		-0.003
			583.7(±7.3)
			0.856
			0.626(±0.015)
			0.054
				424.8(±5.2)
				0.896
				0.566(±0.009)
				0.114
					340.0(±5.2)
					0.916
					0.524(±0.016)
					0.156

winequality white	7194.9(±48.3)
	0
	0.673(±0.012)
	0
		1358.2(±12.3)
		0.811
		0.673(±0.011)
		0.0
			871.7(±5.8)
			0.879
			0.583(±0.017)
			0.09
				661.8(±6.6)
				0.908
				0.5(±0.012)
				0.173
					559.9(±10.3)
					0.922
					0.443(±0.011)
					0.23

waveform	32232.4(±140.8)
	0
	0.849(±0.004)
	0
		5774.0(±26.3)
		0.821
		0.847(±0.006)
		0.002
			1233.6(±12.0)
			0.962
			0.846(±0.009)
			0.003
				646.2(±7.8)
				0.98
				0.838(±0.009)
				0.011
					471.8(±6.6)
					0.985
					0.824(±0.01)
					0.025

robot	9337.7(±171.0)
	0
	0.994(±0.002)
	0
		2966.1(±51.7)
		0.682
		0.994(±0.002)
		0.0
			986.9(±18.3)
			0.894
			0.91(±0.009)
			0.084
				579.1(±11.4)
				0.938
				0.853(±0.009)
				0.141
					396.1(±10.3)
					0.958
					0.854(±0.014)
					0.14

musk	12306.9(±141.4)
	0
	0.977(±0.004)
	0
		6288.6(±66.1)
		0.489
		0.977(±0.004)
		0.0
			2700.7(±26.6)
			0.781
			0.935(±0.008)
			0.042
				1617.5(±25.3)
				0.869
				0.885(±0.01)
				0.092
					1125.2(±20.3)
					0.909
					0.867(±0.008)
					0.11

epileptic seizure	75492.5(±131.4)
	0
	0.696(±0.009)
	0
		23540.7(±61.0)
		0.688
		0.695(±0.008)
		0.001
			6857.6(±29.0)
			0.909
			0.587(±0.006)
			0.109
				3746.4(±21.1)
				0.95
				0.441(±0.008)
				0.256
					2738.8(±13.7)
					0.964
					0.304(±0.008)
					0.392

magic	110441.9(±526.1)
	0
	0.877(±0.003)
	0
		9535.3(±34.1)
		0.914
		0.877(±0.003)
		0.0
			2440.7(±24.7)
			0.978
			0.8(±0.005)
			0.077
				1352.1(±10.2)
				0.988
				0.595(±0.008)
				0.282
					985.6(±8.2)
					0.991
					0.426(±0.007)
					0.451

hepmass	45125069.6(±8951.8)
	0
	0.821(±0.0)
	0
		431851.6(±263.1)
		0.99
		0.821(±0.0)
		0.0
			57199.1(±60.1)
			0.999
			0.809(±0.0)
			0.011
				29626.6(±83.4)
				0.999
				0.773(±0.001)
				0.048
					21704.1(±63.8)
					1.0
					0.684(±0.003)
					0.137