Balanced Random Hyperboxes for Class Imbalanced Problems

Thanh Tung Khuat and My Hanh Le

***Abstract*—A Random Hyperboxes (RH) classifier is a simple but powerful randomization-based ensemble model, including hyperbox-based classifiers used as base learners. Individual learners in this ensemble model are trained on random subspaces of both instance and feature spaces. This facet results in a flexible mechanism to form a high-performing classifier competitive with other ensemble models in the literature. Like other machine learning models, however, the RH classifier also faces inefficiency when dealing with class-imbalanced datasets. Meanwhile, data containing highly imbalanced class distributions are prevalent in practical applications. Hence, this paper proposes a new variant of the original RH model, namely Balance Random Hyperboxes (BRH), to bypass this drawback effectively. The proposed method uses an under-sampling strategy to build individual learners instead of the random sampling method employed in the original RH model. The experiment conducted on software fault datasets, which show a highly class-imbalanced property, indicated the proposed method's efficiency compared to the original RH model and other ensemble models.**

***Index Terms*—Balanced random hyperboxes, ensemble learning, randomization-based learning, class-imbalanced data, software fault prediction.**

I. Introduction

RANDOM hyperboxes (RH) classifier is a novel classification algorithm recently introduced in [1]. Experimental results [1] proved that the RH classifier has been competitive with other ensemble models such as Random Forest [2], Light Gradient Boosted Machine (LightGBM) [3], and Extreme Gradient Boosting (XGBoost) [4]. A RH model contains many individual hyperbox-based classifiers, in which each base learner is trained on a subset of both samples and features. Hyperbox-based classifiers are formed from basic components, i.e., hyperbox fuzzy sets and membership functions, based on a certain architecture, for example, a network. Fig. 1 shows an example of a resulting hyperbox-based classifier trained on a two-class and two-dimensional dataset. A comprehensive survey on hyperbox-based classifiers can be found in [5]. General fuzzy min-max neural network (GFMMNN) [6] is one of the typical hyperbox-based classifiers resulting in high classification performance for pattern recognition problems [7]. Therefore, the original RH model [1] used GFMMNNs as base learners.

Similarly to Random Forest [2], the RH model also belongs to the bagging ensemble group, where individual classifiers are independently built from different subsets of instances and features. The ensemble of classifiers usually increases the classification performance of single models [8], [9] by combining decisions of base learners using several combination strategies such as majority voting. However, if neither of the base learners alone can deal with the class imbalanced problem, then ensemble learning algorithms will also face negative impacts of class imbalanced property on the classification performance [8]. In classification, a class-imbalanced issue happens when one class, frequently the one (minority class) that describes the concept of interest, has a number of instances much smaller than the ones from other classes. If the classes in such datasets are able to be well separated in the feature space, good classification accuracy may be easily achieved regardless of the imbalance between classes. However, the presence of class overlap and small disjuncts [10] in most of the practical problems results in the difficulty for learning algorithms to discriminate classes from each other in imbalanced datasets. In this case, the learning models are usually biased towards the class with a high number of samples to optimize the overall accuracy without taking into consideration of the relative distribution for each class. In addition, the class imbalance problem is almost popular in real-world problems, ranging from software defects prediction [11], [12], [13], fraud detection [14], medical diagnosis [15], [16], anomaly detection [17], [18], chemical engineering applications [19], to many other research fields [20]. As a result, it causes severe troubles when applying machine learning algorithms to practical applications.

Like other ensemble models, however, the RH model also shows the inefficiency on the imbalanced datasets because its base learners, e.g., GFMMNNs, are not designed with a unique mechanism to deal effectively with class-imbalanced problems. Therefore, this paper aims to modify the original RH learning algorithm of the RH classifier so that it can tackle classification problems on imbalanced datasets effectively.

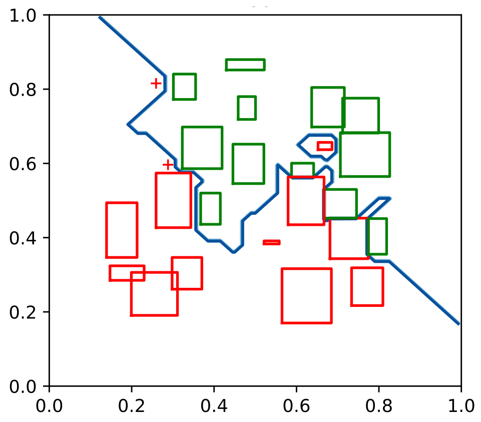


Fig. 1. An example of a hypebox-based classifier with its decision boundaries

Manuscript received January 01, 2021; revised March 25, 2021. This research was financially supported by the Ministry of Education and Training of Vietnam under Grant B2019-DNA-03.

Thanh Tung Khuat is a PhD candidate at the Advanced Analytics Institute, Faculty of Engineering and IT, University of Technology Sydney, NSW 2007, Australia (corresponding author; e-mail: thanhtung09t2@gmail.com).

My Hanh Le is a Senior Lecturer at the IT Faculty, University of Science and Technology, The University of Danang, Danang 550000, Vietnam (email: ltmhanh@dut.udn.vn).

As a result, the random undersampling technique will be deployed to build up the balanced training set for each base learner rather than random subsampling in the original RH classifier. The proposed method's effectiveness will be assessed on the software fault prediction problems, which frequently show a two-class highly imbalanced characteristic.

With the rapid development of software systems in industrial sectors, there have been increased risks of potential defects in software modules. However, testing processes are expensive and time-consuming, and so it is challenging to allocate rational resources and budgets for rigorous testing and validation of all software elements in a software development project. Hence, automated identification of vulnerable software modules is highly expected in projects to support project managers to allocate limited resources effectively for software testing activities. For instance, potentially faulty elements are given more effort to inspect and validate, which aims to achieve a better quality of products. Like other real-world problems, training sets of software fault classification problems are highly imbalanced because most of the defects in software systems only exist in a small number of modules. As a result, numbers of faulty instances in software quality datasets are much smaller than those of non-faulty patterns [21]. Therefore, the software fault classification problem is an appropriate subject to assess the efficiency of the proposed approach.

The main contributions of this study can be listed as follows:

- We propose a balanced random hyperboxes classifier as a variant of the original random hyperboxes model for class-imbalanced problems.
- We assess the performance of the proposed method on the software fault prediction problem, which includes highly class imbalanced instances. The proposed method is also compared to other ensemble models such as the original RH [1], random forest (RF) [2], balanced random forest (BRF) [22], and heterogeneous ensembles with and without sampling techniques [23].

The rest of this paper is organized as follows. Section II provides some background knowledge regarding GFMMNN architecture, its training algorithms, and the original RH model. Next, section III shows the details of the proposed method. The experimental results and discussions of the proposed method are presented in section IV. Finally, section V concludes key findings and informs potential research directions for future studies.

II. Background

Because base learners in the RH model are GFMMNNs, this section first briefly describes the architecture of the GFMMNN and its learning algorithms. Next, the detail of the RH model is presented.

A. General Fuzzy Min-Max Neural Network

GFMMNN is a unified version of fuzzy min-max neural networks for classification [24] and clustering [25]. It contains three layers, i.e., input, hyperbox, and output layers. Fig. 2 shows a general structure of GFMMNN. This kind of neural network accepts the input patterns in the form of intervals with lower and upper bounds. Hence, the input layer comprises $2 \cdot n$ nodes, in which the first n nodes contain n-dimensional lower bounds, and the rest of n nodes are n-dimensional upper bounds of each input pattern. Each input node is fully connected to all m nodes in the hidden layer (hyperbox layer). The connection weights from lower bound input nodes to hyperbox nodes form a matrix $\mathbf{V}$ including all minimum points of hyperboxes. Similarly, the connection weights from upper bound input nodes to hyperboxes nodes generate a matrix $\mathbf{W}$ containing all maximum points for corresponding hyperboxes. The values of $\mathbf{V}$ and $\mathbf{W}$ will be adjusted during the training process. In addition to min-max points V and W, each hyperbox B_i has an activation function, $0 \leq b_i(X, B_i) \leq 1$, called a membership function as shown in Eq. (1) to compute the degree of fit from an input pattern $X = [X^l, X^u]$, where $X^l = \{x_1^l, \ldots, x_n^l\}$ and $X^u = \{x_1^u, \ldots, x_n^u\}$ are lower and upper bounds of X, to that hyperbox.

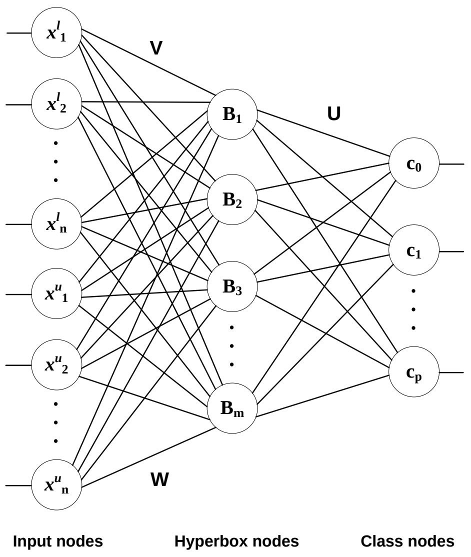


Fig. 2. A general structure of GFMMNN

$$b_i(X, B_i) = \min_{j=1}^{n}(\min([1 - f(x_j^u - w_{ij}, \gamma_j)], [1 - f(v_{ij} - x_j^l, \gamma_j)])) \tag{1}$$

where $f(\mu, \gamma)$ is a ramp function presented in Eq. 2, $\gamma = (\gamma_1, \gamma_2, ..., \gamma_n)$ emcompasses the sensitivity parameters to represent the speed of decreasing of membership values.

$$f(\mu, \gamma) = \begin{cases} 1, & \text{if } \mu \cdot \gamma > 1 \\ \mu \cdot \gamma, & \text{if } 0 \leq \mu \cdot \gamma \leq 1 \\ 0, & \text{if } \mu \cdot \gamma < 0 \end{cases} \tag{2}$$

Each hyperbox B_i is linked to each class node c_j in the output layer by a weight u_{ij} saved in a matrix $\mathbf{U}$ such that:

$$u_{ij} = \begin{cases} 1, \text{if } class(B_i) = c_j \\ 0, \text{otherwise} \end{cases} \tag{3}$$

The output layer contains $p + 1$ nodes for p classes and an unlabelled node p_0, in which node p_0 is used to connect all unlabelled hyperboxes.

In the classification phase, each input pattern X is assigned to the class of the hyperbox B_i showing the maximum membership value of b_i. If there are many hyperboxes with different classes showing the same maximum membership value, hyperbox with the smallest Manhattan distance from the input pattern to its central point is selected as a final winner hyperbox to determine the predicted class.

B. Learning Algorithms for GFMMNN

To build GFMMNN models, we need to use learning algorithms to generate and adjust the sizes of hyperboxes to cover all training input patterns. In this paper, we focus on supervised learning problems, so the learning algorithms in this part are presented for training sets with given classes for all samples. This study assesses the effectiveness of the proposed method on two learning algorithms for GFMMNNs used as base learners in the RH classifier; therefore, this section presents both original and improved versions of online learning algorithms for the GFMMNN.

Online learning algorithms for the GFMMNN use a single scan through training input patterns to build and adjust hyperboxes covering those input patterns. For each input pattern, $X = [X^l, X^u]$, with a class label c_X, online learning algorithms' main steps are presented as follows.

1) Original Online Learning Algorithm:
The original online learning algorithm for the GFMMNN (Onln-GFMM) introduced in [6] includes three main steps, i.e., creation of new hyperboxes or expansion of existing hyperboxes, overlap test, and hyperbox contraction.

As a new input pattern X comes to the GFMMNN, the Onln-GFMM first selects the existing hyperboxes with the same class as c_X to compute membership values between X and these hyperboxes. Next, these hyperboxes are considered as expandable candidates to cover X beginning from the hyperbox with the maximum membership value. If the maximum membership degree gets a value of one, the learning process continues to handle the next training sample. In contrast, a three-step procedure is performed as follows:

a) Hyperbox expansion: The selected expandable hyperbox B_i will be checked for the expansion condition, i.e., a predefined maximum hyperbox size θ using Eq. (4).

$$\max(w_{ij}, x_j^u) - \min(v_{ij}, x_j^l) \leq \theta, \ \forall j \in [1, n] \tag{4}$$

If the constraint in Eq. (4) is satisfied, the hyperbox B_i is extended to a new size to cover X as follows:

$$w_{ij} \leftarrow \max(w_{ij}, x_j^u); \ v_{ij} \leftarrow \min(v_{ij}, x_j^l), \ \forall j \in [1, n] \tag{5}$$

Otherwise, the hyperbox with the next highest membership value is considered for expansion. If there is at least one hyperbox B_i expanded, it will be tested for overlap with the hyperboxes representing other classes than c_X. If all of the selected hyperboxes cannot be extended to contain X, a new hyperbox B_i is generated with the same coordinates as X, and the algorithm continues with the next training sample.

b) Hyperbox overlap test: Overlap test operation is conducted between the newly expanded hyperbox B_i and other hyperboxes B_k representing different classes. There are four overlap test cases for each dimension j as below (initially $\delta^{old} = 1$):

Case 1: $v_{ij} \leq v_{kj} < w_{ij} \leq w_{kj}$;
$\delta^{new} = \min(w_{ij} - v_{kj}, \delta^{old})$

Case 2: $v_{kj} \leq v_{ij} < w_{kj} \leq w_{ij}$;
$\delta^{new} = \min(w_{kj} - v_{ij}, \delta^{old})$

Case 3: $v_{ij} < v_{kj} \leq w_{kj} < w_{ij}$;
$\delta^{new} = \min(\min(w_{kj} - v_{ij}, w_{ij} - v_{kj}), \delta^{old})$

Case 4: $v_{kj} < v_{ij} \leq w_{ij} < w_{kj}$;
$\delta^{new} \leftarrow \min(\min(w_{kj} - v_{ij}, w_{ij} - v_{kj}), \delta^{old})$

If one of the above four cases happens for every dimension j, there is an overlapping region between B_i and B_k. We set up $\Delta = j$ for which δ^{new} gets the minimum value. If there is at least one dimension such that no overlap occurs, we assign $\Delta = -1$, and the learning algorithm continues with the next training pattern.

c) Hyperbox Contraction: If $\Delta \neq -1$, the Δ^{th} dimension is adjusted to narrow down the size of hyperbox B_i. Four contraction procedures corresponding to four overlap test cases are presented as follows:

Case 1: $v_{i\Delta} \leq v_{k\Delta} < w_{i\Delta} \leq w_{k\Delta}$:
$w_{i\Delta}^{new} = v_{k\Delta}^{new} = (v_{k\Delta}^{old} + w_{i\Delta}^{old})/2$

Case 2: $v_{k\Delta} \leq v_{i\Delta} < w_{k\Delta} \leq w_{i\Delta}$:
$v_{i\Delta}^{new} = w_{k\Delta}^{new} = (w_{k\Delta}^{old} + v_{i\Delta}^{old})/2$

Case 3: $v_{i\Delta} < v_{k\Delta} \leq w_{k\Delta} < w_{i\Delta}$:
if $w_{k\Delta} - v_{i\Delta} < w_{i\Delta} - v_{k\Delta}$ then $v_{i\Delta}^{new} = w_{k\Delta}^{old}$
otherwise, $w_{i\Delta}^{new} = v_{k\Delta}^{old}$

Case 4: $v_{k\Delta} < v_{i\Delta} \leq w_{i\Delta} < w_{k\Delta}$:
if $w_{i\Delta} - v_{k\Delta} < w_{k\Delta} - v_{i\Delta}$ then $v_{k\Delta}^{new} = w_{i\Delta}^{old}$
otherwise, $w_{k\Delta}^{new} = v_{i\Delta}^{old}$

2) Improved Online Learning Algorithm:
One of the drawbacks of the Onln-GFMM algorithm is that the contraction process usually results in high misclassification when a high value of θ is used. Therefore, an improved online learning algorithm for the GFMMNN (IOL-GFMM) was introduced in a recent study [26]. The IOL-GFMM learning algorithm eliminates the contraction step from the learning process, and it adds a new constraint for the expanded hyperbox, i.e., no creation of overlapping regions with existing hyperboxes of other classes. As a result, the learning process of this algorithm comprises two main procedures: expansion of hyperboxes and hyperbox overlap test.

a) Hyperbox expansion: Similar to the Onln-GFMM algorithm, when a new training input pattern X goes to the network, the IOL-GFMM learning algorithm will select all existing hyperboxes showing the same class as c_X and then computing membership values from X to all selected hyperboxes. If the maximum membership value is one, the input pattern is fully contained in a hyperbox, and thus the algorithm will continue considering the next input pattern. Otherwise, the selected hyperboxes are considered as expandable candidates to cover X beginning from the hyperbox with the highest membership degree and according to the decreasing order of membership values until there is one hyperbox satisfied. Assuming that B_i is an expandable hyperbox candidate, B_i will be first checked for the maximum hyperbox size condition shown in Eq. (4). If this condition is met, the coordinates of B_i will be temporarily expanded to new sizes using Eq. (5); otherwise, another hyperbox candidate with the next highest membership value is considered. If B_i is expanded, it will be checked for overlap as follows.

b) Hyperbox overlap test: This procedure is carried out between the newly expanded hyperbox B_i and all existing hyperboxes showing other classes than c_X. The overlap test for each pair of hyperboxes is checked utilizing the above four test cases as in the Onln-GFMM algorithm. If there is at least one hyperbox which overlaps with B_i, the cooperates of B_i are reverted to its values before expanding, and another hyperbox with the next highest maximum membership value is chosen, and the above learning process is iterated. In contrast, the new coordinates of B_i remain unchanged, and the learning algorithm continues with the next training input patterns. If no expandable hyperbox candidates can be expanded to cover the input pattern X, a new hyperbox is generated with the same coordinates as X and the learning algorithm continues to consider the next training sample.

C. Random Hyperboxes Model

The RH classifier proposed in [1] is an ensemble model of GFMMNNs used as base learners. Each GFMMNN is trained on a subset of both samples and features. The learning algorithm of this model uses a parameter m_f to specify the maximum number of features used in each base learner and a parameter m_s to regulate the sampling rate for samples. The RH model contains N base learners, and the predicted class is the majority voting of predicted classes returning from all base learners. Given a training set, the building process of base learners is performed as follows.

For each base learner, the algorithm will generate a subset of training samples with a stratified sampling rate of m_s. Then, from this generated subset of training data, the algorithm generates a random integer number d between 1 and m_f. Next, d features are uniformly randomly selected from n input features to form a new training set with a subset of both input samples and features. Finally, this subset is employed to train a GFMMNN base learner using the above online learning algorithms. It can be noted that each base learner is trained on only d features of input training samples, so in the classification phase, this base learner only generates predicted values using the same d features out of n features of each unseen pattern X.

III. The Proposed Balanced Random Hyperboxes

It can be seen that the original RH model uses the stratified random sampling to create a subset of training samples, so the resulting subset of training samples is still imbalanced. This fact reduces the classification performance of the RH classifier on class-imbalanced datasets because GFMMNN base learners do not work well on the imbalanced data. To cope with this drawback, this paper proposes a modified version of the original RH classifier, called Balanced Random Hyperboxes (BRH). The main steps of the proposed method are shown in Fig. 3.

We use a random undersampling technique to select samples from majority classes rather than using stratified random sampling as in the original algorithm. The number of chosen majority samples is equal to the number of samples from the minority class. This operation creates a balanced training set with equal numbers of samples for all classes. Base learners are then trained on different balanced training sets, in which the minority samples are the same, but the majority sample

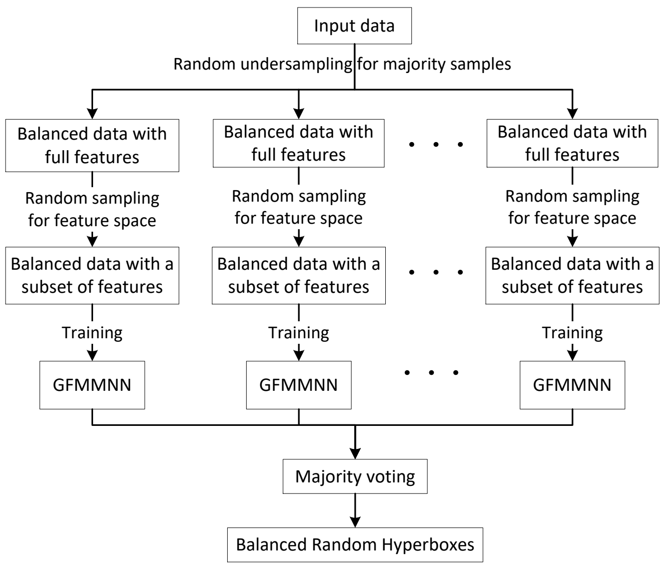

Input data
Random undersampling for majority samples
Balanced data with full features
Random sampling for feature space
Balanced data with a subset of features
Training
GFMMNN
Majority voting
Balanced Random Hyperboxes

Fig. 3. Main steps of a balanced random hyperboxes model

sets are different from each other. Next, from each balanced set with full features, the algorithm selects a set of d features ($1 \leq d \leq m_f$) randomly as in the original RH model to train the GFMMNN base learners. The training algorithms for base learners are the online learning algorithms mentioned above.

Similar to the original RH model, the final predicted class of the BRH model is generated by majority voting from all predicted classes of base learners.

IV. Experiment and Discussion

The classification performance of the proposed method is assessed on the two-class software fault datasets, which are highly imbalanced.

A. Datasets and Evaluation Measures

Experiments were conducted on ten highly imbalanced binary datasets of software defect problems presented in [23]. Table I describes statistics information of the selected datasets, including the number of samples (#Sample), the number of features (#Feature), the number of defective samples (#Defect), the number of non-defective samples (#Non-Defect), the proportion of defective samples to all samples for each dataset (%Defect).

To assess the classification performance of classifiers on the class imbalanced datasets, the research community usually uses precision, recall, and F1-score formed from true positive (TP), true negative (TN), false positive (FP), and false negative (FN) in a confusion matrix. If a defective sample is correctly predicted, then it is called a true positive. If a non-defective sample is exactly classified, it is called a true negative. If a defective sample is classified as a non-defective sample, this case is a fault negative. In case that a non-defective sample is incorrectly classified as a defective sample, a false positive occurs. Precision, Recall, and F1-score are shown as below:

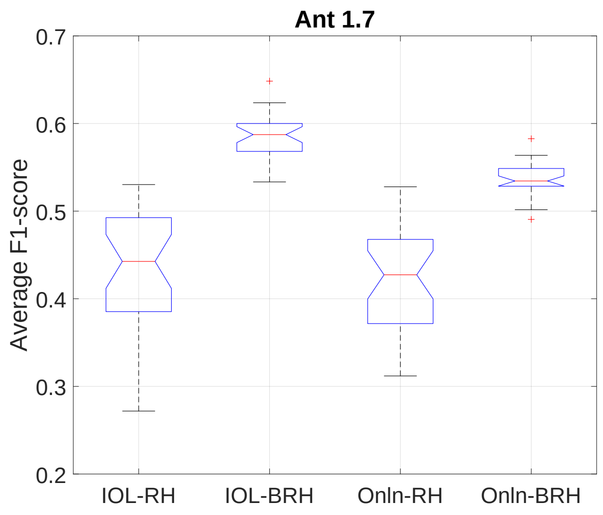
$$Precision = \frac{TP}{TP + FP} \qquad (6)$$

TABLE I
INFORMATION OF EXPERIMENTAL DATASETS

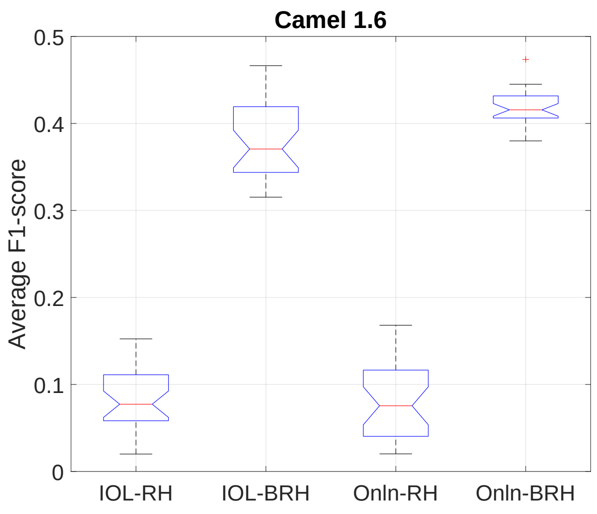
ID	Dataset	#Feature	#Sample	#Defect	#Non-Defect	%Defect
1	JM1	21	7782	1672	6110	21.49
2	KC3	39	194	36	158	18.56
3	PC1	37	705	61	644	8.65
4	ant 1.7	20	745	166	579	22.28
5	ivy 2.0	20	352	40	312	11.36
6	camel 1.6	20	965	188	777	19.48
7	tomcat	20	858	77	781	8.97
8	xalan 2.4	20	723	110	613	15.21
9	poi 2.0	20	314	37	277	11.78
10	synapse 1.2	20	256	86	170	33.59

TABLE II
AVERAGE F1-SCORE OF CLASSIFIERS OVER 30 ITERATIONS

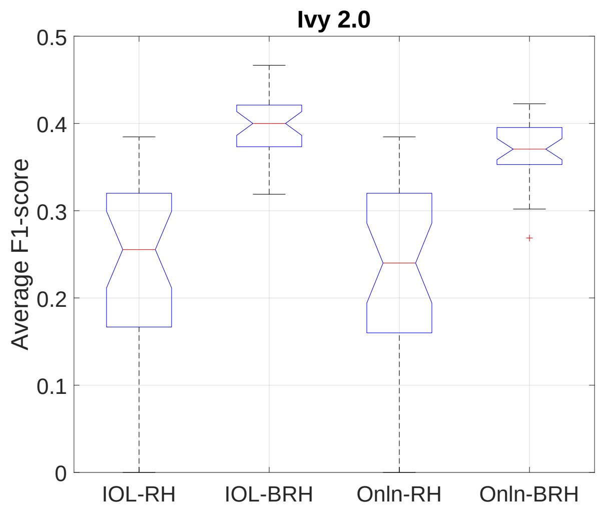
ID	Dataset	IOL-RH	IOL-BRH	Onln-RH	Onln-BRH	RF	BRF	NonSa-HE	DB-HE	SB-HE
1	JM1	0.15266	0.44665	0.16091	0.35371	0.21741	0.43751	0.1944	0.4203	0.4179
2	KC3	0.11585	0.41197	0.11945	0.41011	0.18579	0.40006	0.1667	0.4084	0.4037
3	PC1	0.04521	0.35687	0.03751	0.30977	0.19583	0.33949	0.2	0.3337	0.3263
4	ant 1.7	0.43023	0.58611	0.42095	0.5364	0.54003	0.58312	0.5278	0.6261	0.6037
5	ivy 2.0	0.23635	0.39474	0.23035	0.37009	0.3142	0.38711	0.2759	0.3937	0.3719
6	camel 1.6	0.08073	0.37672	0.08008	0.41829	0.23871	0.43283	0.2321	0.4413	0.4209
7	tomcat	0.03798	0.38772	0.03281	0.32597	0.19955	0.34933	0.2222	0.3899	0.3907
8	xalan 2.4	0.09076	0.45351	0.09153	0.42293	0.2703	0.45372	0.2535	0.4535	0.437
9	poi 2.0	0.11227	0.34526	0.0821	0.34387	0.239	0.35968	0	0.3354	0.335
10	synapse 1.2	0.55093	0.65866	0.56891	0.65669	0.61431	0.66069	0.5333	0.6487	0.6451
Average rank		**8.2**	**2.1**	**8.4**	**4.3**	**6.1**	**2.5**	**7.2**	**2.6**	**3.6**



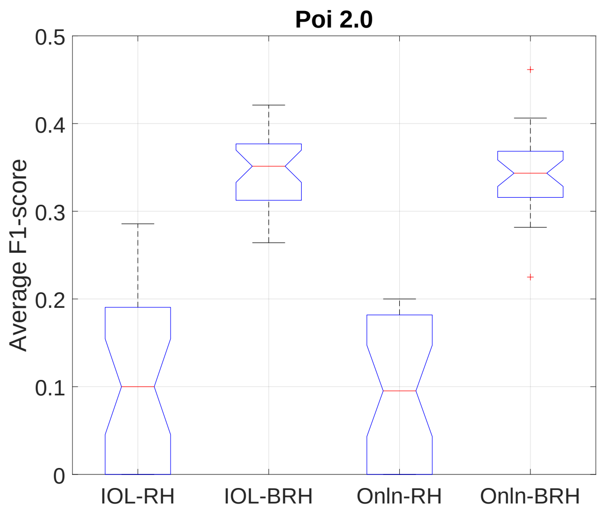
(a) Ant 1.7



(b) Camel 1.6



(c) Ivy 2.0



(d) Poi 2.0

Fig. 4. Box plots show distributions of F1 scores over 30 iterations for the RH model and the proposed BRH model for several representative datasets.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 - score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{8}$$

It can be seen that recall does not show how many samples are incorrectly classified as “defective” samples. Similarly, precision does not provide how many defective samples are misclassified. In contrast, F1-score combines both precision and recall, so it overcomes the drawback of each measure. Hence, F1-score was used to assess the performance of classifiers in this study.

B. Experimental Settings

Experiments were conducted to assess the effectiveness of the BRH model using the Onln-GFMM learning algorithm (Onln-BRH) and the IOL-GFMM algorithm (IOL-BRH) compared to the original RH model using the Onln-GFMM algorithm (Onln-RH) and the IOL-GFMM algorithm (IOL-RH) for building base learners. The performance of the proposed method was also compared to the random forest (RF) [2] and balanced random forest (BRF) [22], the heterogeneous ensemble model without sampling techniques (NonSa-HE), the heterogeneous ensemble model with base learners trained on the different balanced training sets (DB-HE), and the heterogeneous ensemble classifier with base estimators trained on the same balanced training set (SB-HE) [23].

To compare with the outcomes in [23], we used the same experimental settings presented in that paper. For each dataset, 50% of the original data were stratified-sampled randomly to build a training set, and the rest were used as a testing set. This operation was repeated 30 times, and the average F1-score over these 30 iterations is reported in this paper.

For the RH model, we used the same settings presented in [1]: the sampling rate $m_s = 0.5$, the maximum number of used features $m_f = 2\sqrt{n}$, the number of base learners $N = 100$, the maximum hyperbox size $\theta = 0.1$, and sensitivity parameter $\gamma = 1$. For the BRH model, we alse set $m_f = 2\sqrt{n}$, $N = 100$, $\theta = 0.1$, and $\gamma = 1$. For RF and BRF, we used the maximum tree depth of decision trees as 10 [27], $N = 100$, maximum number of used features for each decision tree as $2\sqrt{n}$. For RF, resampling rate was set to 0.5.

C. Experimental Results and Discussions

Table II presents the average F1-score of different classifiers over 30 iterations. To facilitate the comparison step, we rank the performance of classifiers for each dataset. The classifier giving the best F1-score is ranked first, the classifier resulting in the second-best F1-score is ranked second, and so on. After that, the average rank for each classifier over ten datasets is computed and reported in Table II.

Fig. 4 shows the changes in F1-score over 30 iterations of the RH model and BRH model using Onln-GFMM and IOL-GFMM algorithms on four representative datasets. From these outcomes, it can be observed that the classification performance of the proposed method outperforms the original RH model for both Onln-GFMM and IOL-GFMM algorithms used to train base learners. These facts confirm the efficiency of using undersampling techniques to build balanced training sets from which the ensemble model can deal effectively with class-imbalanced datasets. For several datasets with high imbalance, such as *PC1* and *tomcat*, the proposed method is much better than the original RH classifier when the classification performance has been enhanced about ten times.

In comparison to other ensemble models, the performance of the BRH model using the IOL-GFMM learning algorithm, in general, is also better than those using balanced random forest and the heterogeneous ensemble model with different balanced training sets generated by random undersampling. It can also be seen that other ensemble models also achieved much better performance when they were combined with sampling techniques to build balanced training sets.

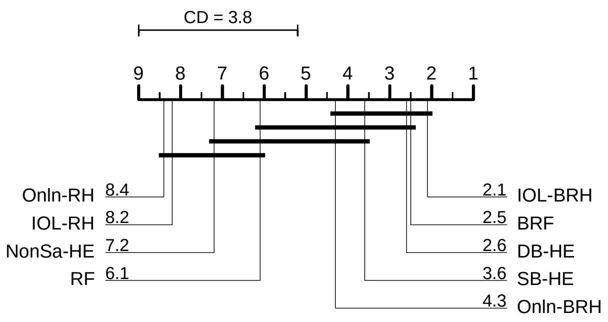


Fig. 5. Critical Difference diagram for the average performance of classifiers using Nemenyi test.

To better understand the difference between classifiers, statistical testing methods were used to calculate the statistical difference. Applying the Friedman rank-sum test [28] for average ranks of nine classifiers on ten datasets, we can obtain the F-distribution value $F_F = 49.1897$. Because the critical value of $F(9-1,(9-1)\cdot(10-1)) = F(8,72)$ for a significance level $\alpha = 0.05$ is 2.0698 $\ll 49.1897$, the null hypothesis is rejected at a high level of confident. This means that there are statistically significant differences between the F1 values of classifiers. Therefore, a post hoc procedure was performed to find the differences among pairs of classifiers. The post hoc test used in this study is Nemenyi test [29]. The Critical Difference (CD) diagram with Nemenyi test for $\alpha = 0.05$ is shown in Fig. 5.

It can be observed that there are statistically significant differences in the classification performance between IOL-BRH and IOL-RH, between Onln-BRH and Onln-RH. This result indicates that the proposed method significantly enhances the original RH classifier’s performance by generating different balanced training sets for base learners. The performance of the IOL-BRH model is also statistically better compared to all other considered ensemble models without using sampling techniques. However, there are no statistically significant differences in the performance among ensemble models using the sampling techniques, i.e., IOL-BRH, BRF, DB-HE, SB-HE, and Onln-BRH.

Although the proposed BRH can achieve high and competitive classification performance for imbalanced datasets in comparison to other classifiers, the interpretability of hyperbox-based individual models is lost. One of the advantages of using hyperbox representations is that we can combine the hyperboxes generated by base learners to create a single interpretable model [30]. However, this technique is only available for the base learners with the same number of used features. In the original RH as well as BRH, the base learners are built from a subset of features, so a new method of aggregation needs to be developed to deal with this issue. Another problem that also needs to put more effort is the construction of mathematical properties and generalization error bounds of the BRH model similar to the original RH model as presented in [1]. With the rapid increase in the volume of data, it is desirable to obtain the algorithms with incremental learning ability. The GFMMNN using the original online learning and IOL-GFMM algorithms may meet this requirement. Nonetheless, when building an ensemble model, we have to build a variety of base learners. As a result, alternative methods need to be developed to ensure both the incremental learning capability and the diversity of base learners.

V. Conclusion and Future Work

This paper proposed the balanced random hyperboxes classifier to improve the original random hyperboxes for class-imbalanced classification problems. The undersampling technique was used to construct balanced training sets for base learners. This modification significantly enhanced performance of the original random hyperboxes model on the imbalanced datasets. Experimental results on highly imbalanced software defect datasets confirmed the efficiency and effectiveness of the proposed approach. The classification performance of our proposed classifier is also competitive with other ensemble models using sampling techniques.

Future studies focus on assessing the performance of the proposed method on multi-class imbalanced datasets. Many other sampling techniques will also be applied to constructing balanced training sets for base learners. We also aim to introduce a cost-sensitive learning method and a weighted learning method to improve the original random hyperboxes model besides the sampling methods proposed in this paper.

References

[1] T. T. Khuat and B. Gabrys, "Random Hyperboxes," *arXiv e-prints 2006.00695*, pp. 1–14, 2020.

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

[3] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Proceedings of Advances in Neural Information Processing Systems*, 2017, pp. 3146–3154.

[4] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[5] T. T. Khuat, D. Ruta, and B. Gabrys, "Hyperbox-based machine learning algorithms: a comprehensive survey," *Soft Computing*, vol. 25, pp. 1325–1363, 2021.

[6] B. Gabrys and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 769–783, 2000.

[7] T. T. Khuat and B. Gabrys, "A comparative study of general fuzzy min-max neural networks for pattern classification problems," *Neurocomputing*, vol. 386, pp. 110 – 125, 2020.

[8] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 463–484, 2012.

[9] H. Dai, W. Wu, J. Li, and Y. Yuan, "Incorporating feature selection in the improved stacking algorithm for online learning analysis and prediction." *Engineering Letters*, vol. 28, no. 4, pp. 1011–1022, 2020.

[10] R. O'Brien and H. Ishwaran, "A random forests quantile classifier for class imbalanced data," *Pattern Recognition*, vol. 90, pp. 232 – 249, 2019.

[11] D. Rodriguez, I. Herraiz, R. Harrison, J. Dolado, and J. C. Riquelme, "Preliminary comparison of techniques for dealing with imbalance in software defect prediction," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 2014, pp. 1–10.

[12] T. T. Khuat and M. H. Le, "Ensemble learning for software fault prediction problem with imbalanced data," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 4, pp. 3241–3246, 2019.

[13] ——, "Binary teaching–learning-based optimization algorithm with a new update mechanism for sample subset optimization in software defect prediction," *Soft Computing*, vol. 23, no. 20, pp. 9919–9935, 2019.

[14] S. Panigrahi, A. Kundu, S. Sural, and A. K. Majumdar, "Credit card fraud detection: A fusion approach using dempster–shafer theory and bayesian learning," *Information Fusion*, vol. 10, no. 4, pp. 354–363, 2009.

[15] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance," *Neural networks*, vol. 21, no. 2-3, pp. 427–436, 2008.

[16] M. A. Aslam, C. Xue, M. Liu, K. Wang, and D. Cui, "Classification and prediction of gastric cancer from saliva diagnosis using artificial neural network," *Engineering Letters*, vol. 29, no. 1, pp. 10–24, 2021.

[17] M. Tavallaee, N. Stakhanova, and A. A. Ghorbani, "Toward credible evaluation of anomaly-based intrusion-detection methods," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 5, pp. 516–524, 2010.

[18] X. Tian, Q. Zheng, and N. Jiang, "An abnormal behavior detection method leveraging multi-modal data fusion and deep mining," *IAENG International Journal of Applied Mathematics*, vol. 51, no. 1, pp. 92–99, 2021.

[19] L. M. Raposo, M. B. Arruda, R. M. de Brindeiro, and F. F. Nobre, "Lopinavir resistance classification with imbalanced data using probabilistic neural networks," *Journal of medical systems*, vol. 40, no. 69, pp. 1–7, 2016.

[20] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.

[21] Z. Sun, Q. Song, and X. Zhu, "Using coding-based ensemble learning to improve software defect prediction," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1806–1817, 2012.

[22] M. Amrehn, F. Mualla, E. Angelopoulou, S. Steidl, and A. Maier, "The random forest classifier in weka: Discussion and new developments for imbalanced data," *arXiv preprint arXiv:1812.08102*, pp. 1–6, 2018.

[23] T. T. Khuat and M. H. Le, "Evaluation of sampling-based ensembles of classifiers on imbalanced data for software defect prediction problems," *SN Computer Science*, vol. 1, pp. 1–16, 2020.

[24] P. K. Simpson, "Fuzzy min-max neural networks. i. classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 776–786, 1992.

[25] ——, "Fuzzy min-max neural networks - part 2: Clustering," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 1, p. 32, 1993.

[26] T. T. Khuat, F. Chen, and B. Gabrys, "An improved online learning algorithm for general fuzzy min-max neural network," in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–9.

[27] D. Bertsimas and J. Dunn, "Optimal classification trees," *Machine Learning*, vol. 106, no. 7, pp. 1039–1082, 2017.

[28] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[29] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[30] B. Gabrys, "Combining neuro-fuzzy classifiers for improved generalisation and reliability," in *Proceedings of the 2002 International Joint Conference on Neural Networks*, 2002, pp. 2410–2415.