Heterogeneous oblique random forest

Katuwal, Rakesh; Suganthan, Ponnuthurai Nagarathnam; Zhang, Le

2019

Katuwal, R, Suganthan, P. N., & Zhang, Le. (2019). Heterogeneous oblique random forest. Pattern Recognition, 99, 107078-. doi:10.1016/j.patcog.2019.107078

<https://hdl.handle.net/10356/138843>

<https://doi.org/10.1016/j.patcog.2019.107078>

© 2019 Elsevier Ltd. All rights reserved. This paper was published in Pattern Recognition and is made available with permission of Elsevier Ltd.

Downloaded on 26 Apr 2023 17:10:27 SGT

Heterogeneous Oblique Random Forest

Rakesh Katuwal^a, P.N. Suganthan^{a,*}, Le Zhang^b

^a*School of Electrical and Electronic Engineering, Nanyang Technological University,
Singapore 639798, Singapore*

^b*Institute for Infocomm Research, Agency for Science, Technology and Research
(A*STAR), 1 Fusionopolis Way, #21-01 Connexis (South Tower), Singapore 138632*

Abstract

Decision trees in random forests use a single feature in non-leaf nodes to split the data. Such splitting results in axis-parallel decision boundaries which may fail to exploit the geometric structure in the data. In oblique decision trees, an oblique hyperplane is employed instead of an axis-parallel hyperplane. Trees with such hyperplanes can better exploit the geometric structure to increase the accuracy of the trees and reduce the depth. The present realizations of oblique decision trees do not evaluate many promising oblique splits to select the best. In this paper, we propose a random forest of heterogeneous oblique decision trees that employ several linear classifiers at each non-leaf node on some top ranked partitions which are obtained via one-vs-all and two-hyperclasses based approaches and ranked based on ideal Gini scores and cluster separability. The oblique hyperplane that optimizes the impurity criterion is then selected as the splitting hyperplane for that node. We benchmark 190 classifiers on 121 UCI datasets. The results show that the oblique random forests proposed in this paper are the top 3 ranked classifiers with the heterogeneous oblique random forest being statistically better than all 189 classifiers in the literature.

Keywords: Benchmarking, classifiers, oblique random forest, heterogeneous, one-vs-all, ensemble learning

---

*Corresponding author

Email addresses: rakeshku001@e.ntu.edu.sg (Rakesh Katuwal), epnsugan@ntu.edu.sg (P.N. Suganthan), zhangleuestc@gmail.com (Le Zhang)

1. Introduction

Random Forest (RaF) [1] is a popular learning algorithm employed in variety of tasks including image classification [2, 3], object recognition and detection [4, 5], action recognition and detection [6, 7], semantic segmentation [8, 9], visual tracking [10], facial expression recognition [11], regression problems [12, 13], time-series forecasting [14], label distribution learning [15] and so on. RaF is an ensemble of decision trees with many appealing characteristics such as inherent multi-class handling capacity, robustness to label noise, fewer tunable parameters, parallel processing, enables variable (feature) selection and the ability to handle categorical as well as numerical data. Although random forest was proposed two decades ago, because of its simplicity in implementation, interpretability and state-of-the art performance [16, 17, 18], it is still widely used in many applications [19, 20].

Decision trees in a random forest employ recursive partitioning of the training data into smaller subsets that further aid in classification by optimizing an impurity criterion such as information gain or gini index [21]. In this paper, we focus on binary trees which partition the data into two partitions (left and right partitions, also known as left and right child nodes) that generates an axis-parallel or orthogonal hyperplane at each node. For example, if $d_5 \geq 0.375$, the data sample is sent to the left otherwise right where d_5 is the 5th feature of the sample and 0.375 is the threshold. That means, at this node, the 5th feature of the sample is compared with the threshold of 0.375 and the sample is either sent to left or right partition or child nodes. Such a rule generates a line parallel to one of the feature axes. Such hyperplanes may not always approximate complex decision boundaries [10, 22, 23, 24]. In a variant of random forest, known as oblique random forest (obRaF) [25], an oblique linear hyperplane is used at each node. Such decision boundary uses a linear combination of features to split the training data. The decision boundaries generated by each type of tree is shown in Fig. 1.

A candidate hyperplane at each internal node of the decision tree is usually

rated based on an impurity criterion. Given a set of training samples and a hyperplane at each node, it is straightforward to identify the set of training samples that fall in left and right child nodes of this parent node. If each of these child nodes have predominance of one class over others, then the hyperplane can be
 35 considered to have contributed positively to further classification [24]. In other words, if the class distribution is nearly uniform, the node is highly impure and if the number of samples of one class is much larger than that of all others, the purity of this node is high. The impurity criterion gives higher rating to a hyperplane which results in higher purity of child nodes. However, impurity criterion
 40 commonly used in decision trees such as gini index, information gain and towing rule are not differentiable with respect to hyperplane parameters [26]. So, decision trees use some greedy search techniques for finding the best hyperplane at each node. The decision trees in random forest exhaustively search for a single feature or the best axis-parallel hyperplane. However, such exhaustive search
 45 for the best oblique hyperplane is computationally expensive [22]. Thus, linear hyperplanes are used in oblique decision trees without optimizing any criteria or the search for the oblique splits are generally based on heuristic approaches and are non-optimal. To circumvent such an issue, we present oblique decision trees that search for an optimal linear hyperplane at each node from a finite pool
 50 of competitive hyperplanes by optimizing the Gini-impurity criterion similar to the trees in RaF.

From extensive applications of random forests in many diverse domains, it is evident that the recursive partitioning of the training data while optimizing some impurity criterion aid random forest to generalize better and by the virtue
 55 of subspace and ensemble method, RaF is able to achieve state-of-the-art performances. As reported in [26], employing linear classifiers in each internal node can lead to both “accurate” and “diverse” decision trees which is of vital importance for the success of random forests [1, 27]. This is further verified by the large-scale benchmarking with state-of-the-art classifiers in [16, 17]. However,
 60 employing a single type of linear classifier may not always be optimal due to the No-Free-Lunch theorem [28]. Such a limitation can be curbed by employing

Figure 1: Classification boundaries generated by axis-parallel and oblique decision tree in a toy example for binary classification problem. Axis-parallel or staircase decision boundaries may fail to approximate the optimal decision boundary. However, oblique trees with linear hyperplanes can better approximate the complex decision boundary. Best viewed in color.

a simple strategy: selecting one best linear classifier from a pool of diversified linear classifiers. Depending on the dataset or subset of training samples at each node, one type of hyperplane may be best suited for partitioning the samples
65 at that particular node. That means, such hyperplane can best optimize the impurity criterion compared to the others. Hence, selecting the one from a pool using an impurity criterion endows the random forest to generalize better with more “accurate” and “diversified” tree structure. In this study, we employ six linear classifiers namely Support Vector Machines (SVM), Multisurface Proxi-
70 mal SVM (MPSVM), Linear Discriminant Analysis (LDA), Least Squares SVM (LSSVM), Ridge Regression (RR) and Logistic Regression (LR) due to their good performance in several domains [16]. Extension of the proposed method to other linear classifiers is straightforward. The key contributions of this paper are summarized as follows:

- First, we expand the work presented in [29]. In [29], a one-vs-all based oblique random forest is proposed. Specifically, MPSVM is employed as a linear classifier to separate one class from all other classes at each node

using the one-vs-all data partitioning technique. In this paper, we extend the proposed idea using five other linear classifiers.

- 80
85
90
95
100
105
 • The one-vs-all based formulation requires evaluating K decision boundaries generated by a linear classifier where K is the number of classes at each node. An oblique random forest employing n linear classifiers requires nK evaluations at each node. Such procedure is computationally expensive when the number of classes is very large. Also, such one-vs-all based partitioning results in unbalanced trees while limiting the searches to only K evaluations. In this paper, we expand the search space of the oblique splits by considering both one-vs-all and two-hyperclasses based partitioning. Intuitively, this increases the computational cost. To avoid evaluating a large number of partitions, we use either an optimization algorithm or an exhaustive search depending on the number of classes at each node in conjunction with ideal gini score and cluster separability to selectively filter the partitions and employ linear classifiers only on the top ranked partitions.
- We then, propose a heterogeneous oblique random forest that searches for the best splitting hyperplane at each node by optimizing the impurity criteria as in standard random forest. Unlike conventional oblique random forest that employs only one type of linear classifier for a splitting hyperplane at every node, the heterogeneous oblique random forest employs diverse linear classifiers throughout the nodes of the trees that can better exploit the geometric structure in the data. The proposed approach is generic and any linear classifier can be used in our framework.
- The trees in the oblique random forests proposed in this paper are generally shallow and require less training time compared to standard random forests with the exception of the heterogeneous oblique random forest which requires more training time.
- In our experiments on 121 UCI machine learning datasets, we find that 3

(out of 7) of our proposed oblique random forest variants take the top 3 ranks by outperforming standard (axis-parallel) random forest and oblique random forest variants and hundreds of other classifiers [16, 17].

110 The rest of the paper is organized as follows: we present a brief review of the related works in the following section. In Section 3, we delineate our approach for building heterogeneous oblique random forests. In Section 4, we present experimental results and compare with different forest based methods and other state-of-the-art classifiers. Finally, we present our conclusions in Section 5.

115 2. Related Works

2.1. Multi-class classification via “One-vs-all” approach

In a multi-class classification problem, the training set consists of data samples belonging to C different classes. In “One-vs-all” (also known as OVA) approach, C different binary classifiers are constructed where each one is trained to
 120 distinguish the examples in a single class (positive samples) from the examples in all other remaining classes (negative samples). To classify a new example, the C classifiers are run, and the one that outputs the highest confidence value is chosen. There exist numerous researches in the literature that compare the performance of OVA with other techniques such as “One-vs-one” (also known
 125 as OVO), and DAGSVM [30] in different scenarios. Some of the notable ones are presented in [31, 32]. Based on the experiments on those paper, it is difficult to explicitly state which scheme is the best and one can only comprehend that the best scheme depends largely on the problem.

2.2. Genetic Algorithm

130 Genetic Algorithm (GA) is a computational method used to solve optimization and search problems by mimicking biological evolution via bio-inspired operators such as mutation, crossover and selection [33]. In GA, a population of candidate solutions to a problem is evolved towards better solutions at each

iteration. The population in each iteration is called a generation. Each indi-
135 vidual solution is normally composed of strings of ones and zeros that can be
mutated and altered. The evolution usually starts from a population of ran-
domly generated individuals. In each generation, the fitness (objective function
of the optimization problem) of every individual solution in the population is
evaluated. The more fit individuals (also known as parents) are selected from
140 the current population and are used to produce the children via crossover and
mutation for the next generation. The new generation of candidate solutions
is then used in the next iteration of the algorithm. The algorithm terminates
when either a maximum number of generations have been produced, or a sat-
isfactory fitness level has been reached for the population. For more details on
145 GA, please refer to [33].

2.3. Random Forest (RaF)

A random forest is an ensemble of T axis-parallel decision trees that are
trained independently. In RaF, each non-leaf node is associated with a split
function $f(\mathbf{x}; \Theta)$ where

$$f(\mathbf{x}; \Theta) = \begin{cases} 1 & \text{if } \mathbf{x}(\Theta_1) < \Theta_2 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $\Theta_1 \in \{1, 2, \dots, d\}$ is the selected feature and $\Theta_2 \in \mathbb{R}$ is a threshold.
The outcome determines the child node to which $\mathbf{x}$ is routed. For instance, 0
may represent left child node while the right child node by 1. The leaf nodes of
the tree either store class probability distributions or class labels based on the
training samples they receive. During testing, for a test sample $\mathbf{x}$, each tree
returns a probability distribution $p_t(y|\mathbf{x})$ stored on the leaf node it falls into,
and the class label is obtained via averaging as shown below or by majority
voting.

$$y^*(\mathbf{x}) = \arg \max_y \frac{1}{T} \sum_{t=1}^T p_t(y|\mathbf{x}) \quad (2)$$

2.4. Oblique Random Forest (ObRaF)

In an Oblique Random Forest, trees employ a linear combination of features to generate an oblique hyperplane at each node. Thus, Eq. 1 can be reformulated as:

$$f(\mathbf{x}; \Theta) = \begin{cases} 1 & \text{if } \sum_{i=1}^q w_i x_i < \Theta_2 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where w_i is the weight coefficient for each feature in q .

In standard (axis-parallel) decision trees, the optimal split at each node is chosen from the $n \cdot q$ possible splits, where n is the unique feature values in $\mathbf{x}$ and q is typically set to the square root of feature dimension d . That means, in the worst possible scenario the exhaustive search for the ‘optimal’ orthogonal split grows linearly in the number of training samples at each node and q . However, there are at most $2^q \cdot \binom{n}{q}$ distinct oblique splits at each node and the exhaustive search of the ‘optimal’ oblique split is NP-hard [22].

2.5. ObRaF with linear and non-linear hyperplanes

As the search for the optimal oblique hyperplane is cumbersome, many heuristic search methods based ObRaF such as Forest-LC [25], OC1 [22] have been proposed in the literature. Apart from these heuristic approaches, linear classifiers such as SVM, MPSVM, LDA, LR have also been used to generate oblique hyperplanes. In [23, 34], LDA is used to generate linear splits. However, they consider only binary classification problems. In [35, 36], SVM is used to generate a linear split at each node. They either employ kernel clustering algorithm [35] or some distance metrics such as Euclidean/Mahalanobis [36] to break down multi-class classification problems into binary classification problems. [26] employs MPSVM to generate linear split at each node. It uses Bhattacharyya distance to group multiple classes into two hyperclasses. Among the two clustering hyperplanes generated at each node, the one that best optimizes the impurity criterion is selected. However, the hyperplane selection is restricted to just two hyperplanes. In this paper, we increase the search space

for the best linear split and also employ diverse linear classifiers. [37] employs
175 logistic regression in several partitions ($2^{K-1} - 1$ partitions where K is the
number of classes at each node) and selects the one that that best optimizes the
impurity criterion. However, such procedure is computationally expensive for
large number of classes. All of these approaches either do not optimize impurity
180 are computationally expensive. The computational complexity associated with
the linear classifiers impede the search or optimization process. In [38, 39], the
authors propose to compute the weight coefficients of the oblique hyperplane
at each node by continuously optimizing the log loss function of each node by
means of gradient descent. However, several hyper-parameters need to be tuned
185 at each node which greatly increases the computational complexity of the trees
(complexity multiplies when growing the forest) without any significant gain
in the performance. Similarly, in [8], a randomized multi-layer perceptron is
used as a splitting function at each node of the tree. However, selecting the
neural network’s model complexity is an issue here since, using a much more
190 complex model might result in overfitting. Thus, the authors propose to use a
heuristic approach to select the neural network hyper-parameters that depend
on the number of training samples reaching a particular node. This method is
also computationally expensive as it has to train a neural network at each node
using back-propagation and such method may not result in optimal network
195 parameters at the lower nodes where there are only few samples. In [3], the
decision nodes do not perform hard routing as in standard decision trees but in-
stead perform soft (stochastic) routing and the node (linear split) parameters are
obtained by gradient descent based approach by minimizing the global loss func-
tion. Such trees, known as differential decision trees, have also been extended
200 to other learning tasks beyond classification such as label distribution learn-
ing (LDL) [15] and regression [13]. Unlike these works, our proposed approach
integrates an impurity optimization technique for optimal oblique hyperplane
search in ObRaF as in standard RaF without incurring great computational
costs by evaluating numerous partitions before actual hyperplane generation.

205 *2.6. ObRaF with linear hyperplanes based on one-vs-all approach*

In this section, we briefly review the work in [29]. As mentioned in Section 2.1, many popular binary classifiers such as SVM use “one-vs-all” approach to breakdown a multi-class classification problem into several binary classification problems. A caveat is that it may not always be the best method to deal with multi-class problems. Such methods can however, be integrated intactly at the internal nodes of the decision trees. A linear classifier at each node doesn’t need to be a perfect classifier but simply aid in further classification or partitioning. Thus, at each node of the decision tree a linear classifier is employed to separate one class from all other classes. That means, the best linear hyperplane at each node is obtained by searching amongst the ‘ K ’ hyperplanes obtained by transforming a multi-class classification problem into ‘ K ’ binary classification problems¹. Thus, a linear decision boundary can be selected at each node that best optimizes the impurity criterion. In the best scenario, a linear classifier may result in all samples of a class in one child node while the rest of the training samples in the other child node which is exactly the objective of decision trees i.e. to make child nodes purer than the parent nodes. Thus, employing linear hyperplanes with impurity optimization technique can help to better exploit the geometric structure of the data than axis-parallel hyperplanes. This idea was implemented using MPSVM as a linear classifier.

At each node of the tree, K oblique hyperplanes generated using MPSVM were compared and the one that best optimized Eq. 4 was selected as the splitting hyperplane. The optimization problem is given as:

$$\max(Gini_p - Gini_c) \quad (4)$$

where

$$Gini_p = 1 - \sum_{i=1}^K \left(\frac{n_{w_i}}{n_t} \right)^2 \quad (5)$$

---

¹The total number of classes in a dataset is denoted by C and the number of classes at each node by K . $C \geq K$ where $C = K$ at the top root node.

Algorithm 1 Oblique Decision Tree (obDT) Training [29]

```

1: Require: Labeled training data  $(X, Y)$ 
2: Require: Maximum tree depth  $\mathcal{D}$ 
3: Output: Linear decision boundaries ( $w$ 's and  $b$ 's)
4: for  $d_T$  from 1 to  $\mathcal{D}$  do
5:   Check stopping criteria at each node at depth  $d_T$ .
6:   Create  $K$  data partitions based on one-vs-all approach.
7:   Train a linear classifier on each partition.
8:   Compute Eqs. (5) and (6).
9:   return  $w$  and  $b$  of the hyperplane that best optimizes Eq. 4 at each
       node at depth  $d_T$ .
10: end for

```

$$Gini_c = \frac{n_t^l}{n_t} \left[1 - \sum_{i=1}^K \left(\frac{n_{w_i}^l}{n_t^l} \right)^2 \right] + \frac{n_t^r}{n_t} \left[1 - \sum_{i=1}^K \left(\frac{n_{w_i}^r}{n_t^r} \right)^2 \right] \quad (6)$$

225 where $Gini_p$ and $Gini_c$ are the values of Gini impurity at the parent node and child nodes respectively, n_t is the number of data samples in the parent node, n_t^l, n_t^r is the number of data samples that reach the left and right child nodes of the current parent node, and $n_{w_i}^l, n_{w_i}^r$ are the number of samples of class w_i in the left and right child nodes respectively.

230 The construction algorithm of the oblique decision tree is presented in Algorithm 1.

One of the issues with the “one-vs-all” technique is that it is computationally expensive when the number of classes is very large. Thus, generating K hyperplanes at each node can be computationally expensive in such cases. Since 235 the linear hyperplane at each node is selected from a pool of hyperplanes based on the impurity criteria (more specifically Eq. 4), it results in pure child nodes faster than previous exhaustive approaches (in case of RaF) and obRaF without impurity optimization approaches. Thus, the trees in such ‘one-vs-all’ oblique random forest variants are generally shallow than the standard trees [29]. This 240 may negate the complexity associated with generating many hyperplanes at each node and it would also improve the classification testing time.

3. Heterogeneous oblique random forest

Different linear classifiers employ different techniques to generate a linear decision boundary (maximum margin hyperplane in case of SVM, clustering
245 hyperplane in case of MPSVM, etc.). One family of linear classifier may not always create the best partition or exploit the geometric structure of the data as the decision boundary depends on the training samples falling at each node. By using different partitions at each node, we can search for the best linear decision boundary for that node among several linear decision boundaries and
250 such boundaries may be obtained from different families of linear classifiers. Thus, we employ several linear classifiers viz., SVM, MPSVM, LDA, LSSVM, RR and LR with decision trees to construct several variants of one-vs-all based oblique decision trees at first.

Based on our experimental results, we observe that, even though some of the
255 linear classifier based oblique random forests consistently perform better (have high ranks compared to others), they are not always the best oblique random forest variants for every dataset. This is congruent with the famous No-Free-Lunch theorem [28] which states that no single classifier is the best for every dataset. Thus, even though some of the linear classifier based variants have lower
260 ranks compared to others, we can still integrate them to form a heterogeneous linear classifiers based oblique random forest. Based on our original formulation as described in 2.6, this would require evaluating n linear classifiers in K binary partitions thus, requiring nK evaluations at each node. Also, in theory, it is highly desirable to obtain highly balanced trees but in practice, it is extremely
265 difficult to guarantee balanced trees [40]. An innate issue with one-vs-all based decision trees is that it leads to heavily unbalanced trees. Another issue with such one-vs-all based partitioning method is that it limits the search space of the optimal oblique splits to K choices. Even though the exhaustive search of the optimal oblique split is NP-hard, it is beneficial to increase the search
270 space of the oblique splits to obtain the best split. Thus, to address or solve the above issues, we employ a hyperclasses based partitioning along with one-vs-all

partitioning while using several linear classifiers at each node as described in the subsequent sections.

3.1. Generating several partitions and their ranking

275 In standard decision trees, each axis-parallel split is rated based on an impurity criterion. Specifically, all the splits at each non-leaf nodes are associated with an impurity measure (we denote this by $g(i) = Gini_p - Gini_c$ using Eqs. 5 and 6) and the one with the maximum value (refer to Eq. 4) is the selected split ('best' split) at that particular node. However, as mentioned in Section 280 2.4, an exhaustive search for the optimal oblique split is NP-hard. To extenuate an exhaustive search, we restricted the search space to limited oblique splits in [29]. We employed a linear classifier to generate K hyperplanes via K one-vs-all partitions and selected the one based on Eq. 4. In other words, instead of looking for the optimal oblique split in the whole search space, we utilized the recursive partitioning property of decision trees (separating samples of one class 285 from all other classes) to generate few oblique splits and compared their $g(i)$'s for the selection of the best oblique split. We observed that in case of one-vs-all partitions, partitions with higher positive training samples (large number of training samples for the one class) basically had higher $g(i)$ values compared to other partitions ². This is intuitive since separating large number of samples of one class produces purer child nodes (refer to Eqs. 5 and 6). In a nutshell, we can employ the information available at each node to obtain the best split.

295 Generally, in case of oblique decision trees, a multi-class classification problem is treated as binary classification problem at each node. Thus, the purity of child nodes depends on the partitioning of the classes and the quality of the split. For the latter, we employ several linear classifiers as discussed in the subsequent section. For the former, we combine the one-vs-all partitioning idea of [29] and hyperclasses based partitioning approach of [26, 37]. In hyperclasses based partitioning, several classes are grouped in two hyperclasses (say c_1 and

---

²We call the 'one' class as positive class and the 'all' class as the negative class in case of one-vs-all partitioning.

300 c_2) based on a criterion. For example, with 5 classes $\{1,2,3,4,5\}$ at a node, they can be grouped into two hyperclasses $c_1 = \{1,2,5\}$ and $c_2 = \{3,4\}$ based on some criteria. [26] uses the Bhattacharyya distance between the classes to group them into two classes while [37] simply uses all possible choices of hyperclasses.

At each node of a decision tree, there are $2^{K-1} - 1$ possible ways of partition-
 305 ing K classes [37]. For example, if there are 10 classes at a node, these classes can be arranged in $2^{10-1} - 1 = 511$ partitions. For large number of classes, the number of partitions is rampant. Thus, in this paper, we selectively filter the partitions and use only the top ranked partitions to train the linear classifiers. Specifically, we use ideal gini score and clustering validity index to rank the par-
 310 titions. Ranking the partitions enable us to use few significant partitions while avoiding the insignificant partitions (those likely to have small $g(i)$). Before delving into the partitioning approach and ranking, we first introduce the two criteria used to rank the partitions.

3.1.1. Ideal Gini score

315 The ideal gini score refers to the gini value obtained in the ideal case where an oblique split perfectly separates all the samples of one class from the other class. Note that, in this paper, we breakdown a multi-class problem into binary class problem recursively. Thus, in one-vs-all partitioning the ideal case is equivalent to separating all samples of the positive class from the rest of the classes while
 320 in hyperclasses based partitioning, it is equivalent to separating all the samples of one hyperclass from the other. Computing ideal gini score does not require generating a hyperplane but simply computing $g(i)$ assuming the scenario when all the samples of one class (or hyperclass) are sent to the right node and all others to the left node³. A higher gini score means purer child nodes. If a split
 325 with high ideal gini score (g_I) is able to replicate its g_I , it is likely to have higher actual gini score (g). Computing g_I beforehand, without actually training a

---

³We represent the ideal gini score $g_I(i)$ as g_I and actual gini score $g(i)$ as g respectively and interchangeably throughout the paper hereafter. We also use the terms partitions and hyperclasses interchangeably throughout the paper.

linear classifier gives us an indication which partition the oblique split should focus on. By training the linear classifiers on the partitions with higher g_I which are likely to give higher g , we can ignore the partitions with lower g_I which are likely to give lower g .

3.1.2. Cluster separability

g_I is a measure of the purity of the child nodes in an ideal setting with the assumption that the partitions are well-separated and a linear decision boundary can exactly classify or separate them. However, in the real settings, this may not always be true. Thus, we also need to consider the geometric structure of the data at each node when ranking the partitions. A partition may have high g_I but low g if the partitions are difficult or hard to separate. Thus, we use a cluster validity index to evaluate the separability of each partition or cluster.

There are several cluster validity indices in the literature. The cluster validity indices aim to identify the sets of clusters that are compact and well-separated. Since we are interested in binary decision trees or binary partitioning, it is highly desirable that the two partitions are well separated. Dunn index (DI) [41] is one of the popular cluster validity index. The Dunn index for k clusters is defined by:

$$DI_k = \min_{i=1,\dots,k} \left\{ \min_{j=1+i,\dots,k} \left(\frac{diss(c_i, c_j)}{\max_{m=1,\dots,k} diam(c_m)} \right) \right\} \quad (7)$$

where $diss(c_i, c_j) = \min_{x \in c_i, y \in c_j} \|x - y\|$ is the dissimilarity between the clusters c_i and c_j (also known as inter-cluster distance) and $diam(C) = \max_{x, y \in C} \|x - y\|$ is the intra-cluster function (diameter) of the cluster. A higher DI indicates better clustering. We use Bhattacharyya distance as the distance metric as it is considered a good measure for class separability between two classes [26]. The Bhattacharyya distance between two classes c_1 and c_2 with means and co-variances (μ_1, Σ_1) and (μ_2, Σ_2) respectively, is defined as:

$$B(c_1, c_2) = \frac{1}{8}(\mu_2 - \mu_1)^T \left(\frac{\Sigma_1 + \Sigma_2}{2} \right)^{-1} (\mu_2 - \mu_1) + \frac{1}{2} \ln \frac{|\Sigma_1 + \Sigma_2|/2}{\sqrt{|\Sigma_1| |\Sigma_2|}} \quad (8)$$

Since we only consider binary partitions and their separability, the two partitions need to be well separated but not necessarily compact, thus, we can ignore the denominator part of DI i.e. the diameter of the cluster. The inter-cluster distance is the distance between the two closest classes, one in each partition. Because of the recursive partitioning in decision trees, it is highly likely to have only a few samples of some classes at a node. Typically in a one-vs-all based partitioning, the separability of the partitions is affected when the class (with only a few samples) belonging to the negative class is very close to the positive class. This plummets the separability of the partitions even when the partitions are well-separated barring the few samples. In such a scenario, the DI associated with such partition is low even when g_I or g can be high. To alleviate such issue, we take the weighted mean of the distances of the closest and the second closest classes of the negative class with the positive class while computing the inter-cluster distance.

The DI criterion complements the g_I criteria. That means, well separated partitions with higher g_I are likely to have higher g or purer child nodes compared to other partitions. We demonstrate this using a toy example as shown in Figure 2. The toy example consists of 4 classes $\{1,2,3,4\}$ at a node. The samples of each class are normally distributed with different means for each class. With 4 classes, there are 7 ($2^{4-1} - 1$) ways of partitioning them as shown by the seven plots. For each partition, we compute the g_I and DI and sort them based on their final rank (obtained via averaging the two ranks). We then train an LDA to find a linear decision boundary to separate the partitions and compute g afterwards. Based on Figure 2, we can see that the top ranked partitions (with higher g_I and DI) actually have higher g . On close observations, we can also see that even though the partition $\{1,3\}\{2,4\}$ has higher g_I than the partition $\{4\}\{1,2,3\}$ (see Figure 2(b) and Figure 2(c)), the partition $\{4\}\{1,2,3\}$ is well-separated than the partition $\{1,3\}\{2,4\}$. On averaging the two ranks obtained via g_I and DI, the partition $\{4\}\{1,2,3\}$ is ranked higher than the partition $\{1,3\}\{2,4\}$. On computing g after obtaining the decision boundary, we can see that the partition $\{4\}\{1,2,3\}$ has higher g than partition $\{1,3\}\{2,4\}$. Similar

370 explanation pertains to Figure 2(e) and Figure 2(f). This shows that we can sort the partitions by averaging the ranks obtained via g_I and DI criteria and is useful to filter out the insignificant partitions as discussed in the next section.

3.1.3. *Employing Exhaustive search or Genetic Algorithm*

Depending on the number of classes at each node, the partitions are obtained
 375 through either Genetic Algorithm (GA) or exhaustive search. We evaluate an equal number of one-vs-all partitions and hyperclasses based partition at each node. For example, if there are K one-vs-all partitions, we obtain another top K hyperclasses based partition from GA or exhaustive search and re-rank these $2K$ partitions. We do this because for K classes, there are $2^{K-1} - 1 - K$
 380 possible hyperclasses based partitions. For large number of classes, the number of hyperclasses based partition drastically increases. For 11 classes, there are 1012 hyperclasses based binary partitions and for 100 classes, there are 6.33×10^{29} possible partitions! To avoid evaluating a colossal number of partitions, we employ the idea of [26]. In [26], several classes at each node are grouped into
 385 two hyperclasses based on the pairwise Bhattacharyya distance between the classes. Such partition may have a higher DI but not necessarily higher g_I . Thus, we introduce a slight variation in this approach. Specifically, keeping the two furthest classes obtained from the pairwise Bhattacharyya distance approach fixed, we randomly assign 50% (value used in our paper) of the classes mid-
 390 way between the two furthest classes to either of the partitions. We rank all these partitions based on g_I and DI and select the top K hyperclasses based partitions. When $K \geq 8$, we employ GA to select these partitions as shown in Algorithm 2 and Figure 3.

3.2. *Employing several linear classifiers at each node*

395 Employing a single type of linear classifier at a node may not yield the best partition. Such a limitation can however, be remedied by applying several diverse linear classifiers. Depending on the training samples at each node, one type of hyperplane (from many) may be best suited for partitioning the sam-

(a) Rank 1, $DI = 18.99$, $g_I = 0.344$, $g = 0.344$

(b) Rank 2, $DI = 27.95$, $g_I = 0.291$, $g = 0.291$

(c) Rank 3, $DI = 14.51$, $g_I = 0.308$, $g = 0.272$

(d) Rank 4, $DI = 9.3$, $g_I = 0.281$, $g = 0.256$

(e) Rank 5, $DI = 11.49$, $g_I = 0.114$, $g = 0.11$

(f) Rank 6, $DI = 6.01$, $g_I = 0.215$, $g = 0.106$

(g) Rank 7, $DI = 8.39$, $g_I = 0.213$, $g = 0.067$

Figure 2: Ranking of different partitions based on their ideal gini values g_I and dunn index DI . LDA is used at each partition to generate a decision boundary. g_I refers to the ideal gini scores computed beforehand to rank the partitions while g refers to the actual gini scores after generating the decision boundary. The numbers within the curly braces represent the binary partitions. db means decision boundary. Best viewed in color.

Figure 3: Flowchart explaining GAOREXHSEARCH function of Algorithm 2.

Figure 4: For the GA based partitioning, we create the initial population by randomly assigning the classes inside the rectangular box (represents the 50% classes) mid-way between the two furthest classes ext1 and ext2 obtained using Bhattacharyya distance to either of the partitions.

ples at that particular node. We employ several diverse linear classifiers at each
 400 node resulting in heterogeneous oblique decision tree. To elucidate the motiva-
 tion behind using several linear classifiers, we use one of the partitions of the
 toy example described above. Specifically, we use the third ranked partition
 $\{1,3\}\{2,4\}$ and employ three different linear classifiers LDA, MPSVM and LR
 to generate three decision boundaries. As can be seen in Figure 5, all three
 405 classifiers generate different decision boundaries to separate the partition with
 the decision boundary generated by LR having the highest g . Since the selec-
 tion of the split is based on the one with the highest g , the decision boundary
 generated by LR is selected as the best oblique split for that particular node.

As discussed in previous sections, even for 10 classes at a node, the total
 410 number of possible partitions is 511. Using all 6 classifiers (in our case) would

Figure 5: Decision boundaries generated by different linear classifiers and their associated g . Best viewed in color.

mean generating 3066 different decision boundaries at that particular node and comparing them. This count increases with the increase in the number of classes, tree depth and the number of linear classifiers used. To eschew this astronomical computational burden, we only employ linear classifiers in some selective
415 partitions based on their ranking. Even though it is compelling to use all linear classifiers only at the top ranked partition to reduce the computational requirements, it is intuitive to select some few top ranked partitions to make the algorithm robust.

The overall algorithm and flowchart for constructing a heterogeneous oblique
420 decision tree is presented in Algorithm 2 and Figure 6 respectively.

4. Experiments

4.1. Machine Learning Datasets

We compare the classifiers on 121 datasets from the UCI repository as done in [17, 16]. Among 121, 66 are multi-class datasets while remaining 55 are binary
425 datasets. Details of the datasets are presented in the Supplementary file.

Algorithm 2 Heterogeneous Oblique Decision Tree Training

```
1: Require: Labelled training data  $(X, Y)$ 
2: Require: Maximum tree depth  $\mathcal{D}$ 
3: for  $d_T$  from 1 to  $D_{max}$  do
4:   Check the stopping criteria for all nodes at depth  $d_T$ .
5:   if  $K \geq 5$  at a node then
6:     Use GAOREXHSEARCH function to get the  $K$  hyperclasses based
7:     partitions and also create  $K$  one-vs-all partitions.
8:     Sort the  $2K$  partitions obtained in line 6 using their average ranks
9:     obtained via both  $g_I$  and DI.
10:    Repeatedly divide the sorted partitions into halves and train the
11:    linear classifiers subsequently on the upper half to get several linear
12:    decision boundaries and compute  $g$ . (refer to Section 4.2.1)
13:    If the number of partitions in the upper half  $\leq 5$ , train remaining
14:    linear classifiers on all remaining partitions.
15:  else
16:    Train all linear classifiers on all possible partitions and compute  $g$ .
17:  end if
18:  Select the decision boundary with the highest  $g$  as the splitting hyper-
19:  plane at depth  $d_T$ .
20: end for
```

```
1: function GAOREXHSEARCH ▷ refer to Sections 3.1.3 and 4.2.1
2:   if  $K < 8$  then ▷ Employ Exhaustive Search
3:     Generate all possible hyperclasses based partitions.
4:     Sort all the partitions obtained in line 3 based on their average
5:     ranks obtained via  $g_I$  and DI.
6:   else ▷ Employ Genetic Algorithm
7:     Compute the pairwise Bhattacharyya distance between the classes.
8:     Find the two classes with largest distance and label them ext1 and
9:     ext2 respectively.
10:    Assign every remaining classes to ext1 and ext2 depending on their
11:    proximity to these classes.
12:    Create initial population by randomly assigning 50% of the classes
13:    mid-way between ext1 and ext2 to either one of the them.
14:    for  $i$  from 1 to  $maxGeneration$  do
15:      Employ GA and sort all the partitions at each generation using
16:      their average ranks.
17:    end for
18:  end if
19:  return the top-ranked  $K$  partitions
20: end function
```

Figure 6: Flowchart depicting our proposed method at each node

4.2. Experimental Settings

For random forest based methods, the number of trees is set to 500. Only the number of features randomly selected at each node, $mtry$, is tuned as in [17, 16]. For linear classifiers such as SVM, RR and LSSVM there is an additional regularization term whose value is randomly selected from the ranges specified in [16]. Specifically, the parameter C of SVM is chosen from the interval $2^{-2}, \dots, 2^7$, λ of RR from $10^{-5}, \dots, 10^5$ and γ of LSSVM from $2^{-5}, \dots, 2^5$. Based on our experiments, tuning the regularization term at each node is computationally expensive as there exist hundreds and thousands of nodes in a single decision tree and there is no significant performance gain. Because decision trees perform recursive partitioning and the trees are fully grown until pure child nodes are obtained, random selection approach is suitable for such methods [17, 42]. We also observed that this approach fosters diversity within the trees (due to hyper-parameter randomization) which is an important criterion for the construction of ensemble methods.

4.2.1. Configuration of Heterogeneous oDT

For the heterogeneous oblique decision tree, we employ all linear classifiers used in our paper in the order of LDA, MPSVM, LSSVM, LR, SVM and RR. These classifiers are ordered based on their performance. These ordered classifiers are subsequently used (beginning with LDA and followed by the rest) in the top ranked half obtained after repeated division of the sorted partitions (refer to line 8 of Algorithm 2). For the GA based partitioning, the population size is set to $2 \times K$ while maximum generation is set to 5. We empirically found that setting these values leads to satisfactory results.

4.3. Performance Comparison

Table A1 in the Appendix reports the classification accuracies of each random forest based classifier on each dataset. From the table, we can see that the oblique random forests with linear decision boundaries generalize better than the random forest with axis-parallel boundaries. Also, the number of wins is distributed across various oblique random forest variants which motivated us to construct the heterogeneous oblique random forest. We follow the procedure of [16, 17], and use the rank of each classifier to depict its performance. In Table 1, we present the mean ranks of the classifiers over all 121 UCI datasets. For binary datasets, ObRaF and obRaF(M) are the same. Thus, when computing the mean rank we use the same accuracy values for these methods and both are assigned average ties. As can be seen, the heterogeneous oblique random forest variant, obRaF(H) obtains the best rank followed by obRaF(L) and obRaF(M). We also perform a statistical comparison of the algorithms using the Friedman test [43] (refer to the Supplementary file for details) .

Based on simple calculations we obtain, $F_F = 11.23$. With 9 classifiers and 121 datasets, F_F is distributed according to the F -distribution with $9 - 1 = 8$ and $(9 - 1)(121 - 1) = 960$ degrees of freedom. The critical value for $F_{(8,960)}$ for $\alpha = 0.05$ is 1.94 , so we reject the null-hypothesis. Based on the Nemenyi test, the critical difference is $CD = q_\alpha \sqrt{(n(n+1))/(6N)} = 3.102 * \sqrt{9 * 10 / (6 * 121)} \simeq 1.09$. From Fig. 7, we can see that obRaF(H) is statistically

Table 1: Average Friedman Rank Based on Classification Accuracies of Each Method on 121 UCI Datasets

	RaF	obRaF[17]	obRaF(L)	obRaF(RR)	obRaF(S)	obRaF(LS)	obRaF(LR)	obRaF(M)	obRaF(H)
Rank	5.4	5.07	4.28	6.13	5.65	5.14	5.46	4.43	3.44

L, RR, S, LS, LR, M and H in the brackets along with obRaF refer to our proposed LDA, Ridge Regression, SVM, LSSVM, LR, MPSVM and heterogeneous oblique random forests respectively. Lower rank reflects better performance.

better than standard random forest and oblique random forest.

Figure 7: Statistical comparison of classifiers against each other based on Nemenyi test. Groups of classifiers that are not significantly different (at $\alpha = 0.05$) are connected.

We also compare the classifiers using Diversity-Error diagrams [44] in the Supplementary file. Our oblique random forests are more diverse and accurate than other random forest based methods. This explains the better performance of our proposed methods compared to other random forests.

4.4. Computational Complexity Analysis

Here, we analyze the computational limitations of the oblique decision trees proposed in this paper. As the computational time required to grow an oblique tree is highly variable, we instead focus on the complexity of partitioning a single node. Consider a training set $\mathbf{X}$ which is a $N \times d$ matrix with N samples and d features. As each node is associated with different sizes of training data, for the sake of simplicity, here we consider the complexity at the root node. Let K denote the number of classes at the root node. The number of splits that need to be evaluated in the one-vs-all based oblique decision trees is upper bounded by K . The complexity of the linear classifiers used in this paper is presented in Table 2. Regardless of the computational time for calculating

the Gini scores, the complexities associated with each type of oblique decision trees at the root node are $\mathcal{O}(Nd^2 \cdot K)$ for obRaF(L), obRaF(RR), obRaF(LS), obRaF(LR), $\mathcal{O}(N^2d \cdot K)$ for obRaF(S), and $\mathcal{O}(d^3 \cdot K)$ for obRaF(M) respectively.

490 As the heterogeneous decision tree employs all the linear classifiers several times, the overall computational complexity for obRaF(H) is the aggregate of all the complexities associated with each linear classifier. In the Supplementary file, we present the training time and tree depth of each method.

Table 2: Computational Complexity of the linear classifiers

Algorithm	Complexity
LDA	$\mathcal{O}(Nd^2)$ [45]
RR	$\mathcal{O}(Nd^2)$
SVM	$\mathcal{O}(N^2d)$ [45]
LS-SVM	$\mathcal{O}(Nd^2)$ [46]
LR	$\mathcal{O}(Nd^2)$ [45]
MPSVM	$\mathcal{O}(d^3)$ [47]

The complexity of an iterative algorithm such as LR is analyzed for one iteration, and so their actual running time may be slower.

4.5. Overall Comparisons Among 190 Classifiers

495 In this section, we compare the performance of the oblique random forests introduced in this paper with the 183 classifiers on 121 UCI datasets as in [16, 17]. Thus, including seven oblique random forest variants proposed in this paper, there are altogether 190 classifiers. However, as discussed in [48], there are some flaws in the experimental procedures in [16]. We discuss the errors/issues in [16] in detail in the Supplementary file and perform additional
500 experiments and analysis using the methodologies described in [48]. For brevity, we present the overall 20 best performing classifiers in Table 3. The details of the relevant classifiers are presented in the Supplementary file. Although the No-Free-Lunch theorem says that no single classifier is always the best, it is
505 intuitive to implement the most robust classifier that generalizes better in most

Table 3: Overall comparison of 20 top ranked classifiers out of 190 methods in 121 datasets

Method	Rank	Friedman Rank	Mean Accuracy
obRaF(H)*	1	29.22	83.39
obRaF(L)*	2	34.28	82.64
obRaF(M)*	3	35.28	82.3
MPRaF	4	36.73	82.05
CoKRR-max	5	37.84	81.79
MPRRoF	6	38.14	82.35
parRaF_t	7	38.6	81.96
RaF_t	8	38.73	82.3
obRaF(LS)*	9	39.37	82.19
obRaF(S)*	10	39.92	82.19
obRaF(LR)*	11	40.33	81.95
svm_C	12	42.29	81.77
svmPoly_t	13	44.57	80.31
KRR	14	45.55	81.52
rForest_R	15	45.65	81.94
svmRadialCost_t	16	46.32	80.54
obRaF(RR)*	17	46.83	81.51
svmRadial_t	18	48.98	80.17
RVFL ensemble	19	49.27	80.94
C5.0_t	20	49.38	80.56

“*” indicates the methods introduced in this work. We follow the naming conventions of [16, 17]. The details of the classifiers are presented in the Supplementary file. This comparison is based on the performance of the classifiers stated in [16, 17].

datasets. Notably, the top three ranked methods are oblique random forest variants introduced in this paper.

4.5.1. Statistical Comparison

We use a Wilcoxon signed-rank test to determine the significance among
510 each pair of the top 20 ranked classifiers as suggested in [49]. As discussed in [48], some of the benchmarked datasets fail to run for some classifiers. In such cases, missing values were imputed by setting the accuracies for benchmarks with errors to 82% in [16]. However, this biases the results and [48] performs the statistical comparison by removing the benchmarked datasets that fail to
515 run for any top 20 ranked classifier. We follow similar procedure as in [48] and report the comparison results in Tables 4 and 5. One can see from the tables that obRaF(H) is statistically better than all other top 19 ranked classifiers. Thus, obRaF(H) could be considered as another competitive baseline for the practitioners.

520 5. Conclusion

In this paper, we first proposed several variants of oblique random forest employing several linear classifiers at the internal nodes of the tree. The oblique splits were generated based on one-vs-all partitioning. We also proposed a heterogeneous variant employing diverse linear classifiers trained on some selective top ranked partitions. The partitions were created using both one-vs-all and two-hyperclasses based approaches and ranked using the ideal gini scores and cluster separability. We showed that these two criteria can be used to rank the partitions. The proposed oblique random forests are more accurate and diverse than other random forest methods. The oblique random forests also lead to smoother decision surfaces than the axis-parallel equivalent. This work also expands the comparison of classifiers as done in [16, 17] on 121 UCI datasets and we hope that the top ranked classifiers would be considered as new baselines for the practitioners. Besides classification, such variants of oblique random forest can also be applied in other domains such as regression and time series forecasting. The oblique decision trees can also be used with gradient boosting. Specifically, the integration of the oblique decision trees with popular decision tree boosting frameworks such as XGBoost [50] and CatBoost [51], application in class imbalance learning problem [52], image retrieval [53], incremental learning [7] require further endeavours and are left as future works.

540 References

- [1] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32. doi: 10.1023/A:1010933404324.
- [2] M. Ristin, M. Guillaumin, J. Gall, L. V. Gool, Incremental learning of random forests for large-scale image classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 38 (3) (2016) 490–503. doi: 10.1109/TPAMI.2015.2459678.

Table 4: Statistical comparison of obRaF(H) and each of the other top 20 ranked classifiers

Method	Rank	Wilcoxon signed-rank p, excluding errors
obRaF(L)*	2	2e-04
obRaF(M)*	3	4e-04
MPRaF	4	4e-05
CoKRR-max	5	2e-03
MPRRoF	6	1e-04
parRaF_t	7	4e-04
RaF_t	8	2e-04
obRaF(LS)*	9	1e-06
obRaF(S)*	10	1e-08
obRaF(LR)*	11	1e-08
svm_C	12	8e-04
svmPoly_t	13	3e-05
KRR	14	1e-03
rForest_R	15	4e-06
svmRadialCost_t	16	6e-04
obRaF(RR)*	17	1e-09
svmRadial_t	18	4e-07
RVFL ensemble	19	3e-07
C5.0_t	20	3e-07

The statistical comparison is done using a Wilcoxon signed-rank test. The test is performed by removing the datasets which failed to run for any one of the top 20 ranked classifiers instead of incorrectly imputing them [48] (refer to the Supplementary file for more details).

- [3] P. Kotschieder, M. Fiterau, A. Criminisi, S. R. Bul, Deep neural decision forests, in: 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1467–1475. doi:10.1109/ICCV.2015.172.
- 550 [4] J. Gall, V. Lempitsky, Class-specific hough forests for object detection, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1022–1029. doi:10.1109/CVPR.2009.5206740.
- [5] S. Schuster, P. Wohlhart, C. Leistner, A. Saffari, P. M. Roth, H. Bischof, Alternating decision forests, in: 2013 IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 508–515. doi:10.1109/CVPR.2013.72.
- 555 [6] G. Garcia-Hernando, T. K. Kim, Transition forests: Learning discriminative temporal transitions for action recognition and detection, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 407–415. doi:10.1109/CVPR.2017.51.

Table 5: Wilcoxon signed-rank test based statistical comparison of the 20 top ranked classifiers out of 190 Methods

	obRaF(H)	obRaF(L)	obRaF(M)	MPRaF	CoKRR-max	MPRRoF	parRaF_t	RaF_t	obRaF(LS)	obRaF(S)	obRaF(LR)	svm_C	svmPoly_t	KRR	rForest_R	svmRadialCost_t	obRaF(RR)	svmRadial_t	RVFL ensemble	C5.0
obRaF(H)		+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
obRaF(L)	-									+	+		+		+		+	+	+	+
obRaF(M)	-									+	+				+		+	+	+	+
MPRaF	-																			+
CoKRR-max	-																			
MPRRoF	-																			+
parRaF_t	-																			+
RaF_t	-																			+
obRaF(LS)	-																		+	
obRaF(S)	-	-	-																	
obRaF(LR)	-	-	-																	
svm_C	-																			
svmPoly_t	-	-																		
KRR	-																			
rForest_R	-	-	-																	
svmRadialCost_t	-																			
obRaF(RR)	-	-	-																	
svmRadial_t	-	-	-																	
RVFL ensemble	-	-	-		-				-											
C5.0_t	-	-	-	-		-	-	-												

+ means the method in the corresponding row is statistically better than the method in the corresponding column. Similarly, - indicates that the method in the corresponding row is statistically worse than the method in the corresponding column. The test is performed by removing the datasets which failed to run for the top 20 ranked classifiers instead of incorrectly imputing them [48] (refer to the Supplementary file for more details).

- 560 [7] C. Hu, Y. Chen, L. Hu, X. Peng, A novel random forests based class incremental learning method for activity recognition, Pattern Recognition Volume 78 (2018) 277 – 290. doi:10.1016/j.patcog.2018.01.025.
- [8] S. Bul, P. Kotschieder, Neural decision forests for semantic image labelling, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 81–88. doi:10.1109/CVPR.2014.18.
- 565 [9] B. Fröhlich, E. Rodner, J. Denzler, Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach, in: Computer Vision – ACCV 2012, Springer, Berlin, Heidelberg, 2013, pp. 218–231.

- 570 [10] L. Zhang, J. Varadarajan, P. N. Suganthan, N. Ahuja, P. Moulin, Robust visual tracking using oblique random forests, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 5825–5834. doi:10.1109/CVPR.2017.617.
- [11] Y. Liu, X. Yuan, X. Gong, Z. Xie, F. Fang, Z. Luo, Conditional convolution
575 neural network enhanced random forest for facial expression recognition, Pattern Recognition Volume 84 (2018) 251 – 261. doi:10.1016/j.patcog.2018.07.016.
- [12] A. Criminisi, D. Robertson, E. Konukoglu, J. Shotton, S. Pathak, S. White, K. Siddiqui, Regression forests for efficient anatomy detection and localiza-
580 tion in computed tomography scans, Medical Image Analysis 17 (8) (2013) 1293 – 1303. doi:10.1016/j.media.2013.01.001.
- [13] W. Shen, Y. Guo, Y. Wang, K. Zhao, B. Wang, A. L. Yuille, Deep regression forests for age estimation, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
- 585 [14] X. Qiu, L. Zhang, P. N. Suganthan, G. A. Amaratunga, Oblique random forest ensemble via least square estimation for time series forecasting, Information Sciences 420 (2017) 249 – 262. doi:10.1016/j.ins.2017.08.060.
- [15] W. Shen, K. ZHAO, Y. Guo, A. L. Yuille, Label distribution learning forests, in: Advances in Neural Information Processing Systems 30, Curran
590 Associates, Inc., 2017, pp. 834–843.
- [16] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, Journal of Machine Learning Research 15 (2014) 3133–3181.
- [17] L. Zhang, P. N. Suganthan, Benchmarking ensemble classifiers with novel
595 co-trained kernel ridge regression and random vector functional link ensembles [research frontier], IEEE Computational Intelligence Magazine 12 (4) (2017) 61–72. doi:10.1109/MCI.2017.2742867.

- [18] R. Katuwal, P. Suganthan, L. Zhang, An ensemble of decision trees with random vector functional link networks for multi-class classification, *Applied Soft Computing* 70 (2018) 1146 – 1153. doi:10.1016/j.asoc.2017.09.020.
- [19] A. Pinto, S. Pereira, D. Rasteiro, C. A. Silva, Hierarchical brain tumour segmentation using extremely randomized trees, *Pattern Recognition Volume* 82 (2018) 105 – 117. doi:10.1016/j.patcog.2018.05.006.
- [20] H. Cao, S. Bernard, R. Sabourin, L. Heutte, Random forest dissimilarity based multi-view learning for radiomics application, *Pattern Recognition Volume* 88 (2019) 185 – 197. doi:10.1016/j.patcog.2018.11.011.
- [21] A. Criminisi, J. Shotton, E. Konukoglu, et al., Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning, *Foundations and Trends® in Computer Graphics and Vision* 7 (2–3) (2012) 81–227.
- [22] S. K. Murthy, S. Kasif, S. Salzberg, A system for induction of oblique decision trees, *Journal of artificial intelligence research* 2 (1994) 1–32.
- [23] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, F. A. Hamprecht, On oblique random forests, in: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2011, pp. 453–469.
- [24] N. Manwani, P. S. Sastry, Geometric decision tree, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 42 (1) (2012) 181–192. doi:10.1109/TSMCB.2011.2163392.
- [25] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen, *Classification and regression trees*, CRC press, 1984.
- [26] L. Zhang, P. N. Suganthan, Oblique decision tree ensemble via multisurface proximal support vector machine, *IEEE Transactions on Cybernetics* 45 (10) (2015) 2165–2176. doi:10.1109/TCYB.2014.2366468.

- [27] G. Armano, E. Tamponi, Building forests of local trees, *Pattern Recognition Volume 76* (2018) 380 – 390. doi:10.1016/j.patcog.2017.11.017.
- [28] D. H. Wolpert, The lack of a priori distinctions between learning algorithms, *Neural Computation* 8 (7) (1996) 1341–1390. doi:10.1162/neco.1996.8.7.1341.
- [29] R. Katuwal, P. N. Suganthan, Enhancing multi-class classification of random forest using random vector functional neural network and oblique decision surfaces, in: *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8. doi:10.1109/IJCNN.2018.8489738.
- [30] J. C. Platt, N. Cristianini, J. Shawe-Taylor, Large margin dags for multi-class classification, in: *Advances in neural information processing systems*, 2000, pp. 547–553.
- [31] C.-W. Hsu, C.-J. Lin, A comparison of methods for multiclass support vector machines, *IEEE Transactions on Neural Networks* 13 (2) (2002) 415–425. doi:10.1109/72.991427.
- [32] R. Rifkin, A. Klautau, In defense of one-vs-all classification, *Journal of machine learning research* 5 (Jan) (2004) 101–141.
- [33] M. Mitchell, *An introduction to genetic algorithms*, MIT press, 1998.
- [34] T. D. Lemmond, B. Y. Chen, A. O. Hatch, W. G. Hanley, An extended study of the discriminant random forest, in: *Data Mining*, Springer, 2010, pp. 123–146.
- [35] L. Zhang, W.-D. Zhou, T.-T. Su, L.-C. Jiao, Decision tree support vector machine, *International Journal on Artificial Intelligence Tools* 16 (01) (2007) 1–15.
- [36] F. Takahashi, S. Abe, Decision-tree-based multiclass support vector machines, in: *Neural Information Processing, 2002. ICONIP '02. Proceedings*

of the 9th International Conference on, Vol. 3, 2002, pp. 1418–1422 vol.3.
doi:10.1109/ICONIP.2002.1202854.

- [37] A. K. Y. Truong, Fast growing and interpretable oblique trees via logistic regression models, Ph.D. thesis, University of Oxford (2009).
655
- [38] M. Norouzi, M. D. Collins, D. J. Fleet, P. Kohli, Co2 forest: Improved random forest by continuous optimization of oblique splits, arXiv:1506.06155.
- [39] M. Norouzi, M. Collins, M. A. Johnson, D. J. Fleet, P. Kohli, Efficient non-greedy optimization of decision trees, in: Advances in Neural Information Processing Systems, 2015, pp. 1729–1737.
660
- [40] I. Millington, J. Funge, Artificial intelligence for games, CRC Press, 2009.
- [41] J. C. Dunn, Well-separated clusters and optimal fuzzy partitions, Journal of Cybernetics 4 (1) (1974) 95–104. arXiv:10.1080/01969727408546059.
- [42] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Machine Learning 63 (1) (2006) 3–42. doi:10.1007/s10994-006-6226-1.
665
- [43] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (Jan) (2006) 1–30.
- [44] D. D. Margineantu, T. G. Dietterich, Pruning adaptive boosting, in: ICML, Vol. 97, 1997, pp. 211–218.
- [45] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, K. Olukotun, A. Y. Ng, Map-reduce for machine learning on multicore, in: Advances in neural information processing systems, 2007, pp. 281–288.
670
- [46] L. Ding, S. Liao, Approximate model selection for large scale lssvm, in: Asian Conference on Machine Learning, 2011, pp. 165–180.
- [47] O. L. Mangasarian, E. W. Wild, Multisurface proximal support vector machine classification via generalized eigenvalues, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (1) (2006) 69–74. doi:10.1109/TPAMI.2006.17.
675

- [48] M. Wainberg, B. Alipanahi, B. J. Frey, Are random forests truly the best classifiers?, *Journal of Machine Learning Research* 17 (110) (2016) 1–5.
 URL <http://jmlr.org/papers/v17/15-374.html>
- [49] A. Benavoli, G. Corani, F. Mangili, Should we really use post-hoc tests based on mean-ranks?, *The Journal of Machine Learning Research* 17 (1) (2016) 152–161.
- [50] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, ACM, New York, NY, USA, 2016, pp. 785–794. doi:10.1145/2939672.2939785.
- [51] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, A. Gulin, Catboost: unbiased boosting with categorical features, in: *Advances in Neural Information Processing Systems 31*, Curran Associates, Inc., 2018, pp. 6638–6648.
- [52] R. O'Brien, H. Ishwaran, A random forests quantile classifier for class imbalanced data, *Pattern Recognition Volume 90* (2019) 232 – 249. doi:10.1016/j.patcog.2019.01.036.
- [53] M. Zhou, X. Zeng, A. Chen, Deep forest hashing for image retrieval, *Pattern Recognition Volume 95* (2019) 114 – 127. doi:10.1016/j.patcog.2019.06.005.

Appendix

700 Results on 121 UCI datasets

Table A1: Classification Accuracies (%) of Different Random Forest Based Methods

Datasets	RaF	obRaF[17]	obRaF(L)	obRaF(RR)	obRaF(S)	obRaF(LS)	obRaF(LR)	obRaF(M)	obRaF(H)
abalone	65.61	65.49	65.83	65.45	65.35	65.23	65.42	65.95	65.95
ac-inflam	100	100	100	100	100	100	100	100	100
acute-nephritis	100	100	100	100	100	100	100	100	100
adult	86.11	85.96	85.97	85.52	85.58	85.92	85.9	85.96	86.1
annealing	76.38	76.38	76.38	95.98	96.86	76.38	76.38	76.38	96.23
arrhythmia	77.21	76.99	78.1	70.35	72.12	66.15	72.79	77.01	78.68
audiology-std	81.98	82.56	81.4	84.88	84.3	84.88	85.47	81.4	87.21
balance-scale	90.06	90.87	95.19	94.07	93.75	94.71	92.15	94.07	95.99
balloons	87.5	81.25	87.5	75	81.25	93.75	87.5	81.25	93.75
bank	90.22	90.13	89.8	89.54	90.04	90.2	90	90.13	89.87
blood	75.67	77.41	76.07	76.74	75.94	76.47	76.47	77.41	77.41
breast-cancer	75	74.65	74.65	73.59	72.89	76.06	73.94	74.65	74.3
bc-wisc	97.57	97.57	97.43	97.57	97.43	97.57	97.71	97.57	97.86
bc-wisc-diag	96.13	97.36	97.36	95.07	97.18	95.95	97.89	97.36	97.36
bc-wisc-prog	79.08	80.1	80.1	78.57	81.12	79.59	80.61	80.1	80.1
breast-tissue	71.15	72.12	72.12	72.12	75	69.23	69.23	70.19	73.08
car	98.09	98.32	98.15	98.38	98.44	98.5	97.92	98.44	98.78
ctg-10classes	87.29	94.96	88.18	84.7	85.22	86.25	86.06	87.34	85.73
ctg-3classes	95.1	86.91	95.43	92.8	94.54	95.06	94.16	95.15	93.41
chess-krvk	84.93	83.83	84.13	79.02	81.19	85.13	79.28	84.47	88.1
chess-krvkv	99.47	99.53	99.53	99.12	99.12	98.72	99.34	99.53	99.31
congress-voting	61.47	61.47	61.24	61.47	61.7	61.47	61.24	61.47	61.7
conn-bench-sonar	82.21	83.17	84.13	78.85	83.17	82.69	82.21	83.17	81.25
conn-bench-vowel	98.81	99.03	100	99.57	99.78	99.89	99.84	100	100
connect-4	86.87	87.4	87	83.43	82.1	86.8	86.28	87.4	87.4
contrac	53.67	53.4	51.29	51.49	51.29	52.72	52.31	52.79	51.83
credit-approval	88.37	87.65	88.08	86.77	86.34	88.37	86.77	87.65	86.63
cylinder-bands	80.86	80.86	81.25	82.62	78.91	80.27	80.86	80.86	80.08
dermatology	97.53	98.08	98.08	97.53	97.53	97.25	97.53	97.25	97.8
echocardiogram	84.85	86.36	84.85	84.85	84.85	83.33	84.85	86.36	84.09
ecoli	87.8	87.8	87.5	86.1	87.2	88.39	87.5	87.5	87.8
energy-y1	95.44	96.09	96.09	95.96	95.18	96.22	95.44	96.48	96.22
energy-y2	89.97	89.32	90.1	90.1	89.19	90.1	88.8	89.32	90.1
fertility	89	90	89	88	89	89	89	90	88
flags	69.27	68.23	65.1	63.02	55.21	61.98	60.42	66.15	63.02
glass	77.83	74.06	75.47	73.11	73.58	70.75	70.28	75.47	76.95
haberman-survival	74.01	73.03	73.68	69.41	73.68	73.36	73.36	73.03	73.03
hayes-roth	87.5	87.5	89.29	88.39	87.5	86.61	86.61	88.39	88.39
heart-cleveland	57.57	58.55	57.57	57.57	57.24	58.88	59.54	58.22	57.89
heart-hungarian	84.93	83.56	85.27	84.59	83.9	84.59	85.62	83.56	85.49
heart-switzerland	41.13	46.77	50	50	44.35	51.61	45.16	46.77	46.77
heart-va	37	37.5	38	35	34.5	38.5	36	36.5	38.55
hepatitis	82.69	82.69	82.69	85.9	84.62	83.97	82.69	82.69	84.62
hill-valley	49.13	49.42	70.63	48.72	49.09	49.9	49.17	49.42	69.14
horse-colic	62.87	60.29	60.29	60.29	60.29	61.76	60.29	60.29	60.29
ilpd-indian-liver	70.55	73.12	72.77	72.77	72.09	72.09	73.29	73.12	72.77
image-segmentation	14.29	14.52	14.29	28.37	14.31	14.29	15.58	14.29	17.29
ionosphere	92.33	94.02	93.75	94.03	92.9	92.33	93.75	94.02	94.89
iris	95.95	96.62	95.95	95.95	95.95	95.95	95.95	95.95	97.97
led-display	69.4	72.8	69.7	71.5	71.4	71.2	71.6	70.6	71.1
lenses	83.3	83.3	83.33	79.17	79.17	87.5	79.17	83.33	83.33
letter	96.35	96.78	96.97	96.12	96.61	96.69	94.44	97.2	97.9
libras	78.89	82.78	86.94	81.39	82.78	83.89	83.89	84.17	87.22
low-res-spect	91.54	91.92	91.17	90.98	91.12	91.73	91.92	92.11	92.11
lung cancer	56.25	53.13	50	50	50	46.88	50	56.25	50
lymphography	85.81	84.46	87.16	86.49	87.84	86.49	87.84	86.49	87.84
magic	87.22	87.27	87.59	87.09	87.35	87.21	87.71	87.27	88.1
mammographic	82.71	81.04	81.04	81.15	80.73	82.19	80.42	81.04	83.13
miniboone	93.2	93.38	93.4	95.5	93.7	92.8	91.11	93.38	94.2
molec-biol-promoter	92.31	84.62	89.42	84.62	87.5	91.35	88.46	84.62	89.42

(continued on next page)

Datasets	RaF	obRaF ^[17]	obRaF(L)	obRaF(RR)	obRaF(S)	obRaF(LS)	obRaF(LR)	obRaF(M)	obRaF(H)
molec-biol-splice	95.29	94.84	93.1	92.97	87.58	93.29	92.13	94.29	86.9
monks-1	61.11	61	60.94	60.19	67.36	60.19	60.53	61	57.7
monks-2	66.2	66.44	66.67	66.78	66.2	65.97	67.19	66.44	66.84
monks-3	52.78	52.78	52.78	53.01	53.36	52.78	54.86	52.78	55.84
mushroom	100	100	100	100	100	100	100	100	100
musk-1	86.76	88.45	86.97	84.66	88.66	87.61	88.45	88.45	88.45
musk-2	97.67	96.92	97.41	97.91	96.97	97.74	97.88	96.92	97.84
nursery	99.64	99.59	99.71	99.37	99.39	99.67	99.65	99.66	99.61
oocMer12F	92.45	92.55	92.35	92.35	92.35	92.65	91.86	92.35	92.75
oocMer14D	78.82	82.25	85.78	84.8	81.57	79.71	82.06	82.25	84.02
oocTris2F	81.03	83.77	84.98	80.15	82.79	81.14	83.77	83.77	84.76
oocTris5B	91.78	92.54	95.18	93.86	93.64	94.63	93.42	93.97	94.96
optical	97.01	96.93	97.2	97.02	97.15	96.88	96.99	97.11	97.19
ozone	97.16	97.16	97.04	97.16	97.16	97.16	97.12	97.16	96.88
page-blocks	97.33	97.37	97.24	97.06	97.3	97.39	97.19	97.24	97.24
parkinsons	89.8	89.29	92.86	91.33	89.9	90.31	89.8	89.29	93.88
pendigits	95.28	96.12	96.95	97.01	97.16	97.01	96.34	96.93	97.14
pima	75.65	75.52	75.13	74.87	74.61	74.61	75.65	75.52	76.36
pb-MATERIAL	90.38	92.31	93.27	93.27	92.31	92.31	93.27	92.31	93.27
pb-REL-L	71.15	77.88	72.12	78.85	76.92	76.92	74.04	75.96	76.92
pb-SPAN	65.7	63.04	66.3	69.57	71.74	67.39	66.3	66.3	68.48
pb-T-OR-D	89	89	88	88	89	89	88	88	88
pb-TYPE	63.46	68.27	68.27	72.12	68.27	70.19	72.12	70.19	69.23
planning	72.22	71.11	71.67	71.67	72.22	72.22	72.22	71.11	72.67
plant-margin	86.19	85.38	86.94	85.13	84.56	83.94	83.31	85.19	87.44
plant-shape	64	66.38	74	71.94	68.31	69.69	62.3	69.06	75.12
plant-texture	82.69	82.5	83.56	81.79	82.44	83.37	83.1	82.44	84.81
post-operative	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73	72.73
primary-tumor	56.1	55.18	56.1	55.79	54.27	54.57	55.49	55.79	57.18
ringnorm	96.95	92.43	97.78	97.42	96.55	96.93	97.74	92.43	98.92
seeds	93.75	94.23	95.67	92.31	92.31	92.31	92.79	94.23	96.19
semeion	93.91	93.59	93.91	93.34	93.84	91.77	93.66	94.22	96.3
soybean	39.1	36.7	45.15	14.1	79.12	38.43	67.62	38.56	72.34
spambase	94.52	94.87	94.76	94.61	94.96	94.74	95.04	94.87	95.5
spect	68.95	66.26	62.63	54.84	61.83	65.59	54.84	66.26	65.05
spectf	91.98	91.98	91.98	91.98	91.98	91.98	91.98	91.98	91.98
st-aus-credit	68.02	68.02	67.88	68.17	68.02	68.02	67.88	68.02	68.22
st-german-credit	78.6	78.9	78.9	75.8	78.2	79	78.3	78.9	77.8
st-heart	87.31	86.57	86.19	83.58	88.06	86.94	85.07	86.57	86.94
st-image	97.79	97.8	98.18	97.79	97.66	98.35	98.09	98.18	98.4
st-landstat	90.89	95.09	90.07	89.54	91.06	90.66	89.62	90.63	91.96
st-shuttle	99.99	99.97	99.95	99.95	99.97	99.99	99.98	99.99	100
st-vehicle	75.59	78.32	83.89	82.46	76.66	82.94	79.15	82.82	84.9
steel plates	78.71	77.94	78.61	77.42	77.94	77.84	78.25	79.12	78.63
synthetic-control	98.67	99	99.5	98.67	99.5	99.5	99.17	99.5	99.17
teaching	62.1	62.5	63.16	62.5	62.5	63.16	62.5	62.5	63.82
thyroid	98.95	98.95	98.98	98.53	98.69	98.92	98.75	98.95	98.5
tic-tac-toe	98.95	97.8	98.54	97.59	98.01	98.64	97.7	97.8	97.91
titanic	78.64	78.68	78.95	78.36	78.45	78.32	78.95	78.68	78.95
trains	100	87.5	87.5	75	87.5	100	87.5	87.5	87.5
twonorm	97.58	97.85	97.88	97.51	97.88	97.45	97.85	97.85	97.73
vc-2classes	83.44	84.09	83.77	83.12	84.42	83.77	84.42	84.09	85.39
vc-3classes	84.42	85.06	85.39	81.82	84.42	84.74	84.74	86.04	85.71
wall-following	99.73	99.7	96.79	97.56	95.77	98.13	98.59	97.84	96.92
waveform	84.84	86.24	86.24	86.08	86.38	85.88	86.04	86.58	86.28
waveform-noise	86.02	86.48	87.22	85.66	86.46	84.1	87.12	87.12	87.06
wine	98.86	98.96	100	100	99.43	99.43	98.86	99.43	98.3
w-qu-a-red	68.56	68.19	70.31	69.75	68.87	69.75	69.44	69.94	70.38
w-qu-a-white	68.77	68.97	69.28	68.83	69.46	69.14	69.3	69.83	69.71
yeast	62.33	62.26	63.34	62.8	62.87	62.47	61.73	63.27	63.81
zoo	99	99	99	99	99	99	99	99	99
Mean Acc.	81.99	82.04	82.64	81.51	82.18	82.19	81.95	82.3	83.36

L, RR, S, LS, LR, M and H in the brackets along with obRaF refer to our proposed LDA, Ridge Regression, SVM, LSSVM, LR, MPSVM and heterogeneous oblique random forests respectively. Bold values represent the highest accuracy.