A Robust Classifier for Imbalanced Datasets

Sori Kang and Kotagiri Ramamohanarao

Department of Computing and Information Systems, The University of Melbourne, Parkville Victoria 3052 Australia

Abstract. Imbalanced dataset classification is a challenging problem, since many classifiers are sensitive to class distribution so that the classifiers' prediction has bias towards majority class. Hellinger Distance has been proven that it is skew-insensitive and the decision trees that employ Hellinger Distance as a splitting criterion have shown better performance than other decision trees based on Information Gain. We propose a new decision tree induction classifier (HeDEx) based on Hellinger Distance that is randomized ensemble trees selecting both attribute and split-point at random. We also propose hyperplane as a decision surface for HeDEx to improve the performance. A new pattern-based oversampling method is also proposed in this paper to reduce the bias towards majority class. The patterns are detected from HeDEx and the new instances generated are applied after verification process using Hellinger Distance Decision Trees. Our experiments show that the proposed methods show performance improvements on imbalanced datasets over the state-of-the-art Hellinger Distance Decision Trees.

1 Introduction

In machine learning, the classification problem aims to predict class labels of new unseen examples on the basis of previously observed training datasets. Many methods have been proposed to generate high accuracy classifiers, but most classification methods show good performance on balanced class problems - the number of instances of classes is balanced - while yielding relatively poor performance on imbalanced class problems.

Imbalanced class distribution - one class (majority class, denoted as '-') vastly outnumbers the other class (minority class, denoted as '+') in training datasets - hinders the accuracy of classification of minority class, since typical classification algorithms, such as decision trees, intend to maximize the overall prediction accuracy and tend to have bias toward the majority class [1]. The class imbalance problem, however, is important, since imbalanced datasets are prevalent in real world (e.g. fraud/intrusion detection, medical diagnosis/monitoring) and the cost of misclassification for a minority class is usually much higher in many cases. For instance, since the examples of patient with a rare cancer are relatively very small, the classifier usually has a poor ability to predict the rare cancer. Therefore, the classifier can simply classify the patient with a rare cancer into the patient with a common cancer while the classifier keeps high accuracy. When

V.S. Tseng et al. (Eds.): PAKDD 2014, Part I, LNAI 8443, pp. 212–223, 2014.

such a misclassification happens, however, the misclassified patient may suffer from misdiagnosis.

There have been various approaches to tackle the class imbalance problems; kernel modification methods, sampling methods, and cost-sensitive methods. In this paper, we focus on methods that can apply to Decision Tree Induction Classification, as decision tree is one of the most effective methods for classification [2]. Based on the decision tree, there have been many studies that showed performance improvement on imbalanced datasets. However, there is still a room for improvement.

In this paper, we propose Hellinger Distance Extra Decision Tree (HeDEx) that employs Hellinger Distance as a split criterion and builds extremely randomized ensemble trees. Hellinger Distance Extra Decision Tree is named after Hellinger Distance Decision Tree [3] and Extra-Trees [4]. We also propose a novel oversampling method that helps not only our proposed decision tree but also other existing classifiers to get better performance for minority class. Our experiments show that HeDEx has generally better performance than other existing decision tree methods in terms of AUC and F-Measure for minority class. The proposed oversampling method improves the performance of F-Measure for minority class.

2 Related Work

Information gain and Gini index are used as the splitting criteria for the popular decision trees such as C4.5 and CART, respectively. However, several studies [5][6][7] have shown that these measures are skew-sensitive so that they have bias toward majority class. Equation 1 and 2 denote Entropy and Information Gain for binary class and binary split, respectively. In order to maximize the information we need to minimize the second term of Equation 2, since the first term $Entropy(S)$ is fixed for the dataset S. The second term of Equation 2 is denoted as $\frac{1}{|S|}\big(-|S_{1+}|log_2\frac{|S_{1+}|}{|S1|}-|S_{1-}|log_2\frac{|S_{1-}|}{|S1|}\big)+\frac{1}{|S|}\big(-|S_{2+}|log_2\frac{|S_{2+}|}{|S2|}-|S_{2-}|log_2\frac{|S_{2-}|}{|S2|}\big)$. Therefore, the minority class could not influence equally on Entropy like majority class, since $|S_{1+}|$ and $|S_{2+}|$ is relatively small in class imbalanced datasets. The classifier that uses Gini Index (Equation 4) also has the same problem. The studies also shows the skew-sensitivity of Information Gain using isometric form [8][3].

Cieslak et al. [3] proposed Hellinger Distance as a splitting criterion that is skew-insenstive. Equation 5 shows Hellinger Distance, and it shows that the class priors do not influence the distance calculation. Thus the minority class would not be ignored on distance calcuation. The experiments of Cieslak et al. showed that Hellinger Distance Decision Tree (HDDT) outperforms C4.4 - unpruned, uncollapsed C4.5 with Laplace smoothing - [3] in imbalanced class problems.

$$Entropy(S) = -\sum_{c=+,-}\frac{|S_c|}{|S|}log_2\frac{|S_c|}{|S|} \tag{1}$$

$$InfoGain(S) = Entropy(S) - \sum_{j=1,2}\frac{|S_j|}{|S|}Entropy(S_j) \tag{2}$$

$$InfoGainRatio(S) = \frac{InfoGain(S)}{\sum_{j=1,2} \frac{|S_j|}{|S|} log_2 \frac{|S_j|}{|S|}} \tag{3}$$

$$Gini(S) = 1 - \sum_{c=+,-} (\frac{|S_c|}{|S|})^2 \tag{4}$$

$$HellingerDisttance(S) = \sqrt{\sum_{j=1,2} (\sqrt{\frac{|S_{+j}|}{|S_+|}} - \sqrt{\frac{|S_{-j}|}{|S_-|}})^2} \tag{5}$$

Sampling methods are typically used to tackle imbalanced class problems. The experimental studies[9] have shown that using sampling methods generally improve the classifier performance. The sampling methods alter the class distribution to make the class distribution balanced.

Synthetic Minority Oversampling TEchnique (SMOTE) [10] takes the same approach with One-Sided Selection; mitigating bias toward majority class. The difference between two methods is that while One-Sided Selection removes the majority class data, SMOTE proposes creating synthesized minority class examples. Synthetic examples are generated by selecting a random point along the line between two minority class examples. Chawla et al. shows that this synthesized examples facilitate larger decision regions in feature space for minority class avoiding overfitting. Despite improved performance of SMOTE, the problem of SMOTE is overgeneralization, which means the region enlarged for minority class could be blindly generalized so that synthetic instances can lead to overlapping between classes. Many other synthetic sampling methods based on SMOTE have been proposed to address the overgeneralization problem of SMOTE; Border-line SMOTE [11], Safe-Level SMOTE [12], and LN-SMOTE [13].

Cluster-Based Oversampling [14] is the other method to solve the small disjunction problem by using clustering approach. It clusters the training data of each class separately and oversamples the minority instances per each cluster. In this idea, the new generated minority instances cannot be located in some majority class cluster. Thus, by clustering approach, it can handle both inter-class imbalance and between-class imbalance simultaneously.

The other approach for oversampling is pattern-based synthetic method. Alhammady et al. [15] proposed Emerging Patterns Decision Tree (EPDT) that is decision tree induction classifier using Emerging Patterns of minority class for generating new minority class instances. Emerging Pattern of minority class is a pattern whose support changes significantly from majority class to minority class in the training dataset.

Ensemble method is known to be very efficient to reduce variance by building multiple classifiers and averaging the classifiers output so that it allows us not to choose the classifier with a poor performance. There have been many ensemble methods based on Bagging [16] and Boosting [17]. In this review, however, we focus on Randomized ensemble trees that use a subset of attributes, since our method is based on Random Subspace [18].

Table 1. Comparison of Randomized Ensemble Methods

Ensemble method	Training samples for tree	Candidate split-points at node
Random Subspace	all training samples	all values of selected attributes
Random Forest	bootstrap replicas	all values of selected attributes
Extra-Trees	all training samples	$c = 1$, a random value of each selected attribute
HeDEx	all training samples	$c \geq 1$, random values of each selected attribute

Instead of using all attributes to find the optimal split-point at a tree node, Random Subspace and Random Forests randomly select a subset of attributes at a tree node and find the optimal split-point among only these selected attributes. The level of randomization is determined by the number K of attributes to be chosen at each node. The difference between two methods is the formation of the training examples of each tree. Random Subspace uses the entire training dataset for each tree so that only the randomly chosen attributes impact the variability of base classifiers. On the other hand, Random Forests is based on Bagging [16] - random sampling of training dataset with replacement - so that both attribute selection and training data impact the variability.

Extra-Trees [4] was proposed as extremely randomized trees. It randomly selects not only the number K of attributes but also a candidate split-point for each chosen attribute. The split-point at a tree node is determined among the number K of candidate split-points so that the split-point is sub-optimal for a corresponding node. Therefore, among three methods (Random Subspace, Random Forests, and Extra-Trees) Extra-Trees has the highest randomness. The advantage of Extra-Trees is computational efficiency while keeping comparative accuracy, reducing variance and slightly increasing bias.

3 Hellinger Distance Extra Decision Tree

In this paper, we propose Hellinger Distance Extra Decision Tree (HeDEx) that employs Hellinger Distance as a splitting criterion and build extremely randomized ensemble trees using entire training dataset for imbalanced dataset problems. HeDEx is basically based on Extra-Trees [4]. The main differences with other ensemble methods are that Extra-Trees and HeDEx randomly choose not only the attributes but also split-points for splitting the tree nodes and use the entire training examples (rather than a bootstrap replica) to build each base classifier. Due to the randomization on both attribute selection and split-point selection, Extra-Trees and HeDEx can achieve high level of varierity of trees even without sampling of training examples (bootstrap replicas).

The two main differences with Extra-Trees and HeDEx are splitting-criterion and the number of candidate split-points. Extra-Trees selects one candidate split-point for each attribute, while HeDEx selects more than one split-points, since we found from experiments that considering multiple candidate split-points to find a sub-optimal split-point showed better accuracy. The other difference is splitting

Table 2. HeDEx Parameters

Parameter	Description
M	the number of trees to be built
K	the number of attributes randomly selected at each node
C	the number of split-points randomly selected for each attribute
n_{min}	the minimum sample size at a leaf node

criterion. HeDEx uses Hellinger Distance for skew-insensitiveness, while Extra-Trees uses a score measure based on information gain. We provide the comparison table of existing random ensemble methods in Table 1. Notice that although the number of split-points c of HeDEx becomes equal to the number of values of the corresponding attribute in learning samples, HeDEx is different from Random Subspace. This is because HeDEx draws the split-points independently from the values in the learning samples.

We use weighted method as the aggregation rule for our HeDEx. Equation 6 shows how to predict the class label of a test example. C in Equation 6 is a set of class labels and $p_{t_i,c}$ is the probability of classifying as class label c in tree t_i such that $\sum_{c \in C} p_{t_i,c} = 1$ for each tree.

$$class = \arg\max_{c \in C} \sum_{i=1}^{M} p_{t_i,c} \tag{6}$$

3.1 Variant of Hellinger Distance Extra Decision Tree

We also propose a variant of HeDEx that employs different decision boundary from the original HeDEx. Hellinger Distance Extra Hyperplane Decision Tree (HeDExh) uses an arbitrary hyperplane as a decision surface, thus selects the optimal hyperplane at a node among K candidate hyperplane decision boundaries. To be brief, we could say that HeDEx uses single variable inequations when it tries to split the node according to the split-point, while HeDExh uses two variable inequations when it tries to split the node. The arbitrary hyperlane is determined by choosing two different points from feature space of the dataset; e.g. (v_{11}, v_{21}) and (v_{12}, v_{22}) where v_{1x} for attribute 1 and v_{2x} for attribute 2. The number of candidate hyperplane for a pair of chosen attributes is also an option parameter, C. The number of attributes to be used for hyperplane could be more than two. In this paper, we present 2-dimensional hyperplane only but the dimensionality can be extended.

4 Pattern-Based New Instance Creation

In this section, we propose the new method for generating minority instances based on patterns that are detected from HeDEx. This process consists of three parts; pattern detection, instance generation, and instance validation.

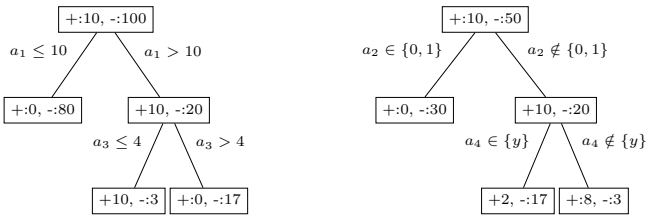

Fig. 1. An example of HeDEx for pattern detection

4.1 Pattern Detection

The main idea of pattern detection is that in decision tree the splitting criteria at each node of tree represent the patterns of the examples at the node. The patterns are detected from HeDEx that are built using the training samples. As HeDEx builds sub-optimal trees, it has a variety of patterns that are not highly coupled with training examples. After building HeDEx, we look at the leaf node of each tree, and if the number of minority instances is greater than that of majority, we build a pattern according to the split rules from root to the leaf node. For example, from the trees in Figure 1, we can build patterns; $\{a_1 > 10$ and $a_3 \leq 4\}$ and $\{a_2 \notin \{0, 1\}$ and $a_4 \notin \{y\}\}$, respectively.

The patterns detected from multiple HeDEx trees get scores with respect to the strength of the pattern, and a pattern with higher score has higher probability of being selected for generating new instances. The equations 7 and 8 show how to get strength score of a pattern. For example, for the pattern detected from Figure 1 has $GrowthRate_{c_+} = {}^{10}/_{3}$ and $Strength_{c_+} = 10 * {}^{10}/_{(10/3+1)}$ for minority class.

$$Strength_{c_+} = Support_{c_+} * \frac{GrowthRate_{c_+}}{GrowthRate_{c_+} + 1} \tag{7}$$

$$GrowthRate_{c_+} = \frac{Support_{c_+}}{Support_{c_-}} \tag{8}$$

4.2 Instance Generation

New instances are generated based on the patterns. If the selected pattern does not cover all attributes, the values of missing attributes are generated at random but based on the distribution of the values of the training examples. Figure 2 shows an example of generating a new instance. As the pattern 1 has the rule $\{a_1 > 10$ and $a_3 \leq 4\}$, we generate random values of a_1 and a_3 according to the rule; $10 < v_1 \leq max(a_1)$ and $min(a_3) \leq v_3 \leq 4$. For the remaining attributes, we randomly draw the values such that $v_i \sim histogram(a_i)$ for ith attribute.

For the nominal attributes, we select the attribute values at random according to the rule. For example, the second pattern of Figure 1 is $\{a_2 \notin \{0, 1\}$ and $a_4 \notin \{y\}\}$. The value of a_2 is drawn uniformly from $Set_{a_2} \backslash \{0, 1\}$ and the value

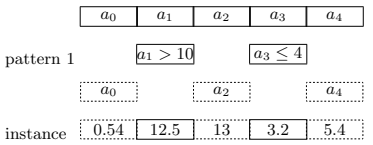

Fig. 2. Instance creation based on patterns and attribute value histogram

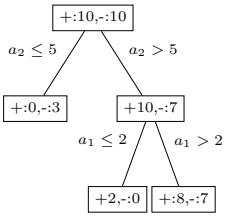

Fig. 3. An example of HDDT for instance validation

of a_4, is drawn uniformly from $Set_{a_4} \backslash \{y\}$ where Set_{a_i} is the set of attribute a_i's distinct values. For remaining attributes, we randomly draw each value such that $v_i \sim histogram(a_i)$ for ith attribute.

4.3 Instance Validation

We propose the instance validation phase to ensure that the newly generated instances are not noisy for minority class. The main idea of this validation phase is that if a classifier cannot distinguish instances of two different classes well, which can mean that the instances of two different classes are similar.

After generating new instances, we make a synthetic training dataset using the original minority instances and newly generated instances to validate new instances. We set the original minority instances as positive class and the new instances as negative class. We build a Hellinger Distance Decision Tree and visit each leaf node to check the instance distribution. Figure 3 shows the example. The right-most leaf node illustrates that the negative instances are not distinguishable from the positive instances, so we can add these negative instances (newly generated instances) as the instances for minority class to the training examples for building a classifier. On the other hand, the left most node contains only the negative instances, which implies that the three negative instances are well distinguishable from the positive class instances so that the three instances could become noisy if we add these instances to training examples. In that case, we do not include such instances for our training examples. Notice that we employ Hellinger Distance Decision Tree for validating instances, since HDDT builds a tree with optimal, discriminative splitting criteria.

Table 3. Datasets for imbalanced classes

Dataset	Classes	Type	Features	Instances	CV
compustat	2	numeric	20	10358	0.93
covtype	2	numeric	10	38500	0.86
estate	2	numeric	12	5322	0.76
german.numer	2	numeric	24	1000	0.4
ism	2	numeric	6	11180	0.95
letter	2	numeric	16	20000	0.92
oil	2	numeric	49	937	0.91
page	2	numeric	10	5473	0.8
pendigits	2	numeric	16	10992	0.79
phoneme	2	numeric	5	5400	0.41
PhosS	2	numeric	480	11411	0.89
satimage	2	numeric	36	6430	0.81
segment	2	numeric	19	2310	0.71
credit-g	2	both	20	1000	0.4
boundary	2	nominal	175	3661	0.93
breast-y	2	nominal	9	286	0.41
cam	2	nominal	132	18916	0.9
car	4	nominal	6	1728	1.25
dna	3	nominal	180	3186	0.48
glass	6	numeric	9	214	0.83
nursery	4	nominal	8	12961	0.60
page-5	5	numeric	10	5473	1.95
sat	6	numeric	36	6435	0.41

5 Experiements

The 23 datasets were chosen from the datasets that were used in HDDT experiments[1]. These datasets originate from UCI[2], LibSVM[3] and two other studies [8][10]. The 23 datasets have a variety of characteristics in terms of the level of imbalance and the number of attributes including both binary class and multi-class. Dataset 'page' and 'satimage' are originally the same dataset with 'page-5' and 'sat', respectively, but have different number of classes. In order to measure the level of imbalance in class distribution, we employ the coefficient of variance ($CV = \sigma/\mu$) of class distribution like the study of HDDT [3]. Higher CV means higher skewness in class distribution. Table 3 provides details of the dataset used in this experiment.

We chose Bagging HDDT and Extra-Trees as the methods to be compared with our proposed methods. As Cieslak et al. showed in the study of HDDT [3] that Bagging HDDT outperforms other methods including C4.4, C4.4 combined with Bagging and Sampling methods, and HDDT combined with Sampling, we excluded other ensemble and sampling methods in this paper. We also use Bagging HDDT for the performance comparison of pattern-based oversampling while excluding other oversampling methods, since Bagging HDDT is proved that it has better performance than other oversampling methods[3]. We included Bagging C4.5 as a baseline and Logistic Regression for comparison purpose as it is insensitive to class distribution. Table 4 shows terminologies for algorithms that we use in this experiment. 5x2 cross-validation is used to evaluate each algorithm.

We used the default parameters that are mentioned on the corresponding papers for both Bagging HDDT and Extra-Trees; 100 unpruned trees (M = 100) with laplace smoothing and $n_{min} = 2$. For Extra-Trees and our methods the K number of candidate attributes is $K = \sqrt{n}$ as Extra-Trees recommended. HeDEx and HeDExh have another parameter for the number of candidate split-points, C. We set $C = 10$ as default, as we found that for most datasets above 10 candidate

[1] `http://www3.nd.edu/~dial/hddt/`
[2] `http://archive.ics.uci.edu/ml/`
[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Table 4. Algorithms

Abbr.	Algorithm
LR	Bagging Logistic Regression
C4.5	Bagging C4.5, pruned
HDDT	Bagging Hellinger Distance Decision Tree
ET	Extra-Trees
HeDEx	Hellinger Distance Extra Decision Trees
HeDExh	Hellinger Distance Extra Hyper Decision Trees
$\mathbf{X^s}$	Algorithm X with pattern-based oversampling e.g. LR^s

Table 5. (Win/Draw/Loss) table of statistical significance test (corrected t-test) result at 95% for F-Measure$^+$

	LR	C4.5	ET	HDDT	HeDEx	HeDExh
LR		8/10/5	8/11/4	9/10/4	10/10/3	11/9/3
C4.5	5/10/8		3/18/2	3/19/1	7/15/1	7/15/1
ET	4/11/8	2/18/3		3/18/2	5/18/0	7/16/0
HDDT	4/10/9	1/19/3	2/18/3		3/19/1	5/18/0
HeDEx	3/10/10	1/15/7	0/18/5	1/19/3		1/22/0
HeDExh	3/9/11	1/15/7	0/16/7	0/18/5	0/22/1	
Total	19/50/46	13/77/25	13/81/21	16/84/15	25/84/6	**31/80/4**

	LR^s	$C4.5^s$	ET^s	$HDDT^s$	$HeDEx^s$	$HeDExh^s$
$\mathbf{LR^s}$		9/12/2	11/12/0	9/12/2	12/10/1	11/12/0
$\mathbf{C4.5^s}$	2/12/9		6/17/0	0/22/1	7/16/0	8/15/0
$\mathbf{ET^s}$	0/12/11	0/17/6		0/19/4	2/21/0	3/20/0
$\mathbf{HDDT^s}$	2/12/9	1/22/0	4/19/0		6/17/0	6/17/0
$\mathbf{HeDEx^s}$	1/10/12	0/16/7	0/21/2	0/17/6		1/21/1
$\mathbf{HeDExh^s}$	0/12/11	0/15/8	0/20/3	0/17/6	1/21/1	
Total	5/58/52	10/82/23	21/89/5	9/87/19	28/85/2	**29/85/1**

Table 6. (Win/Draw/Loss) table of statistical significance test (corrected t-test) result at 95% for AUC

	LR	C4.5	ET	HDDT	HeDEx	HeDExh
LR		9/12/2	13/9/1	9/12/2	14/8/1	12/10/1
C4.5	2/12/9		8/15/0	4/17/2	9/14/0	9/14/0
ET	1/9/13	0/15/8		0/18/5	2/21/0	4/19/0
HDDT	2/12/9	2/17/4	5/18/0		7/16/0	8/15/0
HeDEx	1/8/14	0/14/9	0/21/2	0/16/7		2/21/0
HeDExh	1/10/12	0/14/9	0/17/6	0/15/8	0/21/2	
Total	7/51/57	11/72/32	26/80/9	13/78/24	32/80/3	**35/79/1**

	LR^s	$C4.5^s$	ET^s	$HDDT^s$	$HeDEx^s$	$HeDExh^s$
$\mathbf{LR^s}$		9/11/3	15/7/1	11/9/3	15/7/1	14/8/1
$\mathbf{C4.5^s}$	3/11/9		7/16/0	2/19/2	7/16/0	8/15/0
$\mathbf{ET^s}$	1/7/15	0/16/7		0/15/8	7/16/0	9/12/2
$\mathbf{HDDT^s}$	3/9/11	2/19/2	8/15/0		8/15/0	9/14/0
$\mathbf{HeDEx^s}$	1/7/15	0/16/7	0/16/7	0/15/8		3/20/0
$\mathbf{HeDExh^s}$	1/8/14	0/15/8	2/12/9	0/14/9	0/20/3	
Total	9/42/64	11/77/27	32/66/17	13/72/30	37/74/4	**43/69/1**

split-points has no significant effects on the performance improvement. For the pattern-based oversampling method, the amount of newly generated minority instances makes the class distribution of final training dataset balanced, since we noticed that balanced distribution brought the best performance on HeDEx, although we cannot present the experiments results due to the space limitation.

The popular evaluation measure for imbalanced dataset problems is Area Under the Receiver Operating Characteristic curve (AUC), thus we employ AUC in our paper to compare different classifiers' performance. In addition, we present the comparison of F_1-Measure for the minority class to show the classifiers' performance toward minority class. In this paper, F-Measure$^+$ denotes F_1-Measure for the minority. We used corrected paired t-test to determine the statistical significance of performances of the compared algorithms.

Table 7. F-Measure$^+$ performance results for dataset in Table 3

	LR	LR^s	C4.5	C4.5^s	ET	ET^s	HDDT	HDDT^s	HeDEx	HeDEx^s	HeDExh	HeDExh^s
compustat	0.026	0.222 v	0.101	0.292 v	0.064	0.322 v	0.136	0.298 v	0.153	0.340 v	0.222	0.355 v
covtype	0.369	0.476 v	0.866	0.851	0.869	0.867	0.880	0.859	0.887	0.878	0.904	0.896
estate	0.047	0.216 v	0.007	0.212 v	0.036	0.218 v	0.054	0.208 v	0.067	0.210 v	0.087	0.220 v
german.numer	0.542	0.593 v	0.528	0.571	0.465	0.576 v	0.546	0.580	0.501	0.570	0.539	0.578
ism	0.543	0.507	0.634	0.603	0.612	0.628	0.647	0.613	0.645	0.631	0.647	0.629
letter	0.878	0.801 *	0.948	0.925 *	0.943	0.945	0.957	0.940	0.965	0.959	0.965	0.961
oil	0.476	0.463	0.433	0.473	0.250	0.583	0.392	0.464	0.345	0.570	0.434	0.550
page	0.726	0.705	0.875	0.868	0.865	0.872	0.877	0.864	0.881	0.880	0.869	0.871
pendigits	0.910	0.860 *	0.960	0.969	0.988	0.988	0.977	0.970	0.986	0.985	0.989	0.990
phoneme	0.522	0.626 v	0.791	0.799	0.814	0.816	0.798	0.804	0.823	0.825	0.822	0.824
PhosS	0.178	0.239 v	0.102	0.228 v	0.001	0.251 v	0.083	0.234 v	0.005	0.249 v	0.041	0.248 v
satimage	0.044	0.274 v	0.634	0.626	0.614	0.640	0.647	0.641	0.654	0.655	0.647	0.655
segment	0.990	0.962	0.977	0.973	0.989	0.987	0.980	0.976	0.992	0.984	0.992	0.988
credit-g	0.528	0.571	0.491	0.553 v	0.393	0.557 v	0.534	0.558	0.502	0.567	0.476	0.563 v
boundary	0.162	0.174	0.009	0.146 v	0.000	0.147 v	0.000	0.142 v	0.000	0.154 v	0.006	0.158 v
breast-y	0.381	0.433	0.349	0.449 v	0.392	0.471	0.420	0.452	0.419	0.465	0.439	0.485
cam	0.185	0.317 v	0.015	0.250 v	0.005	0.276 v	0.022	0.241 v	0.014	0.281 v	0.023	0.294 v
car	0.855	0.842	0.603	0.739	0.859	0.806	0.865	0.791	0.913	0.837	0.926	0.842
dna	0.894	0.912	0.913	0.910	0.930	0.929	0.895	0.895	0.930	0.923	0.924	0.914
glass	0.633	0.684	0.687	0.723	0.634	0.628	0.420	0.554	0.635	0.666	0.489	0.607
nursery	0.770	0.730	0.672	0.793 v	0.713	0.891 v	0.004	0.280 v	0.984	0.956	0.984	0.963
page-blocks-5	0.579	0.576	0.805	0.743	0.791	0.660	0.782	0.730	0.809	0.752	0.784	0.641
sat	0.456	0.549 v	0.656	0.649	0.657	0.649	0.650	0.646	0.669	0.659	0.670	0.658
(win/draw/loss)[1]		(9/12/2)		(8/14/1)		(8/15/0)		(6/17/0)		(5/18/0)		(6/17/0)

[1] The result of statistical significance test (corrected paired t-test) at 95% against algorithm X^s. v means win and * means loss.

Table 8. AUC performance results for dataset in Table 3

	LR	LR^s	C4.5	C4.5^s	ET	ET^s	HDDT	HDDT^s	HeDEx	HeDEx^s	HeDExh	HeDExh^s
compustat	0.7843	0.7899	0.8811	0.8595	0.9211	0.8924 *	0.8988	0.8719 *	0.9212	0.8986 *	0.9281	0.9039 *
covtype	0.9026	0.9035	0.9918	0.9894	0.9945	0.9929 *	0.9940	0.9912 *	0.9952	0.9938 *	0.9966	0.9955 *
estate	0.6245	0.6183	0.6400	0.6301	0.6439	0.6433	0.6281	0.6292	0.6328	0.6363	0.6336	0.6381
german.numer	0.7792	0.7802	0.7750	0.7770	0.7786	0.7798	0.7794	0.7781	0.7787	0.7795	0.7819	0.7806
ism	0.9181	0.9264	0.9278	0.9342	0.9494	0.9481	0.9391	0.9435	0.9493	0.9472	0.9481	0.9464
letter	0.9861	0.9825*	0.9973	0.9987	0.9998	0.9994	0.9988	0.9981	0.9999	0.9997	0.9999	0.9997
oil	0.8870	0.9072	0.8964	0.8905	0.9206	0.9164	0.9058	0.8922	0.9251	0.9192	0.9268	0.9256
page	0.9474	0.9203*	0.9903	0.9898	0.9915	0.9908	0.9916	0.9903	0.9922	0.9914	0.9919	0.9914
pendigits	0.9907	0.9864*	0.9992	0.9982	1.0000	0.9999	0.9988	0.9972	0.9999	0.9999	1.0000	1.0000
phoneme	0.8125	0.8129	0.9398	0.9401	0.9531	0.9513	0.9451	0.9437	0.9554	0.9534	0.9567	0.9553
PhosS	0.7269	0.7420 v	0.7431	0.7285	0.7715	0.7594	0.7459	0.7313	0.7697	0.7525	0.7673	0.7510
satimage	0.7657	0.7565	0.9480	0.9416	0.9563	0.9483 *	0.9537	0.9479 *	0.9591	0.9523 *	0.9607	0.9532 *
segment	0.9999	0.9988*	0.9977	0.9977	0.9999	0.9999	0.9971	0.9965	0.9999	0.9999	1.0000	0.9999
credit-g	0.7618	0.7670	0.7568	0.7591	0.7794	0.7773	0.7755	0.7709	0.7775	0.7766	0.7817	0.7818
boundary	0.7357	0.7270	0.5548	0.6691	0.6802	0.6817	0.6732	0.6527	0.6888	0.6811	0.7092	0.6999
breast-y	0.6284	0.6389	0.6754	0.6639	0.6648	0.6718	0.6552	0.6651	0.6659	0.6729	0.6624	0.6679
cam	0.8312	0.8289	0.7045	0.7315	0.7775	0.7805	0.7806	0.7512 *	0.7892	0.7847	0.8051	0.7987
car	0.9893	0.9892	0.9831	0.9821	0.9954	0.9940	0.9941	0.9940	0.9971	0.9967	0.9980	0.9974
dna	0.9855	0.9873	0.9888	0.9880	0.9925	0.9923	0.9797	0.9782	0.9925	0.9915	0.9917	0.9908
glass	0.8410	0.8402	0.8763	0.8790	0.9184	0.9176	0.8772	0.8699	0.9090	0.9095	0.8992	0.8983
nursery	0.9882	0.9871*	0.9958	0.9955	0.9984	0.9989 v	0.9479	0.9367	0.9998	0.9998	0.9999	0.9999
page-blocks-5	0.9889	0.9862*	0.9895	0.9885	0.9916	0.9913	0.9907	0.9895	0.9924	0.9920	0.9917	0.9916
sat	0.9802	0.9811	0.9889	0.9887	0.9897	0.9891 *	0.9887	0.9885	0.9901	0.9897 *	0.9905	0.9900 *
(win/draw/loss)[1]		(1/16/6)		(0/23/0)		(1/18/4)		(0/19/4)		(0/19/4)		(0/19/4)

[1] The result of statistical significance test (corrected paired t-test) at 95% against algorithm X^s. v means win and * means loss.

6 Experiment Results

6.1 Comparison of Methods

Table 5 shows win/draw/loss counts for F-Measure$^+$ of 23 datasets comparing the algorithm in the column versus the algorithm in the row. As can be seen, HeDEx has almost the same or better performance in terms of F-Measure$^+$ than HDDT and has bigger improvements from C4.5 than HDDT does. The only loss of HeDEx against HDDT and C4.5 is the dataset PhoSs. This is due to that Extra-Trees' performance becomes poorer as the dimensionality of dataset becomes higher. However, we noticed that HeDExh overcomes this disadvantage

of extremely randomized trees. We also noticed that Logistic Regression has better performance on imbalanced dataset with higher dimensionality than other classifiers we compared. In terms of both AUC presented in Table 6 and F-Measure$^+$, HeDEx and HeDExh are the best option among others on average. Table 7 and 8 show the figures of F-Measure$^+$ and AUC for each method.

6.2 Effects of Pattern-Based Oversampling

Table 7 and Table 8 show the effects of pattern-based oversampling in terms of F-Measure for minority class and AUC. Pattern-based oversampling helps the performance improvement especially when the dataset has higher dimensionality. The F-Measure$^+$ is improved from 150% to more than 3000% compared to not using pattern-based oversampling. For F-Measure$^+$, we noticed that the degree of performance improvement due to oversampling varies among the classifiers. Decision Trees based on Hellinger Distance splitting criterion (HDDT, HeDEx, HeDExh) generally have lower improvements on performance than other classifiers such as LR, C4.5 and Extra-Trees.

In terms of AUC, however, pattern-based oversampling shows statistically significant losses on performance for 4 datasets among 23 datasets. That is why oversampling favors to minority class and AUC weights more on majority due to the class distribution. This result is similar to other studies [3][7] in which authors said that sampling is not helpful if the classifier employs a skew-insensitive split criterion.

7 Conclusion

In this paper, we have proposed a new decision tree induction classifier (HeDEx) that combines extremely randomized tree (Extra-Trees [4]) with multiple candidate split-points and Hellinger Distance as a splitting criterion for imbalanced dataset problems. We also have proposed a variant of HeDEx that employs a hyperplane decision surface (HeDExh). The main contribution of our proposed methods is that they build robust decision trees against imbalanced datasets with high computational efficiency. Moreover, because of choosing sub-optimal split-points at each node, the ensemble trees produced are all independent from each other. Due to the diversity of the shape of trees we can gather the variety of patterns from training examples, and the patterns are used to generate new minority class instances.

Overall, we ensure that Hellinger Distance is skew-insensitive as a splitting criterion, since applying Hellinger Distance to Extra-Trees shows improvement in the performance for imbalanced datasets. Moreover, we also verify that randomization at both attribute and split-point selection improves the performance of decision tree methods especially when combined with Hellinger Distance. Therefore, HeDEx and HeDExh for imbalanced dataset gives better prediction ability at lower or similar computational cost, respectively. In addition, as shown in study of HDDT[8], HeDEx can also be employed in the case of balanced datasets. Thus, we recommend HeDEx should be considered as one of classification methods to be chosen.

References

1. Hido, S., Kashima, H., Takahashi, Y.: Roughly balanced bagging for imbalanced data. Stat. Anal. Data Min. 2(5-6), 412–426 (2009)
2. Provost, F., Domingos, P.: Tree induction for probability-based ranking. Mach. Learn. 52(3), 199–215 (2003)
3. Cieslak, D.A., Hoens, T.R., Chawla, N.V., Kegelmeyer, W.P.: Hellinger distance decision trees are robust and skew-insensitive. Data Min. Knowl. Discov. 24(1), 136–158 (2012)
4. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning 63(1), 3–42 (2006)
5. Drummond, C., Holte, R.C.: Exploiting the cost (in)sensitivity of decision tree splitting criteria. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 239–246. Morgan Kaufmann (2000)
6. Flach, P.A.: The geometry of roc space: understanding machine learning metrics through roc isometrics. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 194–201. AAAI Press (2003)
7. Liu, W., Chawla, S., Cieslak, D.A., Chawla, N.V.: A Robust Decision Tree Algorithm for Imbalanced Data Sets. In: SDM, pp. 766–777. SIAM (2010)
8. Cieslak, D.A., Chawla, N.V.: Learning decision trees for unbalanced data. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 241–256. Springer, Heidelberg (2008)
9. Van Hulse, J., Khoshgoftaar, T.M., Napolitano, A.: Experimental perspectives on learning from imbalanced data. In: Proceedings of the 24th International Conference on Machine Learning, ICML 2007, pp. 935–942. ACM, New York (2007)
10. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. J. Artif. Int. Res. 16(1), 321–357 (2002)
11. Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In: Huang, D.-S., Zhang, X.-P., Huang, G.-B. (eds.) ICIC 2005, Part I. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005)
12. Bunkhumpornpat, C., Sinapiromsaran, K., Lursinsap, C.: Safe-level-SMOTE: Safe-level-synthetic minority over-sampling tEchnique for handling the class imbalanced problem. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 475–482. Springer, Heidelberg (2009)
13. Maciejewski, T., Stefanowski, J.: Local neighbourhood extension of smote for mining imbalanced data. In: 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 104–111 (2011)
14. Jo, T., Japkowicz, N.: Class imbalances versus small disjuncts. SIGKDD Explor. Newsl. 6(1), 40–49 (2004)
15. Alhammady, H., Ramamohanarao, K.: Using emerging patterns and decision trees in rare-class classification. In: Fourth IEEE International Conference on Data Mining, ICDM 2004, pp. 315–318 (2004)
16. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
17. Schapire, R.E.: The strength of weak learnability. Machine Learning 5(2), 197–227 (1990)
18. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)