Random Oracle Ensembles for Imbalanced Data

Juan J. Rodríguez, José-Francisco Díez-Pastor, and César García-Osorio

University of Burgos
{jjrodriguez,jfdpastor,cgosorio}@ubu.es

Abstract. In the Random Oracle ensemble method, each base classifier is a mini-ensemble of two classifiers and a randomly generated oracle that selects one of the two classifiers. The performance of this method have been previously studied, but not for imbalanced data sets. This work studies its performance for this kind of data. As the Random Oracle ensemble method can be combined with any other ensemble method, this work considers its combination with four ensemble methods: Bagging, SMOTEBoost, SMOTEBagging and RUSBoost. The last three methods combine classical, not specific for imbalance, ensemble methods (i.e., Bagging, Boosting), with pre-processing approaches designed for imbalance (i.e., random undersampling, SMOTE). The results show that Random Oracles improves all these methods.

1 Introduction

The classification of imbalanced data sets may require specific techniques. In addition, special performance measures are required, since with imbalanced data, conventional classification measures (for example, accuracy) are not useful. A measure used for the imbalance is the AUROC: area under the ROC curve (Receiver Operating Characteristics) [1]. Often when these curves are only considered, this area is called AUC. Another useful curve for imbalance is the Precision-Recall curve [2], the area under this curve is called AUPRC. These two curves (and their corresponding measures) are generated from the confidences given by the classifiers to their predictions. Another measure used for imbalance is the F-measure, the harmonic mean of precision and recall.

Various approaches for dealing with imbalanced data sets have been proposed [3]. In the data level approached the data is pre-processed, altering the classes distribution. For example, SMOTE [4] generates artificial instances of the minority class. Other approaches are based on ensemble methods. A common approach is to combine a method of ensembles that imbalance is not specific to a pre-processing technique.

A combination of Boosting and SMOTE is SMOTEBoost [5]. For each Boosting iteration, SMOTE is applied to the data, generating artificial instances of the minority class. SMOTEBagging [6] used Bagging and SMOTE. Each base classifiers is trained with a balanced data set: the classes have the same number of instances. These training data sets have instances that are obtained from resampling the original training data and artificial instances generated with SMOTE.

Z.-H. Zhou, F. Roli, and J. Kittler (Eds.): MCS 2013, LNCS 7872, pp. 247–258, 2013.

In order to increase the diversity, the percentage of instances from resampling and SMOTE is variable. RUSBoost [7] is a method based on SMOTEBoost, but it used random undersampling of the majority class instead of SMOTE of the minority class.

This paper studies the performance of the Random Oracle ensemble method [8,9,10] in imbalanced data. As this method can be used in conjunction with other ensemble methods, it will be combined with these ensemble methods for imbalance.

The rest of the paper is organized as follows: Sect. 2 describes the Random Oracles ensemble method. The experiments and results are presented in Sect. 3. Diversity and error of the base classifiers is considered in Sect. 4. Finally, Sect. 5 shows the conclusions.

2 Random Oracles

A Random Oracle classifier is formed by two classifiers and a randomly generated oracle. The oracle selects one of the two classifiers. It can be seen as a random discriminant function, it splits the data into two subsets without taking into consideration the class labels or cluster structure. Moreover, a Random Oracle classifier can be used as the base classifier of any other ensemble method.

Given a base method, the process for training a Random Oracle classifier is: 1) Select randomly the Random Oracle. 2) Split the training data in two subsets using the Random Oracle. 3) For each subset of the training data, build a classifier.

Given a test instance, the prediction is obtained in the following way: 1) Use the Random Oracle to select one of the two classifiers. 2) Return the prediction (and its confidence) given by the selected classifier.

For oracles with small computational complexity (both in training and prediction), the computational complexity of a Random Oracle classifier is very similar to the complexity of the base classifier. In the prediction phase, only one of the two classifiers is used. In the training phase, two classifiers are built. Nevertheless, they are trained with a disjoint partition of the training examples and the training time of any method depends, usually at least linearly, on the number of training examples.

Different types of Oracles can be considered. In this work two are considered: Linear and Spherical Random Oracles. The linear oracle divides the space into two subspaces using a hyperplane. To build the oracle, two different training objects are selected at random, the points that are at the same distance from the two training objects define the hyperplane. Each remaining training object is assigned to the subspace of the selected training object for which is closer. Algorithms 1 and 2 show the training and testing phases of this method.

The spherical oracle also divides the space into two subsets: inside and outside of a sphere. A training object is selected randomly as the center of the sphere. Another seven training examples are selected randomly, the distances from the center to these examples is calculated, the radius is the median of these seven

Input: Training dataset D; base learning method $\mathbb{L}$
Output: Random Oracle Model RO
RO.instance [1] $\leftarrow \{\mathbf{x} \mid (\mathbf{x}, y)$ is a random instance from $D\}$
RO.instance [2] $\leftarrow \{\mathbf{x} \mid (\mathbf{x}, y)$ is a random instance from $D\}$
$D_1 \leftarrow \emptyset$ // The training dataset for the 1^{st} sub-model
$D_2 \leftarrow \emptyset$ // The training dataset for the 2^{nd} sub-model
foreach *instance* $(\mathbf{x}, y) \in D$ **do**
  if distance $(RO.\text{instance}[1], \mathbf{x}) <$ distance$(RO.\text{instance}[2], \mathbf{x})$ **then**
    $D_1 \leftarrow D_1 \cup \{(\mathbf{x}, y)\}$ // Add the instance to the 1^{st} subset
  else
    $D_2 \leftarrow D_2 \cup \{(\mathbf{x}, y)\}$ // Add the instance to the 2^{nd} subset
RO.model [1] $\leftarrow \mathbb{L}(D_1)$ // Train the 1^{st} sub-model
RO.model [2] $\leftarrow \mathbb{L}(D_2)$ // Train the 2^{nd} sub-model

Algorithm 1. Random Linear Oracle method: training phase

Input: Trained Random Oracle RO; instance $\mathbf{x}$
Output: Predicted value
if distance $(RO.\text{instance}[1], \mathbf{x}) <$ distance$(RO.\text{instance}[2], \mathbf{x})$ **then**
  return RO.model [1].predict($\mathbf{x}$) // Predict with the 1^{st} sub-model
else
  return RO.model [2].predict($\mathbf{x}$) // Predict with the 2^{nd} sub-model

Algorithm 2. Random Linear Oracle method: prediction phase

distances. This is done with the purpose of having some guarantee that there will be training examples inside and outside of the sphere. As an additional source of diversity, the distances are calculated in a random subspace (a subset of the features). This random subspace is only used for defining the sphere, the two classifiers are trained using all the features.

In Random Subspaces [11] and Bagging each classifier is trained with a data set randomly obtained from the original training data. In these methods the used data sets are different because some information of the original training data is lost, diversity is obtained at the potential cost of decreased accuracy of the base classifiers. The objective of Random Oracles is to have diversity without losing information. The Random Oracle classifier is trained using all the attributes and instances, the diversity is obtained because the two classifiers in the Random Oracle are trained with different partitions of the training data. These ensemble methods can be used together.

It is also possible to use Random Oracles with more than two sub-regions, but in our experiments they do not improve the performance.

In this work, the distances are calculated according to the Euclidean distance, numerical attributes are scaled within [0,1]. For nominal attributes we consider that the distance is 0 or 1 depending if the two values are different or equal.

3 Experiments

3.1 Data Sets

Two collections of datasets were used. The HDDT collection[1] contains the binary imbalanced datasets used in [12]. The KEEL collection[2] contains the binary imbalanced datasets from the KEEL repository of [13].

Table 1 shows the characteristics of the 20 data sets in the HDDT collection and Table 2 the 66 data sets in the KEEL collection. Many data sets in these two collections are available or are modifications of data sets in the UCI Repository [14].

Table 1. Characteristics of the data sets from the HDDT collection.(#E: number of examples, #A: number of attributes (numeric/nominal), IR: imbalance ratio).

Data set	Examples	Attributes		Imbalance
		Numeric	Nominal	Ratio
boundary	3505	0	175	27.50
breast-y	286	0	9	2.36
cam	18916	0	132	19.08
compustat	13657	20	0	25.26
covtype	38500	10	0	13.02
credit-g	1000	7	13	2.33
estate	5322	12	0	7.37
german-numer	1000	24	0	2.33
heart-v	200	5	8	2.92
hypo	3163	7	18	19.95
ism	11180	6	0	42.00
letter	20000	16	0	24.35
oil	937	49	0	21.85
optdigits	5620	64	0	9.14
page	5473	10	0	8.77
pendigits	10992	16	0	8.63
phoneme	5404	5	0	2.41
PhosS	11411	480	0	17.62
satimage	6430	36	0	9.29
segment	2310	19	0	6.00

3.2 Settings

Weka [15] was used for the experiments. Unless explicitly specified, the parameters for the different methods take the default values given by Weka.

The decision tree method used for constructing the base classifiers was J48 (Weka's re-implementation of C4.5 [16]). As recommended for imbalanced data [12], it was used without pruning and collapsing but with Laplace smoothing at the leaves. C4.5 with this options is called C4.4 [17].

Several ensemble methods were considered. The first ensemble method is Bagging [18]. It was the best method for imbalanced data in [12] and [19].

[1] Available at `http://www.nd.edu/~dial/hddt/`
[2] Available at `http://sci2s.ugr.es/keel/imbalanced.php`

Table 2. Characteristics of the data sets from the KEEL collection. #E: number of examples, #N: number of numeric attributes, #D: number of discrete attributes, IR: imbalance ratio.

data set	#E	#N	#D	IR	data set	#E	#N	#D	IR
abalone19	4174	7	1	129.44	glass5	214	9	0	22.78
abalone9-18	731	7	1	16.40	glass6	214	9	0	6.38
cleveland-0_vs_4	177	13	0	12.62	haberman	306	3	0	2.78
ecoli-0-1-3-7_vs_2-6	281	7	0	39.14	iris0	150	4	0	2.00
ecoli-0-1-4-6_vs_5	280	6	0	13.00	led7digit-0-2-4-5-6-7-8-9_vs_1	443	7	0	10.97
ecoli-0-1-4-7_vs_2-3-5-6	336	7	0	10.59	new-thyroid1	215	5	0	5.14
ecoli-0-1-4-7_vs_5-6	332	6	0	12.28	new-thyroid2	215	5	0	5.14
ecoli-0-1_vs_2-3-5	244	7	0	9.17	page-blocks-1-3_vs_4	472	10	0	15.86
ecoli-0-1_vs_5	240	6	0	11.00	page-blocks0	5472	10	0	8.79
ecoli-0-2-3-4_vs_5	202	7	0	9.10	pima	768	8	0	1.87
ecoli-0-2-6-7_vs_3-5	224	7	0	9.18	segment0	2308	19	0	6.02
ecoli-0-3-4-6_vs_5	205	7	0	9.25	shuttle-c0-vs-c4	1829	9	0	13.87
ecoli-0-3-4-7_vs_5-6	257	7	0	9.28	shuttle-c2-vs-c4	129	9	0	20.50
ecoli-0-3-4_vs_5	200	7	0	9.00	vehicle0	846	18	0	3.25
ecoli-0-4-6_vs_5	203	6	0	9.15	vehicle1	846	18	0	2.90
ecoli-0-6-7_vs_3-5	222	7	0	9.09	vehicle2	846	18	0	2.88
ecoli-0-6-7_vs_5	220	6	0	10.00	vehicle3	846	18	0	2.99
ecoli-0_vs_1	220	7	0	1.86	vowel0	988	13	0	9.98
ecoli1	336	7	0	3.36	wisconsin	683	9	0	1.86
ecoli2	336	7	0	5.46	yeast-0-2-5-6_vs_3-7-8-9	1004	8	0	9.14
ecoli3	336	7	0	8.60	yeast-0-2-5-7-9_vs_3-6-8	1004	8	0	9.14
ecoli4	336	7	0	15.80	yeast-0-3-5-9_vs_7-8	506	8	0	9.12
glass-0-1-2-3_vs_4-5-6	214	9	0	3.20	yeast-0-5-6-7-9_vs_4	528	8	0	9.35
glass-0-1-4-6_vs_2	205	9	0	11.06	yeast-1-2-8-9_vs_7	947	8	0	30.57
glass-0-1-5_vs_2	172	9	0	9.12	yeast-1-4-5-8_vs_7	693	8	0	22.10
glass-0-1-6_vs_2	192	9	0	10.29	yeast-1_vs_7	459	7	0	14.30
glass-0-1-6_vs_5	184	9	0	19.44	yeast-2_vs_4	514	8	0	9.08
glass-0-4_vs_5	92	9	0	9.22	yeast-2_vs_8	482	8	0	23.10
glass-0-6_vs_5	108	9	0	11.00	yeast1	1484	8	0	2.46
glass0	214	9	0	2.06	yeast3	1484	8	0	8.10
glass1	214	9	0	1.82	yeast4	1484	8	0	28.10
glass2	214	9	0	11.59	yeast5	1484	8	0	32.73
glass4	214	9	0	15.46	yeast6	1484	8	0	41.40

As ensemble methods for imbalance the following were used: SMOTEBoost, SMOTEBagging and RUSBoost. SMOTEBoost has a parameter, the number of artificial instances to generate. Three values were considered: 100%, 200% and 500% of the number of instances in the minority class.

For each considered ensemble method there are three versions: one without Random Oracles, and two with Random Oracles: Linear and Spherical For all the ensembles the number of classifiers was 100.

The results were obtained with 5×2-fold cross validation [20]. Average ranks [21,22] were used for comparing several methods across several data sets.

3.3 Results

Table 3 shows the comparison of methods with and without Random Oracles, according to the AUROC. Each entry in the table shows the number of wins (W), ties (T) and losses (L) when comparing the method with the oracle (linear or spherical) with the original method. For 20 data sets, according to a two-tailed sign test at $\alpha = 0.05$ a classifier is significantly better than another if the

number of wins plus half the number of ties is at least 15 [21]. For the HDDT collection in all the considered pairs the number of wins is at least 15, with the only exception of RUSBoost and the Linear Oracle. For 66 data sets, the required number of wins (plus half the number of ties) is 41. For the KEEL collection the number of wins is at least 41.

Table 3. Comparison of methods with and without Random Oracles, according to the AUROC

Method	Linear Oracle		Spherical Oracle	
	HDDT W/T/L	KEEL W/T/L	HDDT W/T/L	KEEL W/T/L
Bagging	20/0/0	57/1/8	20/0/0	59/1/6
SMOTEBagging	20/0/0	61/2/3	20/0/0	61/2/3
SMOTEBoost (100%)	19/0/1	50/2/14	19/0/1	52/2/12
SMOTEBoost (200%)	16/0/4	53/2/11	18/0/2	56/2/8
SMOTEBoost (500%)	17/0/3	49/2/15	18/0/2	56/2/8
RUSBoost	14/0/6	41/2/23	16/0/4	44/2/20

Table 4 shows the comparison of methods with and without Random Oracles, according to the AUPRC. In all the cases the balance if favorable for the method with Random Oracles, in the great majority of the cases the balances are significant according to the sign test.

Table 4. Comparison of methods with and without Random Oracles, according to the AUPRC

Method	Linear Oracle		Spherical Oracle	
	HDDT W/T/L	KEEL W/T/L	HDDT W/T/L	KEEL W/T/L
Bagging	19/0/1	56/1/9	20/0/0	62/1/3
SMOTEBagging	19/0/1	55/2/9	20/0/0	59/2/5
SMOTEBoost (100%)	19/0/1	59/2/5	16/0/4	58/2/6
SMOTEBoost (200%)	14/0/6	56/2/8	16/0/4	60/2/4
SMOTEBoost (500%)	15/0/5	57/2/7	17/0/3	57/2/7
RUSBoost	16/0/4	39/2/25	15/0/5	41/2/23

Table 5 shows the comparison of methods with and without Random Oracles, according to the F-measure. In this case the advantage for the Random Oracles, compared with the other considered measures, is reduced. Nevertheless, the balance is never favorable for the method without Random Oracle.

Table 5. Comparison of methods with and without Random Oracles, according to the F-measure

Method	Linear Oracle		Spherical Oracle	
	HDDT W/T/L	KEEL W/T/L	HDDT W/T/L	KEEL W/T/L
Bagging	11/0/9	33/4/29	10/0/10	36/3/27
SMOTEBagging	15/0/5	51/4/11	17/0/3	50/4/12
SMOTEBoost (100%)	13/0/7	51/3/12	14/0/6	48/4/14
SMOTEBoost (200%)	12/0/8	52/3/11	15/0/5	51/4/11
SMOTEBoost (500%)	14/0/6	52/4/10	14/0/6	55/4/7
RUSBoost	12/0/8	42/1/23	13/0/7	43/3/20

Until now the results in this section show the comparison of a method and the corresponding method augmented with Random Oracles. They show clearly the advantage of using Oracles, but another interesting issue is what are the best methods among all the considered. Average ranks [21,22] are used to compare all these methods. For each data set all the methods are sorted, from best to worst, according to the considered performance measure. The best method has rank 1, the second rank two and so on. If several methods have the same performance they are assigned an average rank (e.g., if two methods have the best result, both have a rank of 1.5). The average rank of a method is calculated as the mean across all the data sets.

The Iman and Davenport Test is used to determine the presence of differences among all the compared methods. As a post-hoc procedure, using the best method as the control, the Hochberg procedure is used [22]. Table 6 shows the average ranks according to the AUROC. For the two collections of data sets, all the methods with Random Oracles are above all the methods without random oracles. The top positions are for SMOTEBoost with Spherical Random Oracles. In these tables a double horizontal line is used to indicate for which methods the Hochberg procedure rejects ($\alpha = 0.05$) the hypotheses.

Table 7 shows the average ranks according to the AUPRC. For the KEEL collection all the methods with Random Oracles have better ranks than all the methods without Random Oracles. This is not the case for the HDDT collection, due to the positions of RUSBoost with Random Oracles. Nevertheless, all the methods with Random Oracles have a better rank than the corresponding method without Random Oracles. The top positions are for SMOTEBoost with Random Oracles.

Table 8 shows the average ranks according to the F-measure. The advantage of Random Oracles is less clear than for the other measures, but still there is an advantage. For all the methods with Random Oracle the rank is better than the corresponding method without Random Oracles. The top positions are, for the two collections, for SMOTEBoost with Spherical Oracle.

Table 6. Average ranks according to the AUROC. The prefix "**L**-" is used for methods with Linear Random Oracle and "**S**-" for methods with Spherical Random Oracle.

(a) HDDT

Method	Ranking
S-SMOTEBoost (100%)	5.950
S-SMOTEBoost (200%)	6.350
L-SMOTEBoost (100%)	7.025
S-SMOTEBagging	7.400
L-SMOTEBoost (200%)	7.400
S-SMOTEBoost (500%)	7.425
L-SMOTEBoost (500%)	8.200
L-SMOTEBagging	8.650
S-Bagging	9.450
S-RUSBoost	9.450
L-Bagging	9.900
L-RUSBoost	10.400
SMOTEBoost (100%)	11.100
SMOTEBoost (200%)	11.300
RUSBoost	11.300
SMOTEBoost (500%)	11.950
SMOTEBagging	13.500
Bagging	14.250

(b) KEEL

Method	Ranking
S-SMOTEBoost (500%)	6.9242
S-SMOTEBoost (200%)	7.2803
S-SMOTEBagging	7.5303
L-SMOTEBoost (200%)	7.5758
S-RUSBoost	7.9470
S-Bagging	7.9848
L-SMOTEBagging	8.0833
L-SMOTEBoost (500%)	8.1742
S-SMOTEBoost (100%)	8.4470
L-Bagging	8.5303
L-SMOTEBoost (100%)	8.5606
L-RUSBoost	8.5909
RUSBoost	9.9621
SMOTEBoost (500%)	12.1061
SMOTEBoost (200%)	12.3485
SMOTEBoost (100%)	13.0758
SMOTEBagging	13.7727
Bagging	14.1061

Table 7. Average ranks according to the AUPRC

(a) HDDT

Method	Ranking
L-SMOTEBoost (100%)	6.05
S-SMOTEBoost (100%)	6.45
L-SMOTEBoost (200%)	6.85
S-SMOTEBagging	7.20
S-SMOTEBoost (200%)	7.25
S-SMOTEBoost (500%)	8.15
S-Bagging	8.70
L-SMOTEBoost (500%)	9.05
L-SMOTEBagging	9.10
L-Bagging	9.50
SMOTEBoost (200%)	10.00
SMOTEBoost (100%)	10.50
S-RUSBoost	10.85
L-RUSBoost	11.05
SMOTEBoost (500%)	12.05
RUSBoost	12.40
SMOTEBagging	12.55
Bagging	13.30

(b) KEEL

Method	Ranking
S-SMOTEBoost (500%)	7.2121
L-SMOTEBoost (200%)	7.2348
S-Bagging	7.2803
S-SMOTEBoost (200%)	7.3182
S-SMOTEBoost (100%)	7.5152
L-SMOTEBoost (100%)	7.5303
L-SMOTEBoost (500%)	7.8561
S-SMOTEBagging	7.9394
L-Bagging	8.2576
S-RUSBoost	8.4242
L-SMOTEBagging	8.7121
L-RUSBoost	8.9773
RUSBoost	10.3636
SMOTEBagging	13.2424
Bagging	13.1667
SMOTEBoost (200%)	13.1970
SMOTEBoost (500%)	13.2879
SMOTEBoost (100%)	13.4848

Table 8. Average ranks according to the F-measure

(a) HDDT

Method	Ranking
S-SMOTEBoost (500%)	6.200
S-SMOTEBoost (200%)	6.500
L-SMOTEBoost (500%)	6.950
L-SMOTEBoost (200%)	7.600
S-SMOTEBagging	8.050
L-SMOTEBoost (100%)	8.050
S-SMOTEBoost (100%)	8.750
S-RUSBoost	8.800
SMOTEBoost (500%)	8.900
SMOTEBoost (200%)	9.250
L-SMOTEBagging	9.350
L-RUSBoost	9.550
SMOTEBoost (100%)	9.650
RUSBoost	9.900
SMOTEBagging	11.200
L-Bagging	13.825
S-Bagging	14.125
Bagging	14.350

(b) KEEL

Method	Ranking
S-SMOTEBoost (500%)	6.3864
S-RUSBoost	7.1894
L-RUSBoost	7.5682
S-SMOTEBagging	7.6591
L-SMOTEBoost (200%)	7.7121
L-SMOTEBoost (500%)	7.2955
S-SMOTEBoost (200%)	8.1818
L-SMOTEBoost (100%)	8.6061
L-SMOTEBagging	8.7045
S-SMOTEBoost (100%)	9.1894
RUSBoost	9.6288
SMOTEBagging	10.8788
SMOTEBoost (500%)	10.9318
L-Bagging	11.4091
S-Bagging	11.7424
SMOTEBoost (200%)	12.0833
Bagging	12.8712
SMOTEBoost (100%)	12.9621

4 Diversity Study

One possible explanation for the improvements obtained with the use of Random Oracles is that they can introduce additional diversity in the base classifiers. This diversity is a necessary ingredient of successful ensembles. The other is the accuracy of the base classifiers, but if the classifiers are very accurate they cannot be very diverse.

Kappa-error diagrams [23] are used to represent the diversity (measured with κ) and error of the classifiers in an ensemble. Each pair of classifiers is represented as a point, the x axis is the diversity between the two classifiers, the y-axis is the average error of the two classifiers.

Figure 1 shows for two data sets these diagrams, for Bagging and **L**-Bagging. They also show the average diversity and error across all the pairs of classifiers.

Each κ-error diagram shows a single data set. For several data sets, their information can be summarized with κ-error relative movement diagram [24]. Figure 2 shows these diagrams for Bagging and **L**-Bagging. Each arrow represents a data set, the head coordinates are given by the differences between the average diversity and error of the two considered classifiers.

The number in the corners (e.g., 74, 1) indicate how many arrows go to the corresponding quadrant (e.g., top-left, right-bottom). From 85 data sets, in 84 cases the arrows go to the left, that is, the diversity is increased when using Bagging with linear oracles instead of Bagging without oracles. As expected, the

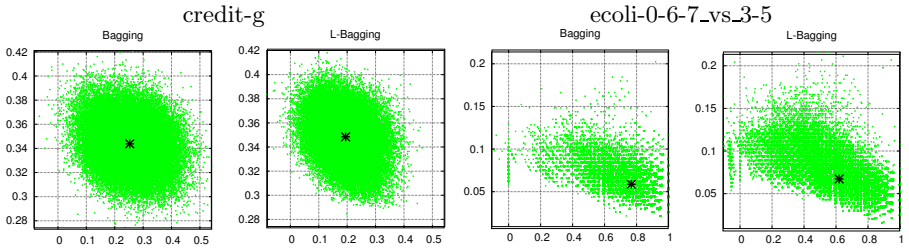

Fig. 1. Kappa error diagrams

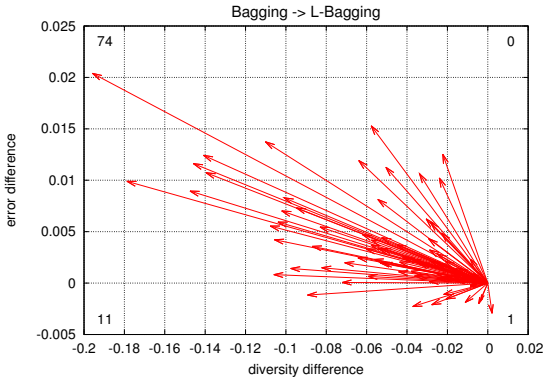

Fig. 2. Kappa error relative movement diagram

increased diversity usually (in 74 data sets) comes with increased error in the base classifiers.

5 Conclusions

The performance of Random Oracles have been studied when included in ensemble methods for imbalanced data. Four ensemble methods have been considered: Bagging, SMOTEBagging, SMOTEBoost and RUSBoost. According to the AUROC, the AUPRC and the F-Measure in two sets of data sets (HDDT and KEEL) including Random Oracles improves the results.

From the four ensemble methods, the best global results are for SMOTEBoost. This method has a parameter, the number (or percentage) of artificial instances of the minority class to generate. In this work three values have been considered, the results could improve if the parameter were adjusted for each data set.

The diversity of the base classifiers has been studied, Random Oracle usually improves this diversity. This can be the cause of the better performance of ensembles with Random Oracles.

As base classifiers, only decision trees have been used. Other base classifiers could give better results or affect in different ways the behaviour of the ensemble

methods. The performance of the different ensemble methods with other base classifiers can be studied in future work.

Acknowledgements. This work was supported by the Projects TIN2011-24046 and IPT-2011-1265-020000 of the Spanish Ministry of Economy and Competitiveness.

We wish to thank the developers of Weka [15]. We also express our gratitude to the donors of the different data sets.

References

1. Flach, P., Hernandez-Orallo, J., Ferri, C.: A coherent interpretation of AUC as a measure of aggregated classification performance. In: 28th International Conference on Machine Learning (ICML 2011), pp. 657–664. ACM (June 2011)
2. Davis, J., Goadrich, M.: The relationship between Precision-Recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine Learning, ICML 2006, pp. 233–240. ACM, New York (2006)
3. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., Herrera, F.: A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 42(4), 463–484 (2012)
4. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research 16, 321–357 (2002)
5. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: SMOTEBoost: Improving prediction of the minority class in boosting. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 107–119. Springer, Heidelberg (2003)
6. Wang, S., Yao, X.: Diversity analysis on imbalanced data sets by using ensemble models. In: IEEE Symposium Series on Computational Intelligence and Data Mining (IEEE CIDM 2009), pp. 324–331 (2009)
7. Seiffert, C., Khoshgoftaar, T., Van Hulse, J., Napolitano, A.: Rusboost: A hybrid approach to alleviating class imbalance. IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 40(1), 185–197 (2010)
8. Kuncheva, L.I., Rodríguez, J.J.: Classifier ensembles with a random linear oracle. IEEE Transactions on Knowledge and Data Engineering 19(4), 500–508 (2007)
9. Rodríguez, J.J., Kuncheva, L.I.: Naïve bayes ensembles with a random oracle. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 450–458. Springer, Heidelberg (2007)
10. Pardo, C., Rodríguez, J.J., Díez-Pastor, J.F., García-Osorio, C.: Random oracles for regression ensembles. In: Okun, O., Valentini, G., Re, M. (eds.) Ensembles in Machine Learning Applications. SCI, vol. 373, pp. 181–199. Springer, Heidelberg (2011)
11. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
12. Cieslak, D., Hoens, T., Chawla, N., Kegelmeyer, W.: Hellinger distance decision trees are robust and skew-insensitive. Data Mining and Knowledge Discovery 24(1), 136–158 (2012)

13. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S.: KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. Multiple-Valued Logic and Soft Computing 17(2-3), 255–287 (2011)
14. Frank, A., Asuncion, A.: UCI machine learning repository (2010), http://archive.ics.uci.edu/ml
15. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. SIGKDD Explorations 11(1) (2009)
16. Quinlan, J.R.: C4.5: Programs for Machine Learning. Machine Learning. Morgan Kaufmann, San Mateo (1993)
17. Provost, F., Domingos, P.: Tree induction for Probability-Based ranking. Machine Learning 52(3), 199–215 (2003)
18. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
19. Rodríguez, J.J., Díez-Pastor, J.F., García-Osorio, C., Santos, P.: Using model trees and their ensembles for imbalanced data. In: Lozano, J.A., Gámez, J.A., Moreno, J.A. (eds.) CAEPIA 2011. LNCS, vol. 7023, pp. 94–103. Springer, Heidelberg (2011)
20. Dietterich, T.G.: Approximate statistical test for comparing supervised classification learning algorithms. Neural Computation 10(7), 1895–1923 (1998)
21. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
22. García, S., Fernández, A., Luengo, J., Herrera, F.: Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences 180(10), 2044–2064 (2010)
23. Margineantu, D.D., Dietterich, T.G.: Pruning adaptive boosting. In: Proc. 14th International Conference on Machine Learning, pp. 211–218. Morgan Kaufmann (1997)
24. Maudes, J., Rodríguez, J.J., García-Osorio, C.: Disturbing neighbors diversity for decision forests. In: Okun, O., Valentini, G. (eds.) Applications of Supervised and Unsupervised Ensemble Methods. SCI, vol. 245, pp. 113–133. Springer, Heidelberg (2009)