SMOTE-D a Deterministic Version of SMOTE

Fredy Rodríguez Torres(✉), Jesús A. Carrasco-Ochoa,
and José Fco. Martínez-Trinidad

Instituto Nacional de Astrofísica, Óptica y Electrónica, Puebla, Mexico
{frodriguez,ariel,fmartine}@inaoep.mx

Abstract. Imbalanced data is a problem of current research interest. This problem arises when the number of objects in a class is much lower than in other classes. In order to address this problem several methods for oversampling the minority class have been proposed. Oversampling methods generate synthetic objects for the minority class in order to balance the amount of objects between classes, among them, SMOTE is one of the most successful and well-known methods. In this paper, we introduce a modification of SMOTE which deterministically generates synthetic objects for the minority class. Our proposed method eliminates the random component of SMOTE and generates different amount of synthetic objects for each object of the minority class. An experimental comparison of the proposed method against SMOTE in standard imbalanced datasets is provided. The experimental results show an improvement of our proposed method regarding SMOTE, in terms of *F-measure*.

Keywords: Imbalanced datasets · Oversampling · Supervised classification

1 Introduction

In supervised classification, imbalanced data arises when in some classes the number of objects is much lower than in other classes. Usually, the minority class (the class with the lowest amount of objects) is the class of interest in a class imbalance problem. However, when working on imbalanced datasets, classifiers tend to have a bias towards the majority class (the class with the largest amount of objects), resulting in poor classification performance for the minority class. This behavior is more notorious when the imbalance among classes is higher.

Some researchers have tackled the class imbalance problem [12,13] through oversampling methods [4–10]. Oversampling methods have the advantage of being independent of the classification method to be used, since they generate synthetic objects based only on the training set. SMOTE [2] is one of the most used and well known oversampling methods, which generates synthetic objects along the line segments joining objects in the minority class with some of their nearest neighbors. Thus, by increasing the amount of objects of the minority class, SMOTE tries to balance the amount of objects for all the classes. SMOTE

© Springer International Publishing Switzerland 2016
J.F. Martínez-Trinidad et al. (Eds.): MCPR 2016, LNCS 9703, pp. 177–188, 2016.
DOI: 10.1007/978-3-319-39393-3_18

has a random component for generating synthetic objects, producing a different result each time it is applied. Therefore, whether SMOTE is applied several times, choosing the best result becomes an issue.

In this paper, we propose a new oversampling method based on SMOTE, which computes in a deterministic way how many new synthetic objects should be generated from each object of the minority class and where these new objects should be placed. According to our experiments the proposed method performs better than SMOTE in different datasets with different imbalance level.

The rest of this document is organized as follows: in the Sect. 2, some related works are described; in the Sect. 3, the proposed method is introduced; in the Sect. 4, the experimental setup for the experiments is described; in the Sect. 5, the experimental results are shown; and in the Sect. 6, our conclusions and some future work directions are discussed.

2 Related Work

In the literature there are two types of extensions of SMOTE, those which combine SMOTE with other methods like noise filters (SMOTE-IPF) [6], subsampling methods (SMOTE-RSB*) [5] or feature selectors (E-SMOTE) [7]; and those that modify SMOTE like Bordeline-SMOTE [4], Safe-Level-SMOTE [8], SMOTE-OUT [9], SMOTE-COSINE [9] or Random-SMOTE [10]. Our work belongs to these last kind of methods. Thus we briefly describe some methods that modify SMOTE:

Borderline-SMOTE only oversamples the objects in the borderline of the minority class. First, it finds out the borderline objects of the minority class; then, synthetic objects are generated from these objects and they are added to the original training set. Borderline-SMOTE works as follows:

- For each object in the minority class its k nearest neighbors, from the whole training set, are calculated.
- If the k nearest neighbors contain objects from the minority and majority classes and the amount of its nearest neighbors in the majority class is larger than the amount of its nearest neighbors in the minority class, the object is considered as a borderline object.

For each borderline object, Borderline-SMOTE calculates its k nearest neighbors in the minority class and generates synthetic objects in the same way as SMOTE.

Safe-Level-Synthetic Minority Oversampling technique, assigns to each object in the minority class its safe level before generating synthetic objects. Each synthetic object is positioned closer to the largest safe level object, in this way all the synthetic objects are generated in safe regions. The safe level of an object is defined as the number of minority class objects among it's k nearest neighbors.

SMOTE-OUT randomly generates synthetic objects along the outside line of attributes between an object of the minority class and its nearest neighbor of the majority class. Once a synthetic object has been generated then SMOTE-OUT

finds out its nearest object in the minority class and randomly generates another synthetic object along the line of attributes between them.

SMOTE-Cosine works as SMOTE but it computes the k-nearest neighbors by voting using two distance metrics (Euclidean and Cosine).

Random-SMOTE generates temporally synthetic objects along the line of attributes between two objects of the minority class which are selected randomly. After, synthetic objects along the line of attributes between each temporal synthetic object and one object of the minority class are generated.

3 Proposed Method

The proposed method takes into account the distances between each object of the minority class and its k-nearest neighbors in the same class in order to determine how many synthetic objects should be generated from each object. For each object in the minority class, the higher its distance dispersion against its k-nearest neighbors, the higher the number of synthetic objects to generate. Additionally, the distance between an object and each one of its k-nearest neighbors is also taken into account individually to determine how many objects should be generated between each pair of objects. The larger the distance between a pairs of objects allows generating a greater amount of synthetic objects between them. The new synthetic objects are generated by dividing the difference in attributes between two objects by the number of objects to be generated between them. In this way, the synthetic objects will be created in a deterministic and uniform way.

For evaluating the dispersion of objects around each $object_i$ of the minority class we propose to use the standard deviation (σ_i) of the distances between the $object_i$ and its k nearest neighbors ($object_{ij}$ $j = 1, ..k$). We propose generating an amount of synthetic objects around each $object_i$ in the minority class such that it be proportional to the fraction of the standard deviation of the distances (σ_i) with respect to the sum of all the standard deviations of the distances computed for all the objects in the minority class ($\sum_{i=1}^{m} \sigma_i$, where m is the amount of objects in the minority class). Then, around objects of the minority class with higher standard deviation of distances, with respect to its k-nearest, neighbors more objects will be created.

After determining the proportion (p_i) of synthetic objects to generate from each object of the minority class, we proceed to calculate the proportion (p_{ij}) of synthetic objects to generate between each object of the minority class and each one of its k nearest neighbors. This proportion of synthetic objects is calculated as the fraction that represents the distance between the object and each nearest neighbor regarding to the sum of all the distances between the object and all its k nearest neighbors.

In the Fig. 1, we can see an example with three objects of the minority class and its 3-nearest neighbors (into the minority class), where $\sum_{i=1}^{m} \sigma_i = 2$. Here, the fraction ($p_1$) of σ_1 is $p_1 = \sigma_1 / \sum_{i=1}^{m} \sigma_i = 0.5$, therefore 50 % of the synthetic objects to be generated will be generated from $object_1$. For σ_2 the fraction is 30 % and 20 % for σ_3, If we have to generate a total of 10 synthetic objects, 5 would be generated around $object_1$, 3 around $object_2$, and 2 around $object_3$.

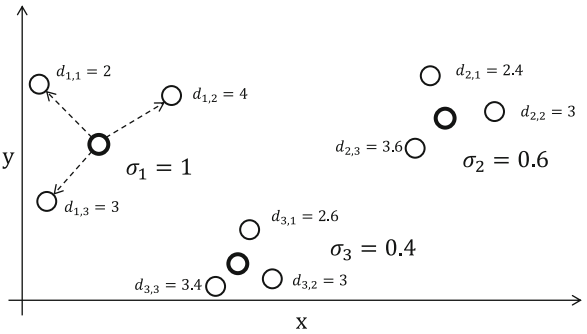

Fig. 1. Three objects of a minority class with distance values and standard deviation of distances for 3-nearest neighbors.

In order to determine the number of objects to generate ($s_{ij} = p_{ij} * p_i * n$) between an $object_i$ and each one of its nearest neighbors ($object_{ij}$) we take into account the amount of objects to generate for the minority class (n), this amount is given by the difference between the number of objects in the minority and majority class ($n = (M - m) * R$, where M is the number of objects in majority class, m is the number of objects in the minority class and R is a parameter defining the proportion of the difference to be reached).

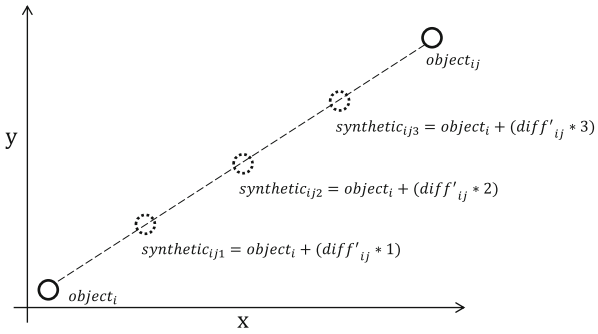

Fig. 2. Generation of 3 synthetic objects between an $object_i$ of the minority class and one of its nearest neighbors $object_{ij}$.

After calculating the amount of synthetic objects s_{ij} to generate between each $object_i$ an each one of its nearest neighbors ($object_{ij}, j = 1, \ldots, k$), attribute differences $diff_{ij}$ ($object_i, object_{ij}$) between $object_i$ and $object_{ij}$ are calculated. These differences are divided by the amount of synthetic objects to generate plus one, obtaining $diff'_{ij} = diff_{ij}/(s_{ij} + 1)$. This difference is added s_{ij} times to $object_i$; with each addition we obtain a new synthetic object and all the new synthetic objects are added to the original minority class. In the Fig. 2 we can

see an example of the generation of three synthetic objects between an object of the minority class $object_i$ and one of its k-nearest neighbors $object_{ij}$.

If for an object in the minority class the amount of objects to be generated regarding to the proportion representing the fraction of the standard deviation of the distances with respect to the sum of all standard deviations of distances represents less than 1 object, SMOTE-D does not generate synthetic objects from this object. The same happens if the proportion that represents the distance between the object and one of its nearest neighbors regarding to the sum of all the distances between the object and all its k nearest neighbors represents less than the 1 %.

Given a training set with M objects in the majority class and m objects in the minority class. The detailed procedure of SMOTE-D is as follows:

- Calculate the amount of objects to be generated for the minority class ($n = (M - m) * R$) according to a parameter $R \in [0, 1]$.
- Calculate the distances (d_{ij}) between each $object_i$ in the minority class and its k nearest neighbors, $j = 1, \ldots, k$ (k is a parameter).
- Calculate the standard deviation (σ_i) of the distances between each $object_i$ and its k-nearest neighbors.
- Calculate for each $object_i$ the fraction (p_i) of its standard deviation (σ_i) from the total sum of all standard deviations as $p_i = \sigma_i / \sum_{i=1}^{m} \sigma_i$.
- Calculate the fraction (p_{ij}) of each distance (d_{ij}) with respect to the sum of distances of each $object_i$ and its k nearest neighbors as $p_{ij} = d_{ij} / \sum_{j=1}^{k} d_{ij}$.
- Calculate the number of objects (s_{ij}) to generate between an $object_i$ and one of its nearest neighbors $object_{ij}$ as $s_{ij} = p_i * p_{ij} * n$.
- Get the attribute difference $diff_{ij}$ between an $object_i$ and each one of its k nearest neighbors $object_{ij}$ as $diff_{ij} = object_{ij} - object_i; j = 1, \ldots, k$.
  - Divide the difference between an object and each one of its neighbors by the amount of synthetic objects to be generated from this pair plus 1. ($diff'_{ij} = diff_{ij}/(s_{ij} + 1)$)
  - Add the difference $diff'_{ij}$ to the object of the minority class as many times as objects to generate (s_{ij}).
- Add the generated synthetic objects to the minority class.

4 Experimental Setup

For evaluating the proposed method, we use 66 datasets taken from the repository KEEL [1], we used 5-fold cross validation. In order to measure the degree of imbalance in a dataset, in the literature, the imbalance ratio (IR) is commonly used. The IR of a dataset with M objects in the majority class and m objects in the minority class is computed as $IR = M/m$. All the datasets used in our experiments are from binary problems with numeric attributes, and IR ranging from 1.82 to 129.44 (see Table 1).

Into the KEEL repository these 66 datasets are provided together with the results of applying SMOTE with $k = 5$ as the number of nearest neighbors,

Table 1. Sumary of the datasets used in our experiments.

Name	# attributes	# objects	IR	Name	# attributes	# objects	IR
glass1	9	214	1.82	glass-0-4_vs_5	9	92	9.22
ecoli-0_vs_1	7	220	1.86	ecoli-0-3-4-6_vs_5	7	205	9.25
wisconsin	9	683	1.86	ecoli-0-3-4-7_vs_5-6	7	257	9.28
pima	8	768	1.87	yeast-0-5-6-7-9_vs_4	8	528	9.35
iris0	4	150	2.00	vowel0	13	988	9.98
glass0	9	214	2.06	ecoli-0-6-7_vs_5	6	220	10.00
yeast1	8	1484	2.46	glass-0-1-6_vs_2	9	192	10.29
haberman	3	306	2.78	ecoli-0-1-4-7_vs_2-3-5-6	7	336	10.59
vehicle2	18	846	2.88	led7digit-0-2-4-5-6-7-8-9_vs_1	7	443	10.97
vehicle1	18	846	2.90	glass-0-6_vs_5	9	108	11.00
vehicle3	18	846	2.99	ecoli-0-1_vs_5	6	240	11.00
glass-0-1-2-3_vs_4-5-6	9	214	3.20	glass-0-1-4-6_vs_2	9	205	11.06
vehicle0	18	846	3.25	glass2	9	214	11.59
ecoli1	7	336	3.36	ecoli-0-1-4-7_vs_5-6	6	332	12.28
new-thyroid1	5	215	5.14	cleveland-0_vs_4	13	177	12.62
new-thyroid2	5	215	5.14	ecoli-0-1-4-6_vs_5	6	280	13.00
ecoli2	7	336	5.46	shuttle-c0-vs-c4	9	1829	13.87
segment0	19	2308	6.02	yeast-1_vs_7	7	459	14.30
glass6	9	214	6.38	glass4	9	214	15.47
yeast3	8	1484	8.10	ecoli4	7	336	15.80
ecoli3	7	336	8.60	page-blocks-1-3_vs_4	10	472	15.86
page-blocks0	10	5472	8.79	abalone9-18	8	731	16.40
ecoli-0-3-4_vs_5	7	200	9.00	glass-0-1-6_vs_5	9	184	19.44
yeast-2_vs_4	8	514	9.08	shuttle-c2-vs-c4	9	129	20.50
ecoli-0-6-7_vs_3-5	7	222	9.09	yeast-1-4-5-8_vs_7	8	693	22.10
ecoli-0-2-3-4_vs_5	7	202	9.10	glass5	9	214	22.78
glass-0-1-5_vs_2	9	172	9.12	yeast-2_vs_8	8	482	23.10
yeast-0-3-5-9_vs_7-8	8	506	9.12	yeast4	8	1484	28.10
yeast-0-2-5-7-9_vs_3-6-8	8	1004	9.14	yeast-1-2-8-9_vs_7	8	947	30.57
yeast-0-2-5-6_vs_3-7-8-9	8	1004	9.14	yeast5	8	1484	32.73
ecoli-0-4-6_vs_5	6	203	9.15	ecoli-0-1-3-7_vs_2-6	7	281	39.14
ecoli-0-1_vs_2-3-5	7	244	9.17	yeast6	8	1484	41.40
ecoli-0-2-6-7_vs_3-5	7	224	9.18	abalone19	8	4174	129.44

HVDM [3] as distance function and an oversampling rate N such that $IR = 0.0$. Thus, for our experiments SMOTE-D was configured in the same way ($k = 5$, HVDM and $R = 1$ in order to get $IR = 0.0$).

For comparing the results of SMOTE-D and SMOTE, decision trees, support vector machines (SVM) and KNN with $k = 5$ were used. We applied SMOTE-D in all datasets and compare the classification results against those obtained by using SMOTE.

One of the measures commonly used for assessing the quality of classifiers on imbalanced datasets is the *F-measure* [11]. Therefore, in our experiments we used this measure to asses our results, additionally we applied a t-test with statistical significant difference at the 5 % of significance level between the results of SMOTE and SMOTE-D.

5 Experimental Results

The results of comparing SMOTE-D and SMOTE with different classifiers in terms of *F-measure* are shown in Tables 2 and 3. In both tables, the first column shows the name of the databases, the following columns show the classification

Table 2. Results of applying the evaluated classifiers over the results of SMOTE and SMOTE-D over the tested datasets using HVDM metric distance.

Dataset name	Decision tree		KNN		SVM	
	SMOTE	SMOTE-D	SMOTE	SMOTE-D	SMOTE	SMOTE-D
glass1	0.6910	**0.6967**	0.7502	**0.7575**	0.5693	**0.5746**
ecoli-0_vs_1	0.9744	**0.9822**	**0.9715**	0.9675	**0.9861**	**0.9861**
wisconsin	0.9315	**0.9444**	**0.9501**	0.9456	**0.9590**	0.9527
pima	**0.6073**	0.5533	**0.5692**	0.5378	**0.6699**	0.6664
iris0	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
glass0	**0.6904**	0.5935	**0.7073**	0.6861	0.6464	**0.6691**
yeast1	0.5226	**0.5278**	**0.5240**	0.4833	0.5859	**0.5885**
haberman	0.3670	**0.3755**	**0.4026**	0.3723	**0.4506**	0.4147
vehicle2	**0.9343**	0.8946	0.8605	**0.8712**	0.9212	**0.9329**
vehicle1	**0.5591**	0.5183	0.4265	**0.4385**	0.6581	**0.6651**
vehicle3	0.5283	**0.5630**	0.4555	**0.4701**	**0.6267**	0.6257
glass-0-1-2-3_vs_4-5-6	**0.8549**	0.8496	0.8650	**0.9197**	0.8378	**0.8688**
vehicle0	0.8295	**0.8812**	0.8434	**0.8658**	**0.9220**	0.9167
ecoli1	**0.7501**	0.7369	**0.7366**	0.7282	**0.7788**	0.7761
new-thyroid1	**0.9448**	0.8959	**0.9581**	**0.9581**	**0.9733**	0.9713
new-thyroid2	**0.9617**	0.8694	0.9600	**1.0000**	0.9617	**0.9713**
ecoli2	0.6610	**0.7394**	**0.8623**	0.8124	0.7192	**0.7402**
segment0	**0.9786**	0.9728	0.9701	**0.9715**	0.9822	**0.9850**
glass6	**0.7719**	0.7178	0.8469	**0.8621**	0.7872	**0.8256**
yeast3	0.7320	**0.7447**	0.6700	**0.6893**	0.6585	**0.7076**
ecoli3	0.5078	**0.6084**	0.5837	**0.6054**	0.5916	**0.6872**
page-blocks0	0.7927	**0.8303**	0.7653	**0.7982**	**0.1747**	0.1603
ecoli-0-3-4_vs_5	0.7597	**0.7921**	0.6678	**0.8000**	0.6854	**0.7351**
yeast-2_vs_4	0.6642	**0.7371**	0.7092	**0.7343**	0.6999	**0.7423**
ecoli-0-6-7_vs_3-5	0.5044	**0.6521**	0.6033	**0.7460**	0.4835	**0.5442**
ecoli-0-2-3-4_vs_5	0.6732	**0.8100**	0.6946	**0.7937**	0.6745	**0.7254**
glass-0-1-5_vs_2	**0.3464**	0.2244	**0.2956**	0.2817	**0.1489**	0.0000
yeast-0-3-5-9_vs_7-8	**0.3314**	0.3188	**0.4597**	0.4146	**0.3606**	0.3360
yeast-0-2-5-7-9_vs_3-6-8	0.4249	**0.4673**	0.4676	**0.4964**	0.5197	**0.5941**
yeast-0-2-5-6_vs_3-7-8-9	**0.7654**	0.6747	**0.6954**	0.6648	0.7004	**0.7766**
ecoli-0-4-6_vs_5	0.6057	**0.6797**	0.7461	**0.7784**	0.6844	**0.7695**
ecoli-0-1_vs_2-3-5	0.6005	**0.6523**	0.5571	**0.6587**	0.5690	**0.6945**
ecoli-0-2-6-7_vs_3-5	**0.6255**	0.6094	**0.7034**	0.6714	0.5710	**0.7052**

Table 2. (*Continued*)

Dataset name	Decision tree		KNN		SVM	
	SMOTE	SMOTE-D	SMOTE	SMOTE-D	SMOTE	SMOTE-D
glass-0-4_vs_5	0.8800	**0.9333**	0.8743	**1.0000**	0.7886	**0.8933**
ecoli-0-3-4-6_vs_5	0.7429	**0.7562**	0.7621	**0.8548**	0.7168	**0.7529**
ecoli-0-3-4-7_vs_5-6	0.6582	**0.7740**	0.6472	**0.7321**	0.6524	**0.7349**
yeast-0-5-6-7-9_vs_4	0.4268	**0.4624**	**0.4760**	0.4061	0.4509	**0.5086**
vowel0	**0.8989**	0.8959	**1.0000**	**1.0000**	**0.8131**	0.7696
ecoli-0-6-7_vs_5	0.5600	**0.6871**	0.6793	**0.6976**	0.5814	**0.7611**
glass-0-1-6_vs_2	0.3174	**0.3333**	0.2256	**0.3665**	**0.1382**	0.0000
ecoli-0-1-4-7_vs_2-3-5-6	0.6229	**0.7140**	0.6349	**0.7228**	0.5956	**0.6845**
led7digit-0-2-4-5-6-7-8-9_vs_1	0.7450	**0.7766**	0.3622	**0.5133**	0.5363	**0.6542**
glass-0-6_vs_5	0.6167	**0.7714**	0.6760	**0.7792**	0.5797	**0.6358**
ecoli-0-1_vs_5	0.7933	**0.9333**	0.8600	**0.9200**	0.6489	**0.7200**
glass-0-1-4-6_vs_2	0.3456	**0.3837**	0.2734	**0.2939**	**0.1931**	0.1435
glass2	0.3825	**0.4089**	**0.3262**	0.2876	**0.1806**	0.0500
ecoli-0-1-4-7_vs_5-6	0.5765	**0.6737**	0.6781	**0.7766**	0.5235	**0.7065**
cleveland-0_vs_4	0.4544	**0.5410**	**0.1653**	0.1500	0.5367	**0.6324**
ecoli-0-1-4-6_vs_5	0.6106	**0.6133**	0.6368	**0.8159**	0.6240	**0.6444**
shuttle-c0-vs-c4	**1.0000**	0.9884	**0.9959**	0.9841	**0.9959**	**0.9959**
yeast-1_vs_7	0.2685	**0.2900**	0.3047	**0.3574**	0.2939	**0.3036**
glass4	**0.6276**	0.5743	0.7833	**0.7881**	0.6308	**0.6558**
ecoli4	0.6543	**0.7790**	0.8148	**0.8492**	0.6328	**0.8047**
page-blocks-1-3_vs_4	0.9110	**0.9667**	0.7231	**0.8013**	0.3535	**0.4013**
abalone9-18	0.2573	**0.4583**	0.3176	**0.4992**	0.3684	**0.6419**
glass-0-1-6_vs_5	0.5619	**0.6800**	**0.6933**	0.5943	0.4870	**0.5743**
shuttle-c2-vs-c4	**1.0000**	0.9333	**1.0000**	**1.0000**	**1.0000**	**1.0000**
yeast-1-4-5-8_vs_7	0.1393	**0.2179**	**0.1880**	0.1791	0.1276	**0.1490**
glass5	**0.7867**	0.7276	**0.6933**	0.6076	0.4918	**0.7156**
yeast-2_vs_8	0.4653	**0.5418**	**0.4879**	0.4599	0.5610	**0.6681**
yeast4	0.2915	**0.3118**	**0.3446**	0.3190	0.2812	**0.3117**
yeast-1-2-8-9_vs_7	**0.1678**	0.1473	0.1748	**0.2012**	0.1428	**0.1726**
yeast5	0.5546	**0.6338**	0.6953	**0.7120**	0.4709	**0.5003**
ecoli-0-1-3-7_vs_2-6	0.2833	**0.5400**	0.3667	**0.5333**	0.2183	**0.3994**
yeast6	0.3706	**0.4041**	**0.4743**	0.4324	0.2647	**0.3465**
abalone19	**0.0515**	0.0235	0.0293	**0.0353**	0.0466	**0.0587**
Average	0.6199	**0.6513***	0.6310	**0.6583***	0.5831	**0.6257***

Table 3. Results of applying the evaluated classifiers over the results of SMOTE and SMOTE-D over the tested datasets using euclidean metric distance.

Dataset name	Decision tree		KNN		SVM	
	SMOTE	SMOTE-D	SMOTE	SMOTE-D	SMOTE	SMOTE-D
glass1	0.6469	**0.6511**	**0.7642**	0.6511	0.5688	**0.6511**
ecoli-0_vs_1	0.9618	**0.9862**	0.9474	**0.9862**	0.9659	**0.9862**
wisconsin	0.9253	**0.9338**	**0.9475**	0.9338	**0.9572**	0.9338
pima	**0.5908**	0.5762	0.5552	**0.5762**	**0.6658**	0.5762
iris0	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
glass0	0.6838	**0.7161**	0.7156	**0.7161**	0.6533	**0.7161**
yeast1	**0.5280**	0.5269	0.5042	**0.5269**	**0.5790**	0.5269
haberman	0.3707	**0.3807**	0.3645	**0.3807**	**0.4349**	0.3807
vehicle2	0.9167	**0.9193**	0.8672	**0.9193**	0.9186	**0.9193**
vehicle1	**0.5339**	0.5175	0.4399	**0.5175**	**0.6656**	0.5175
vehicle3	0.5443	**0.5608**	0.4470	**0.5608**	**0.6268**	0.5608
glass-0-1-2-3_vs_4-5-6	0.8370	**0.8402**	**0.9114**	0.8402	**0.8423**	0.8402
vehicle0	**0.8626**	0.8508	**0.8591**	0.8508	**0.9292**	0.8508
ecoli1	0.7658	**0.7700**	0.7488	**0.7700**	**0.7759**	0.7700
new-thyroid1	**0.9228**	0.8979	**0.9713**	0.8979	**0.9713**	0.8979
new-thyroid2	**0.9027**	0.8739	**0.9920**	0.8739	**0.9713**	0.8739
ecoli2	0.7470	**0.7645**	**0.7980**	0.7645	0.7259	**0.7645**
segment0	**0.9816**	0.9743	**0.9759**	0.9743	**0.9894**	0.9743
glass6	0.7314	**0.8221**	**0.8517**	0.8221	**0.8315**	0.8221
yeast3	**0.7300**	0.7195	0.6957	**0.7195**	0.6755	**0.7195**
ecoli3	**0.5287**	0.5086	**0.6249**	0.5086	**0.5862**	0.5086
page-blocks0	0.8173	**0.8474**	0.7904	**0.8474**	0.2182	**0.8474**
ecoli-0-3-4_vs_5	**0.7626**	0.7042	**0.7871**	0.7042	0.6815	**0.7042**
yeast-2_vs_4	**0.6963**	0.6934	**0.7361**	0.6934	**0.7011**	0.6934
ecoli-0-6-7_vs_3-5	**0.6051**	0.5535	**0.7040**	0.5535	**0.5937**	0.5535
ecoli-0-2-3-4_vs_5	**0.7254**	**0.7254**	**0.7725**	0.7254	0.7058	**0.7254**
glass-0-1-5_vs_2	0.3150	**0.4103**	0.3091	**0.4103**	0.1648	**0.4103**
yeast-0-3-5-9_vs_7-8	0.3031	**0.3903**	0.3375	**0.3903**	0.3641	**0.3903**
yeast-0-2-5-7-9_vs_3-6-8	0.4661	**0.4996**	0.4747	**0.4996**	**0.5223**	0.4996
yeast-0-2-5-6_vs_3-7-8-9	0.7641	**0.7690**	0.7180	**0.7690**	0.6997	**0.7690**
ecoli-0-4-6_vs_5	**0.7176**	0.6478	**0.7837**	0.6478	**0.7664**	0.6478
ecoli-0-1_vs_2-3-5	0.6160	**0.6796**	**0.6887**	0.6796	0.6738	**0.6796**
ecoli-0-2-6-7_vs_3-5	**0.6340**	0.6157	**0.7284**	0.6157	**0.6418**	0.6157
glass-0-4_vs_5	**0.9333**	**0.9333**	**1.0000**	0.9333	0.8793	**0.9333**
ecoli-0-3-4-6_vs_5	0.7235	**0.7862**	**0.8498**	0.7862	**0.8014**	0.7862

Table 3. (*Continued*)

Dataset name	Decision tree		KNN		SVM	
	SMOTE	SMOTE-D	SMOTE	SMOTE-D	SMOTE	SMOTE-D
ecoli-0-3-4-7_vs_5-6	0.6427	**0.6567**	**0.7390**	0.6567	**0.6845**	0.6567
yeast-0-5-6-7-9_vs_4	0.4379	**0.4812**	0.4579	**0.4812**	0.4485	**0.4812**
vowel0	**0.8902**	0.8577	**1.0000**	0.8577	0.8095	**0.8577**
ecoli-0-6-7_vs_5	0.6841	**0.7743**	0.7245	**0.7743**	0.7060	**0.7743**
glass-0-1-6_vs_2	**0.2728**	0.1602	**0.2564**	0.1602	**0.1673**	0.1602
ecoli-0-1-4-7_vs_2-3-5-6	0.6514	**0.7260**	0.6895	**0.7260**	0.6189	**0.7260**
led7digit-0-2-4-5-6-7-8-9_vs_1	**0.7625**	0.7614	0.5207	**0.7614**	0.6838	**0.7614**
glass-0-6_vs_5	0.6969	**0.7702**	0.7685	**0.7702**	0.6989	**0.7702**
ecoli-0-1_vs_5	0.8667	**0.9333**	0.9200	**0.9333**	0.8200	**0.9333**
glass-0-1-4-6_vs_2	**0.4101**	0.3417	**0.3445**	0.3417	0.1926	**0.3417**
glass2	**0.3580**	0.2416	**0.2876**	0.2416	0.1844	**0.2416**
ecoli-0-1-4-7_vs_5-6	0.6811	**0.6911**	**0.7468**	0.6911	0.6066	**0.6911**
cleveland-0_vs_4	**0.6593**	0.6181	0.1479	**0.6181**	0.5657	**0.6181**
ecoli-0-1-4-6_vs_5	**0.5986**	0.5739	**0.7833**	0.5739	**0.6080**	0.5739
shuttle-c0-vs-c4	**1.0000**	0.9920	**0.9955**	0.9920	**1.0000**	0.9920
yeast-1_vs_7	**0.2823**	0.2624	0.2613	**0.2624**	**0.2804**	0.2624
glass4	**0.7019**	0.5500	**0.8155**	0.5500	**0.6027**	0.5500
ecoli4	0.7221	**0.7312**	**0.8068**	0.7312	**0.7801**	0.7312
page-blocks-1-3_vs_4	**0.9667**	0.9624	0.7951	**0.9624**	0.3344	**0.9624**
abalone9-18	0.4217	**0.4974**	**0.5117**	0.4974	**0.5074**	0.4974
glass-0-1-6_vs_5	0.6600	**0.7200**	**0.7533**	0.7200	0.6953	**0.7200**
shuttle-c2-vs-c4	**0.9333**	**0.9333**	**1.0000**	0.9333	**1.0000**	0.9333
yeast-1-4-5-8_vs_7	0.0984	**0.1314**	0.1089	**0.1314**	**0.1328**	0.1314
glass5	0.7600	**0.7733**	0.6476	**0.7733**	0.7538	**0.7733**
yeast-2_vs_8	**0.4217**	0.3800	0.3424	**0.3800**	**0.5369**	0.3800
yeast4	0.3273	**0.3329**	0.3152	**0.3329**	0.2891	**0.3329**
yeast-1-2-8-9_vs_7	0.1673	**0.1753**	0.1092	**0.1753**	0.1386	**0.1753**
yeast5	**0.7110**	0.6051	**0.7208**	0.6051	0.4720	**0.6051**
ecoli-0-1-3-7_vs_2-6	0.5054	**0.5889**	0.5133	**0.5889**	0.2974	**0.5889**
yeast6	**0.4023**	0.3911	0.3613	**0.3911**	0.2996	**0.3911**
abalone19	0.0434	**0.0451**	**0.0582**	0.0451	**0.0511**	0.0451
Average	0.6444	**0.6470**	**0.6540**	0.6470	0.6169	**0.6470**

results, in terms of *F-measure*, for decision tree, KNN and SVM classifiers, respectively. In the last row, the average over the 66 datasets for each classifier is shown. The best results of *F-measure* appear boldfaced. Datasets with (*) are those where the t-test showed statistical significant difference at the 5 % of significance level.

In the results that appear in the Table 2, the proposed method obtains a better performance in 67 % of the datasets using decision trees, in 61 % using KNN, and 73 % using SVM. In average the results show a statistical improvement of 3.14 %, 2.73 % and 4.26 % in terms of *F-measure* for decision trees, KNN and SVM respectively. Considering only those datasets with an IR greater than 15.8 the results obtained by all the classifiers when SMOTE-D is applied are statistical significant better for all used classifiers, these databases can be seen with the names in bold.

In the results that appear in the Table 3, the proposed method obtains a better performance in 50 % of the datasets using decision trees, in 46 % using KNN, and 50 % using SVM. In average the results do not show a statistical difference for all classifiers. Considering only those datasets with an IR greater than 10.59 the results obtained by SVM classifier when SMOTE-D is applied are statistical significant better for SVM classifier, these databases can be seen with the names in bold

6 Conclusions

This paper introduces a new oversampling method, SMOTE-D, which is a deterministic version of SMOTE. Comparisons against SMOTE in terms of *F-measure* using decision trees, KNN and SVM classifiers show that SMOTE-D get better results than SMOTE. These results give evidence that estimating the dispersion of the objects of the minority class (based on the standard deviation of distances) to determine how many objects should be generated around each object in the minority class and how many objects should be created between each object and its nearest neighbors, together with a deterministic and uniform creation of the synthetic objects, allows an oversampling of the minority class such that better results than SMOTE can be obtained.

From our experiments, we can conclude that when a dataset has an imbalance ratio higher than 10.0, then SMOTE-D,using either Euclidean or HVD distance, performs better than SMOTE.

As future work, we are going to extend the proposed method for working with nominal attributes.

Acknowledgment. This work was partly supported by National Council of Science and Technology of Mexico under the scholarship grant 627301.

References

1. Alcalá-Fdez, J., Fernandez, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J. Multiple-Valued Logic Soft Comput. **17**(2–3), 255–287 (2011)
2. Chawla, N.V., et al.: SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. **16**, 321–357 (2002)
3. Wilson, D., Randall Martinez, T.R.: Improved heterogeneous distance functions. J. Artif. Intell. Res. **6**, 1–34 (1997)
4. 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. LNCS, vol. 3644, pp. 878–887. Springer, Heidelberg (2005)
5. Ramentol, E., et al.: SMOTE-RSB*: a hybrid preprocessing approach based on over-sampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. Knowl. Inf. Syst. **33**(2), 245–265 (2012)
6. Sáez, J.A., et al.: SMOTE IPF: addressing the noisy and borderline examples problem in imbalanced classification by a re-sampling method with filtering. Inf. Sci. **291**, 184–203 (2015)
7. Deepa, T., Punithavalli, M.: An E-SMOTE technique for feature selection in high-dimensional imbalanced dataset. In: 2011 3rd International Conference on Electronics Computer Technology (ICECT), vol. 2. IEEE (2011)
8. 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)
9. Koto, F.: SMOTE-OUT, SMOTE-COSINE, and selected-SMOTE: an enhancement strategy to handle imbalance in data level. In: 2014 International Conference on Advanced Computer Science and Information Systems (ICACSIS). IEEE (2014)
10. Dong, Y., Wang, X.: A new over-sampling approach: random-SMOTE for learning from imbalanced data sets. In: Xiong, H., Lee, W.B. (eds.) KSEM 2011. LNCS, vol. 7091, pp. 343–352. Springer, Heidelberg (2011)
11. Larsen, B., Aone, C.: Fast and effective text mining using linear-time document clustering. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM (1999)
12. Shakiba, N., Rueda, L.: MicroRNA identification using linear dimensionality reduction with explicit feature mapping. In: BMC Proceedings. BioMed Central (2013)
13. Batuwita, R., Palade, V.: Adjusted geometric-mean: a novel performance measure for imbalanced bioinformatics datasets learning. J. Bioinf. Comput. Biol. **10**(04), 1250003 (2012)