Classification on Imbalanced Data Sets, Taking Advantage of Errors to Improve Performance

Asdrúbal López-Chau[1(✉)], Farid García-Lamont[2], and Jair Cervantes[2]

[1] Centro Universitario UAEM, Universidad Autónoma Del Estado de México, CP 55600 Zumpango, Estado de Mexico, México
alchau@uaemex.mx
[2] Centro Universitario UAEM, Universidad Autónoma Del Estado de México, 56159 Texcoco, Estado de Mexico, México

Abstract. Classification methods usually exhibit a poor performance when they are applied on imbalanced data sets. In order to overcome this problem, some algorithms have been proposed in the last decade. Most of them generate synthetic instances in order to balance data sets, regardless the classification algorithm. These methods work reasonably well in most cases; however, they tend to cause over-fitting.

In this paper, we propose a method to face the imbalance problem. Our approach, which is very simple to implement, works in two phases; the first one detects instances that are difficult to predict correctly for classification methods. These instances are then categorized into "noisy" and "secure", where the former refers to those instances whose most of their nearest neighbors belong to the opposite class. The second phase of our method, consists in generating a number of synthetic instances for each one of those that are difficult to predict correctly. After applying our method to data sets, the AUC area of classifiers is improved dramatically. We compare our method with others of the state-of-the-art, using more than 10 data sets.

Keywords: Imbalanced · Classification · Synthetic instances

1 Introduction

Achieving a good performance on imbalanced data sets is a challenging task for classification methods [3]. They usually focus on majority class, almost ignoring the opposite class [8]. Currently, there are many real-world applications that generate this type of data sets, for example: software defect detection [6], medical diagnosis [1], fraud detection in telecommunications [4], financial risks [7] and DNA sequencing [9], among others. In this type of applications, there are two objectives in conflict, on the one hand, for the classifier should be more important to predict the minority class instances with the minimal errors, and on the other hand, the classification accuracy for majority class instances should not be severely damaged. The AUC ROC measure is one of the most widely used to capture this requirement.

The problem of classification on imbalanced data sets has attracted the attention of the machine learning and data mining communities in the last past few years [2]. The state-of-the-art methods to deal this problem can be categorized into:

D.-S. Huang and K. Han (Eds.): ICIC 2015, Part III, LNAI 9227, pp. 72–78, 2015.
DOI: 10.1007/978-3-319-22053-6_8

(1) external methods, which pre-processes the data sets to balance them before applying a classification method;
(2) internal methods, which modify the algorithms to make them more suitable to this problem;
(3) ensembles, that use two or more classifiers and then combine their outputs to predict the class;
(4) cost-sensitive methods, which use cost matrices to penalize misclassification, or
(5) other methods, that include combinations of the strategies mentioned, and application of genetic algorithms.

External methods work at the data level, regardless the classifier to be used. These methods are based on two main techniques: *under-sampling* and *over-sampling*, both of them balance the data sets, either by removing objects from the majority class or inserting synthetic minority class objects, respectively. One of the most representative methods is SMOTE. It balances data sets by creating synthetic instances between the line that joins a minority class instance and their nearest neighbors. Variants of SMOTE guide the creation of minority instances towards specific parts of the input space, considering characteristics of the data such as density of minority class instances, the decision boundaries or using ensembles of classifiers.

In this paper, we propose a method to pre-process imbalanced data sets for classification. It works in two phases: the first one identifies instances, which are difficult to predict for a classification method. These instances are important because represent regions in the input space where the classifier is unable to perform adequately, and therefore, it is necessary to clarify the concepts or sub-concepts by generating synthetic instances in such regions. The instances that are difficult to predict, are categorized into "noisy" and "secure" instances, where the former refers to those which most of their nearest neighbors belong to the opposite class. Noisy instances are usually near to decision boundaries, or in overlapped class regions [5]. The second phase of our method, consists in generating a number of synthetic instances considering the noisy ones. Depending on the imbalance ratio, the number of generated instances is adapted. We tested our method on 11 data sets, and compare the performance of C4.5 classifier using other balancing algorithms. According to the results, AUC is improved significantly in most cases.

The rest of this paper is organized as follows. Our proposal is shown in detail in Sect. 2. The experiments, results and a discussion is shown in Sect. 3. The conclusions and references are in the last part of this paper.

2 Method Based on Observations of Errors

The method presented in this paper is effective and very easy to implement. Different from SMOTE and other similar algorithms that generate instances regardless the classification method or class distributions, our approach takes advantage of observations about the correctness of predictions. These are used to identify difficult regions of the input space, and then the generation of synthetic instances focuses on such regions.

Given an imbalanced data set: $X = \{(x_i, y_i)_{i=1}^{N}, y_i \in \{+1, -1\}\}$, where N is the number of instances, $y_i = +1$ is the minority class, and $y_i = -1$ the majority class. In our method, we create some sets, in order to detect the regions of the input space are difficult to predict for the classifier.

Minority = $\{(x_i, y_i), x \in X, y_i = +1\}$, this set contains all the instances of the minority class in X. The following two subsets of X, contain only instances of the majority class:

$$TrM_j = \{(x_i, y_i), x_i \in X \text{and} x_i \notin TeM_j, y_i = -1\}$$
$$TeM_j = \{(x_i, y_i), x \in X \text{and} x_i \notin TrM_j, y_i = -1\}$$

such that $TrM_j \cup TeM_j = X -$ Minority, and $TrM_j \cap TeM_k$ is empty.

The elements of TrM_j and TeM_j are chosen randomly. The size of these sets is 60 % of $|X -$ Minority$|$ and 40 % of $|X - Minority|$, respectively.

Algorithm 1. Counter of errors in predictions

```
Input  : X: Training data set, C: Type of classifier, I: Number of iterations
Output: E: Mean of of missclassifications for each instance
begin
    for j ← 1 to I do
        Create the sets Tr_j and Te_j and Build a classifier C of type C from Tr_j;
        foreach instance x ∈ Te_j do
            Use C to predict the class of x;
            if C incorrectly classifies x then
                Update the counter of errors E for this instance;
            end
        end
    end
    return mean( E)
end
```

The sub-training set, Tr_j, is composed of all instances of the minority class and the elements of TrM_j: $Tr_j = TrM_k \cup$ Minority. Also, we create the sub-testing set, Te_k, composed of all instances of the minority class, and those instances of the majority class that are not in Tr_j: $Te_j = TeM_j \cup$ Minority. Having these sets created, a classifier is trained and tested several times. The errors in predictions are stored in a vector ε to be analyzed later. Algorithm 2 shows the pseudo code that implements this part of our method

Once ε obtained, those instances which have been classified incorrectly a number of times that exceeds a certain threshold, are categorized into two types:

(a) Noisy instances, difficult to predict instances and most of their k-nearest neighbors have opposite class.
(b) Secure instances, difficult to predict instances and most of their k-nearest neighbors have the same class.

During the experiments, we found that $k = 5$ produces good results for most data sets. Different from other approaches that only take into account a number of nearest neighbors, in our approach, the noisy instances play an important role in the generation

of new synthetic ones. The latter are generated in the lines that joins a noisy instance and its nearest L-neighbors.

2.1 Run-Time Complexity

In our method, the separation of majority and minority class instances is realized in linear time, $O(n)$. The creation of sub-training and sub-testing sets is also a linear time task. Training time varies form a type of classifier to other, we represent it with $\mathrm{T}(|Tr_j|)$. The prediction of the class for each an instance depends on the classification method, so we represent time with C, therefore, the time to predict all the instances in the sub-testing set is $|Te_j|\mathrm{C}$. Updating the vector ε is a constant time task, C_0. In current implementation of the algorithm, the generation of synthetic instances requires a linear search of the L-nearest neighbors for each noisy instance, the worst case is $O(n^2)$. Our method is slow for large data sets. The time-complexity of our method is therefore:

$$O(n) + IO(n) + I\mathrm{T}(Tr_j) + I|Te_j|\mathrm{C} + IC_0 + IO(n^2) \approx$$
$$I\mathrm{T}(0.6n) + I0.4n\mathrm{C} + IO(n^2)$$

3 Experiments and Results

In order to observe how the performance of classifiers is improved by pre-processing the data with our method, we select the C4.5 classifier, which is one of the most commonly algorithms chosen to test the performance of balancing methods. The data sets used to test the experiments are publicly available on the Internet,[1] their main features are shown in Table 1.

Table 1. Data sets for experiments

Data set	D	S	IR	Data set	D	S	IR
yeast-2_vs_4	8	514	9.08	glass-0-1-6_vs_2	99	192	10.29
glass2	9	214	11.59	ecoli4	77	336	15.80
page-blocks-1-3_vs_4	10	472	15.86	abalone9-18	88	731	16.4
glass-0-1-6_vs_5	9	184	19.44	glass5	99	214	22.78
car-good	6	1,728	24.04	yeast5	88	1,484	32.73
abalone19	8	4,174	129.44				

In Table 1, D is the number of attributes, S is the number of instances, and IR is the imbalance ratio. In these sets IR varies from 9, up to more than 120. We present the comparative of our method against SMOTE, re-sampling with and without

[1] http://sci2s.ugr.es/keel/datasets.php.

replacement. SMOTE algorithm generates synthetic instances using the 5 nearest neighbors, re-sampling makes copies of minority class instances.

All the experiments were conducted on a computer with the following characteristics: 2.6 GHz Intel Core i5 processor, 8 GB RAM, Mavericks Operating System. The size of RAM allocated to the JVM is 256 MB. In the experiments, each data set was partitioned into two subsets, randomly: training and testing. The former contains 60 % of instances of data set; the latter contains the rest. The training set is processed using our method, SMOTE and re-sampling with and without replacement. Then, a classifier C4.5 is trained with the processed data. The testing set is used to test performance of classifier. This process was repeated 30 times and the average is reported in the results.

4 Results

The application of Algorithm 2 provides with the information presented in Table 2, whose column have the following meaning. Data set: Name of data set analyzed; P: Number of minority class instances in the sub-training set; N: Number of majority class instances in the sub-training set; D_p: Number of minority class instances which are difficult to predict for the classifier; D_n: Number of majority class instances which are difficult to predict for the classifier; N_p: Number of noisy minority class instances; N_n: Number of noisy majority class instances. In order to achieve repeatable results for other researchers, the C4.5 (J48 Weka implementation) classifier was used with default parameter values. The threshold used in the experiments was set to one.

Table 2. Identification of difficult instances for the C4.5 classifier

Data set	P	N	D_p	D_n	N_p	N_n	Data set	P	N	D_p	D_n	N_p	N
yeast-2_vs_4	37	323	11	19	9	2	glass-0-1-6_vs_2	14	121	4	18	4	1
glass2	15	135	15	34	15	2	ecoli4	15	221	4	6	3	0
page-blocks-1-3_vs_4	23	308	0	2	0	0	abalone9-18	32	480	19	27	19	1
glass-0-1-6_vs_5	7	122	1	6	1	1	glass5	8	142	1	5	1	0
car-good	53	1,157	53	70	53	20	yeast5	37	1,002	2	22	2	8
abalone19	25	2,897	25	0	25	0							

Based on the average results shown in Table 2, the following can be observed:

(1) Most of the instances that are difficult to predict, belong to majority class. This is probably due to between-class imbalance, because of the large number of majority class instances.
(2) In general, the minority class instances that are difficult to predict, are also noisy instances. We attribute this to within-class imbalance.
(3) Most of majority class instances that are difficult to predict, are secure instances. This result is different from the informed in the literature, further investigation is necessary.
(4) All the minority class instances of data sets glass2, car-good and abalone19 are noisy instances, i.e., these data sets do not contain secure instances of the minority class.

(5) Data sets glass-0-1-6_vs_5, glass5 a yeast 5, contain just a few noisy instances of minority class. This makes difficult for our method to generate many instances.

In our method, we use the noisy instances to generated a number of synthetic instances, such that a balance of approximately 30 % is achieved. The underlying idea is to warn the classifier on regions not considered important, but they are.

Table 3 shows the area under the ROC for classifier C4.5. *None* corresponds to the performance of classifier without a pre-processing step of data. *Proposal* column is the method presented in this paper. *SMOTE* is the classic method with $K = 5$ nearest neighbors. R1 and R2 are re-sampling of minority class instances with and without replacement, respectively. In general, our method outperforms SMOTE, R1 and R2 in the cases where the number of difficult and noisy instances is not too small. In the other cases, our method produces results that are acceptable. Due to space issues, we don't present more results with other classification methods.

Table 3. AUC for classifier C4.5

Data set	None	Proposal	SMOTE	R1	R2
yeast-2_vs_4	0.895	**0.913**	0.880	0.857	0.895
glass-0-1-6_vs_2	0.675	**0.730**	0.712	0.634	0.675
glass2	0.744	**0.778**	0.689	0.657	0.744
ecoli4	0.821	**0.887**	0.875	0.869	0.821
page-blocks-3vs4	0.969	**0.987**	0.978	0.978	0.969
abalone9-18	0.619	0.685	**0.692**	0.622	0.619
glass-0-1-6_vs_5	0.875	0.965	**0.967**	0.903	0.875
glass5	**0.953**	0.862	0.905	0.867	**0.953**
car-good	0.444	0.911	**0.942**	0.904	0.444
yeast5	0.882	**0.921**	0.907	0.873	0.882

5 Conclusions

The performance of classifiers on imbalanced data sets is generally unacceptable. This problem is complex, since there are many factors involved, such as rare instances, between-class imbalanced within-class imbalance and noisy instances.

In this paper, we introduce a method to tackle with the classification task on imbalanced data sets. Different from other state-of-the-art proposals, our method is based on the philosophy that classification algorithms need to be involved in the generation of synthetic instances. We identify those instances that are difficult to predict correctly for a classifier. These instances are considered to detect regions in the input space that need to be reinforced with new synthetic instances. The method proposed in this paper was tested with 11 data sets and compared with other state-of-the-art methods. According to the results, our approach outperforms the current methods in most cases.

References

1. Esfandiari, N., Babavalian, M.R., Moghadam, A.-M.E., Tabar, V.K.: Review: knowledge discovery in medicine: current issue and future trend. Expert Syst. Appl. **41**(9), 4434–4463 (2014)
2. García, V., Sánchez, J.S., Mollineda, R.A.: On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. Knowl. Based Syst. **25**(1), 13–21 (2012). Special Issue on New Trends in Data Mining
3. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. **21**(9), 1263–1284 (2009)
4. Hilas, C.S., Mastorocostas, P.A.: An application of supervised and unsupervised learning approaches to telecommunications fraud detection. Knowl. Based Syst. **21**(7), 721–726 (2008)
5. Lemnaru, C., Potolea, R.: Imbalanced classification problems: systematic study, issues and best practices. In: Zhang, R., Zhang, J., Zhang, Z., Filipe, J., Cordeiro, J. (eds.) ICEIS 2011. LNBIP, vol. 102, pp. 35–50. Springer, Heidelberg (2012)
6. Sheng, V.S., Gu, B., Fang, W., Wu, J.: Cost-sensitive learning for defect escalation. Knowl. Based Syst. **66**, 146–155 (2014)
7. Sun, J., Li, H., Huang, Q.-H., He, K.-Y.: Predicting financial distress and corporate failure: a review from the state-of-the-art definitions, modeling, sampling, and featuring approaches. Knowl. Based Syst. **57**, 41–56 (2014)
8. Tomasev, N., Mladenic, D.: Class imbalance and the curse of minority hubs. Knowl. Based Syst. **53**, 157–172 (2013)