Balancing Distribution of Intrusion Detection Data Using Sample Selection

Ikram Chaïri, Souad Alaoui, Abdelouahid Lyhyaoui
LTI Lab, Department of Electrical and Industrial
ENSA of Tangier
Abdelmalek Essaâdi University
BP: 1818, Tanger Principal, Tanger
Morocco
{Chairikram, lyhyaoui}@gmail.com, souad_a2002@yahoo.fr

ABSTRACT: *The majority of learning systems usually assume that training sets are balanced, however, in real world data this hypothesis is not always true. The problem of between-class imbalance is a challenge that has attracted growing attention from both academia and industry, because of its critical influence on the performance of learning systems. Many solutions were proposed to resolve this problem: Generally, the common practice for dealing with imbalanced data sets is to rebalance them artificially by using sampling methods. Unfortunately, these methods can't give a high performance of learning. In this paper, we propose a new method based on Sample Selection (SS), to deal with the problem of between class imbalance. We consider that creating balance between classes by maintaining those examples located near the border line improves the performance of the classifier. To reduce the computational cost of selecting all samples, we propose a clustering method as a first step in order to determine the critical centers, and then select samples from those critical clusters. Experimental results with Multi-Layer Perceptron (MLP) architecture, on well known Intrusion Detection data set, show that our approach allows to attend the precision of Boosting methods, that we will explain how it can be considered like a SS method.*

Keywords: Imbalanced Data, Intrusion Detection System, Nearest Opposing Pairs, Sample Selection, Boosting, Classification

Received: 12 August 2012, Revised 29 September 2012, Accepted 5 October 2012

1. Introduction

The growth of raw data caused by the development of sciences and technologies has created an immense opportunity to improve data engineering. The problem of imbalanced data emerged as more and more researchers realized that their data sets were imbalanced and that this imbalance caused suboptimal classification performance.

In recent years, the imbalanced learning problem has generated a significant amount of interest from academia, industry, and government funding agencies. The main problematic was and still to find a classifier which can learn from an imbalanced data without ignoring the minority class. To deal with this class imbalance problem, many solutions were proposed, such as the case of sampling methods, cost function method, kernel based method and active learning method [1], [2], [3]. Sampling methods still the most widely used method to deal with the problem of imbalanced class [4]. Instead of the problem of imbalanced data, the approximation of the misclassification error used in the learning system can also contribute negatively in decreasing the accuracy and the quality of learning. That is why a different method of sample selection has been proposed to deal with this

problem [5], [6]. The main idea of SS methods is to focus training on those samples with most difficulty (Samples which are close to the border or having a big error) [5].

SS can be an alternative to random sampling methods as selection is focused on critical examples, and the balance between classes is achieved by keeping just the most important examples for majority class.

However, SS is still presenting the drawbacks of high computational cost involved in the process of selecting examples. To deal with this inconvenient we propose a clustering method and select critical centers instead of selecting samples. Another method to apply techniques of sample selection is to use Boosting methods for improving the accuracy of learning.

Another way to improve performance of learning is using Boosting techniques which use a consolidation of weak classifiers to find a strong one. We show in this paper how Boosting can be considered as a SS method. In this way, we compare the results of the proposed approach with those of Boosting.

We dedicate a particular interest to Intrusion Detection System (IDS), where in general, the number of fraudulent operations causes an imbalanced distribution.

We start this paper by presenting the IDS, in section 2, and we introduce in section 3 the problem of imbalanced data, with a quick overview on the solutions proposed in the literature. In section 4 we give a summary about the use of SS methods in classification problem, and we present our method, in section 5 an overview about Boosting techniques. We then describe the data set in section 6, and we reserve section 7 to present experimental results. Finally, in section 8 we conclude the paper and outline future research.

2. Overview about Instrusion Detection System

An Intrusion Detection System (IDS) is defined as a protection system that monitors computers or networks for unauthorized activities based on network traffic or system usage behaviors. In response to those identified adversarial transactions, IDS can inform relevant authorities to take corrective actions.

There is a large number of IDS available on the market to complement firewalls and other defense techniques. These systems are categorized into two types of IDS, namely (1) misuse-based detection in which events are compared against pre-defined patterns of known attacks and (2) anomaly-based detection which relies on detecting the activities deviating from system "*normal*" operations. Intrusion detection is a binary classification problem, a deal match normal connections or intrusion, hence the necessity of methods and classification algorithms.

Since the number of data that an IDS needs to examine is very large even for a small network, analysis is difficult even with computer assistance because extraneous features can make it harder to detect suspicious behavior patterns [7]. IDS must therefore reduce the amount of data to be processed. This is very important if real-time detection is desired. Reduction can occur by data filtering, data clustering and feature selection. In our purpose, we will be interested by the last technique, where in complex classification domains features may contain false correlations, which hinder the process of detecting intrusions. Furthermore, some features may be redundant since the information they add is contained in other features. Extra features can increase computation time, and can have an impact on the accuracy of IDS. Feature selection improves classification by searching for the subset of features, which best classifies the training data [8].

3. The Problem of Imbalanced Data

Different aspects can influence the performance achieved by existing machine learning. It has been reported that one of these aspects is related to class imbalance in which there are many more instances of some classes than others. The imbalanced learning represents a recurring problem of high importance because of its implication in various fields, it is prevalent in many applications, including fraud/intrusion detection, risk management, medical diagnosis/monitoring [2], [9].

Since 2000 , the imbalanced learning problem has drawn a significant interest that gave rise to two workshops held in 2000 and 2003 at the AAAI and ICML conferences, respectively [10]. Most standard algorithms assume balanced class distribution or equal misclassification cost, that's why learning algorithm often fails to generalize inductive rules over the sample space when

presented with this form of imbalance. In such cases, standard classifiers tend to be overwhelmed by the large classes and ignore the small ones [10], [11]. The justification should be clear: For instance, 90% of the data are from one class, for most realistic problems a learning algorithm will be hard pressed to do better than the 90% accuracy achievable by the trivial classifier that labels everything with the majority class Proposed solutions:

A number of solutions to the class-imbalance problem were previously proposed both at the data and algorithmic levels. At the algorithmic level, solutions include Cost sensitive learning which assigns a high cost to misclassification of the minority class, and try to minimize the overall cost [1], [2]. And Kernel-Based methods that use the theories of statistical learning and Vapnik-Chervonenkis (VC) dimensions in order to mmaximize the generalization [2].

Other solutions try to adjust the probabilistic estimate at the tree leaf (when working with decision trees), or adjust the decision threshold, and recognition-based (i.e., learning from one class) rather than discrimination-based (two class) learning [12].

Those methods generally provide a viable alternative to sampling methods which are the most useful solution of the imbalance problem. These techniques that work at the data level, include many different forms of re-sampling such [4], [2]:

• **Random Oversampling and Undersampling:** The principle of these methods is to add (oversampling) or remove (undersampling) entities selected in a random manner [13], [2]. Even if these methods are very simple, it introduces a set of problematic consequences. For undersampling methods, removing randomly examples from majority class can cause missing of important concepts. On the other hand, the oversampling can increase the risk of occurring over-fitting, because it makes exact copies of the minority class examples [2], [12].

• **Informed Undersampling:** The main idea of these methods is to sample multiple subsets of the majority class, train an ensemble from each of these subsets, and combine all weak classifiers in these ensembles into a final output. Two examples of informed undersampling that have shown good results are EasyEnsemble and BalanceCascade [2]. The difference between these two ways of sampling is that the EasyEnsemble samples independent subsets, while BalanceCascade uses trained classifiers to guide the sampling process for subsequent classifiers [13]. An apparent weakness of these methods is the lack of comprehensibility [13].

• **Synthetic Sampling with data generation:** Synthetic sampling or synthetic minority oversampling technique (SMOTE) consist to create artificial data based on the feature space similarities between existing minority examples. Because it generates the same number of synthetic data samples for each original minority example and does so without consideration to neighboring example, SMOTE causes in general problem of over generalization [2].

There are other types of re-sampling solutions which were proposed, and which provide an improvement of precision of learning, such as Adaptive Synthetic sampling and Cluster-Based Sampling Method [2].

These solutions cause in general problems of overlapping, redundancy or over-learning [2]. But they are still the most used ones because of their computational cost and performance.

4. Sample Selection and the Proposed Method

4.1 Related works

The classical learning algorithms of NN generally use all the training set to adjust the weights of the network without taking into account the contribution of every sample in the training, considering by this approach that all the samples have the same contribution in the definition of the border. However, in practical cases, in which sometimes the data base is oversized, the use of all training set is computationally expensive, or can prevent the training to converge to an acceptable minimum.

At the end of Eighties new works emerged in the literature, to deal with those problems, called Sample Selection techniques (SS) [14] [15]; such works took as point of practices the capacity of the NN to learn from examples, which was considered as a beginning of a line of works that gave fruitful results in several fields of application[17]. The main idea of these techniques is that the convergence of learning will be accelerated when a selected sample are presented in learning than a random one [16].

SS methods consist in dividing, implicitly, the training set in samples that contribute to the definition of the border (those that

will be used to update the weights of the classifier during training), named "*critical* " samples, and other redundant ones that do not give any information during the training.

The process of SS has different objectives, according to the type of the NN and problem. [5] says that SS, or sample editing techniques can resolve problems of approximation by estimating the adequate measure of classifiers quality: The misclassification rate, by paying more attention to samples that are more important and have a potential participation in the definition of the classification borders.

However, techniques of SS suffer from some limitations; one of the most outstanding aspects is the problem of selecting samples which are "*critical* ": This choice will depend, in general, on the classifier used, and an arbitrary preliminary classifier could introduce an unacceptable dependence on this preliminary selection [17].

Denker and Huyser show that for a well chosen MLP architecture, a training set samples formed only by examples near the border is sufficient to ensure a good generalization [14], [15]. In [18] a criterion of nearest neighbor is used to distinguish examples that generate confusion, intuitively, a sample that has a nearest neighbor belonging from other class is probably near to the border. In 1991 and 1994 Zhang proposed a new method of SS named Incremental Selection in which the sample of training is growing by the learning [19], [20], [21], [22].

Another method of selecting critical samples proposed by Cachin named "*Pedagogocal Pattern Selection Strategies*" that favorize the selection of samples having high error [23]. Munro [17] says that it is better to apply more frequently the samples that are more difficult to learn; generally, they are those samples near the boundary and showing a high error.

One alternative is to reduce the size of the selection problem by means of clustering [6]. The approximation proposed in [6] combines techniques of clustering and new criterion of SS giving rise to a new method of SS.

4.2 The proposed method

The proposed method aims at improving the performance of classifier by balancing the data base. We realize an Undersampling of data set by applying a SS method on a critical clusters, in order to train the classifier using the examples the most difficult to learn.

In [6] SS is used to construct a classifier, by selecting their parameters. In particular, they select centroids that will be used as centers of RBF function; meanwhile in our procedure, we select examples within critical centroids. By this way, we provide a balance between classes.

We can summarize the proposed method in two steps:
First step: Determining the critical centroids

Using a Vectorial Quantification (VQ) of data we determine centers of cluster for each class, followed by a Supervised Learning for Clustering by applying Kohonen's LVQ3. This method can reduce the computational cost of the procedure by working with the centroids instead of using the whole database. Once the centers of each class are determined, the process select critical centroids in two steps:

• **The closest opposite pairs:** This step is to determine the closest opposing pairs; by this way, one local border corresponds to the middle of each opposite pairs [24]. All the centroids are visited, and, for each one, the nearest centroid of the other class is determined. When two centroids of different classes are the nearest in both senses, both are included in the first group of critical centroids [24]. Considering C1 and C2 two sets formed by centers of positive and negative class respectively. To construct the set of nearest opposing pairs, we firstly determine P1, the set of nearest centers for C1, then we define P2 the set of nearest centers for C2. The final set of closest opposite pairs is the intersection between P1 and P2.

As the figure can show, defining the boundary according only to the closest opposite pairs can sometimes cause a misclassification of other centers. In this case we pass to the second step.

• **The rest of critical centers:** We constitute a new set of centers by including the center not correctly classified nearest to a critical center from the opposite class. We repeat this process until all centres are correctly classified.

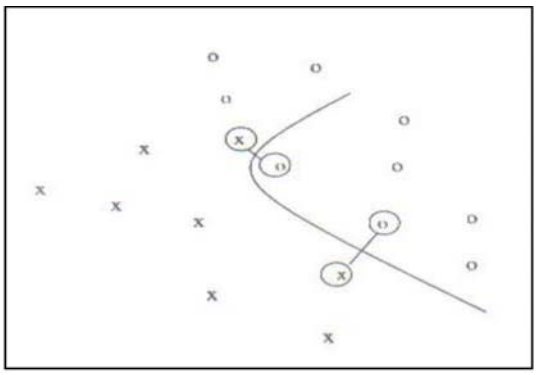

Figure 1. Selection of the closest opposite pairs

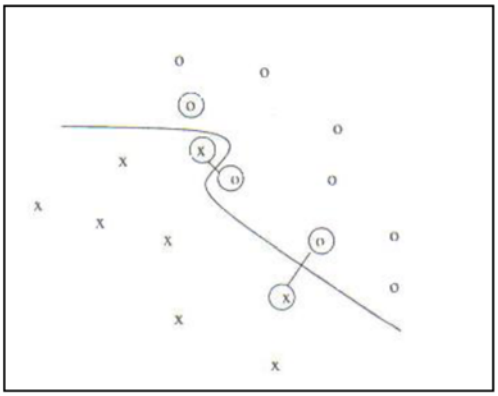

Figure 2. Correction of the classifier

As is shown in figure 2, by adding the rest of misclassified critical centres we correct the boundary and ensure that all the centres are correctly classified.

The second step: Selection of examples from the critical clusters.

This step consists on selecting samples that are difficult to learn from the critical cluster. We update the training sample by eliminating samples having small error from critical cluster of majority class. By this method we provide a focused under-sampling of data. Another method of selecting examples is to choose those that generate confusion.

In this paper, experiments were realized by whole examples from critical clusters in order to keep balance between classes.

Unfortunately, this approach has some apparent drawback like the use of some additional parameters, e.g., initial number of centroids (for each class), clustering parameters, number of selected samples per cluster, etc. All of these are known problems with a variety of solutions that we will try to explore in our experiments.

5. Boosting Techniques

5.1 Theorical aspect of Boosting

Boosting is a method of finding a highly accurate hypothesis (classification rule) by combining many "*weak* " hypotheses, each of which is only moderately accurate. Typically, each weak hypothesis is a simple rule which can be used to generate a predicted classification for any instance [25]. Kearns and Valiant were the first to pose the question of whether a "*weak* " learning algorithm which performs just slightly better than random guessing in the PAC model can be "*boosted* " into an arbitrarily accurate "*strong* " learning algorithm. Schapire came up with the first provable polynomial-time boosting algorithm in 1989. A year later, Freund developed a much more efficient boosting algorithm which, although optimal in a certain sense, nevertheless suffered from certain practical drawbacks [26].

The AdaBoost (Adaptive Boosting) algorithm, introduced in 1995 by Freund and Schapire [26], solved many of the practical difficulties of the earlier boosting algorithms. Pseudocode for AdaBoost is given below. The algorithm takes as input a training set (x_1, y_1), ..., (x_N, y_N) where each x_i belongs to some domain or instance space X and each label y_i is in some label set Y. AdaBoost calls a given weak or base learning algorithm repeatedly in a series of rounds t = 1…T. One of the main ideas of the algorithm is to maintain a distribution or set of weights over the training set. The weight of this distribution on training example *i* on round *t* is denoted $D_t(i)$. Initially, all weights are set equally, but on each round, the weights of incorrectly classified examples are increased so that the weak learner is forced to focus on the hard examples in the training set. The weak learner's job is to find a weak hypothesis $h_t: X \rightarrow Y$ appropriate for the distribution $D_t(i)$. The goodness of a weak hypothesis is measured by its error

$$e_t \sum_{i:h_t(x_i) \neq y_i} D_t(i)$$

Notice that the error is measured with respect to the distribution on which the weak learner was trained. In practice, the weak learner may be an algorithm that can use the weights Dt on the training examples.

Algorithm of AdaBoost :

Input $S_1 = \{(x_1, y_1),....,(x_N, y_N)\}$.

Initialize : $D_1(i) = \frac{1}{N}$

For t=1,..., T**:**

- Train weak learner using distribution D_t
- Get weak hypothesis $h_t: X \rightarrow Y$ with error

$$e_t \sum_{i:h_t(x_i) \neq y_i} D_t(i)$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-e_t}{e_t}\right)$
- Update: $D_{t+1}(i) = \frac{D_t(i)}{z_t} \exp(-\alpha_t y_i h_t(x_i))$

Where z_t is a normalization factor (chosen so D_{t+1} will be a distribution).

Output: $H(x) = sin(\sum_{t=1}^{T} \alpha_t h_t(x))$

The AdaBoost algorithm is very successful thanks to its simplicity and ease of programming. However, a disadvantage of the AdaBoost algorithm is the dependence of its performance to data and weak learner. The performance of the algorithm is not good if we do not have enough data or that have only weak hypothesis.

5.2 Boosting as Sample Selection technique

AdaBoost updates the weights of the training data to focus on examples difficult to classify. Correctly classified examples are assigned a small weight and those incorrectly classified a greater one.

In [27] a method of emphasis that pays attention to the samples that produce a big error and they are near the border has been proposed, in order to improve the performance of AdaBoost. A new function of weight of distribution has been defined:

$$D_{\lambda,t+1}(i) = \frac{1}{z_t} \exp[\lambda (h_t(x_i) - y_i)^2 + (1-\lambda) h_t(x_i)^2]$$

Where λ $(0 \leq \lambda \leq 1)$ is a weighting parameter. This formulation allows us to choose how much to consider the "*proximity*" to the boundary or the quadratic error of each sample, by selecting different values of λ. We highlight three particular cases associated with the values of λ:

• Classifiers focus on the critical samples ($\lambda = 0$)

• In both terms of emphasis (classical AdaBoost function) ($\lambda = 0.5$)

• Sampling examples that generate more square error ($\lambda = 1$).

In 2009, Garìa-Pedrajas developed a Boosting method based Sample Selection technique by training firstly all the data base using standard Adaboost method and selecting then examples with difficulties in learning and that minimize the weighted error [28]. This proposed classifier assume that each set trained with samples that are relevant to the learning process, improve the performance, and also reduce the training set (reduce complexity of space characteristics) during training [28].

6. Intrusion Data Set

The first requirement of each IDS is a set of input data to be processed in order to determine the security level. In this experiment, we train and test our system using a standard dataset KDD Cup's 99 dataset, the raw data used by the KDD Cup 1999 intrusion detection contest.

6.1 Kddcup Dataset

In 1998 DARPA (KDD-cup dataset) [8] intrusion detection evaluation programme on an environment was set up to get raw TCP/IP dump data for a network by simulating a typical US Air Force LAN. The LAN was operated like a real environment, but was blasted with several attacks.

A standard set of data to be audited, including a wide variety of intrusions simulated in a military network environment, was provided. The 1999 KDD intrusion detection contest used a version of this dataset [8].

For each TCP/IP connection, 41 various quantitative and qualitative features were extracted. Each connection record was labelled as either normal, or as an attack, with exactly one specific attack type, which can be arranged into 4 categories namely - Probe, DOS, U2R and R2L, as can be shown in Table 1.

The four different categories of attack patterns are:

1. Denial of Service (DOS) Attacks: which prevent a computer from complying with legitimate requests by consuming its resources.

2. User to Root Attacks (U2R): which have the goal of obtaining illegal or non authorized super-user or root privileges.

3. Remote to Local Attacks (R2L): which are local non authorized access attempts from a remote machine.

4. Probing (Probe): which are scanning and polling activities that information on vulnerabilities for future attacks

Class name	Training data set		Test data set	
	Number of instance	%	*Number of instance*	%
Normal	79277	19.6 9	60593	19.48
DOS	391458	79.24	229851	74.90
U2R	59	0.01	228	0.07
R2L	1126	0.23	16189	5.21
Prob	4107	0.83	4168	1.34

Table 1. Distribution of Training Data and Test Data KDD-CUP-99

For this research, the labeled 10% training and test datasets were used [29]. The training set is composed by 22 attack types and the test set consists of 37 attack types, which can be arranged into 4 categories (Probe, DOS, U2R and R2L). The percentage

distribution of both the training and test datasets with respect to the 5 categories (Normal, Probe, DOS, U2R and R2L), is also shown in Table 1.

The training subset of this database, is composed of 494014 records, from which about 20% represent normal patterns. The rest of 80% of patterns are attacks belonging to the four different categories cited above. Indeed, the test set was composed of 311029 data records.

Each instance in the KDD Cup 1999 datasets contains 41 features that describe a connection: (*duration, su_attemted, same_srv_rate, protocol_type, num_root, diff_srv_rate, service, nu_file_creations, srv_diff_host_rate, flag, num_shells, dst_host_count, src_bytes, num_access_file, dst_host_srv_count, dst_bytes, num_outbond_cmds, dst_host_same_srv_rate, land, is_host_login, dst_host_diff_srv_rate, wrong_fragment, is_guest_login, dst_host_same_src_port_rate, urgent, count, dst_host_srv_diff_host_rate, hot, srv_count, dst_host_serror_rate, num_failed_logins, serror_rate, dst_host_srv_serror_rate, logged_in, srv_serror_rate, dst_host_rerror_rate, num_compromised, rerror_rate, dst_host_srv_rerror_rate, root_shell,srv_rerror_rate*)

6.2 Data pre-processing

Features in the KDD datasets have all forms - continuous, discrete, and symbolic, with significantly varying resolution and ranges. Most pattern classification methods are not able to process data in such a format.

We propose the following four steps to overcome this drawback:

• Attributes numeric-valued to each symbolic- valued.

• Examples were first mapped to one of the tow classes, 0 for Normal, 1 for attack (Probe, DoS, U2R, or R2L).

• Normalization was applied to attributes to reduce the dynamic range to [0…1].

• Filtering out the duplicate records.

7. Experiments and Results

Given the large data set (about 5 million of examples), the experiments were performed on a small database sampled to 10% of data, so that the percentage of the minority class was maintained. All proposed dataset were used in order to build an intrusion detection model. Then, 10-fold cross validation method was used in order to test the effectiveness of the model built during the training phase based on the proportional normal data in each group of dataset.

We used an MLP architecture as classifier during learning process. It consists of three layers: the input, one hidden, and the output layer, the sigmoid function was used for each neuron; the back-propagation learning algorithm was used with a mean squared error as a cost function.

The input layer is of dimension 51 and we used one hidden layer; in addition the weights were randomly initialized with small values.

7.1 Learning using MLP architecture

Before applying the proposed method, we first train the whole data base using MLP architecture, and explore two parameters: The number of neurons and the learning rate (*lr*).

We present in table 2, the variation of precision of learning with respect to those parameters.

The best result is achieved by using 6 neurons and a value of learning rate equal to 10^{-4}. In the rest of experiments we use these two parameters.

We realized an exploration of the number of centroids in both minority and majority class. For majority class, the number of centroïdes varies between 10 and 30, mainwhile the number of minority class are determined by varying a new defined parameter named "*Weighing parameter* " (WP), which is the weight affected to the ratio of imbalance between the majority and the minority class, and is defined as follows:

Number of cluster in minority class = (*WP* × 0.2) × *Number of cluster in minority class*

		Number of neurons			
lr	2	4	6	8	10
10^{-2}	20	80	20	79.7	20
10^{-3}	20	80	75.4	20	20
10^{-4}	20	93.6	93.7	83.1	89.7
10^{-5}	94	93.5	93.6	93.3	93
10^{-6}	89.8	82.5	84	93.5	93
10^{-7}	81.3	62.4	75.9	73.7	84.5

Table 2. Variation of precision of learning in function of Number of neurons and learning rate

Detection and identification of attack and non attack behaviors can be generalized as follows:

• True positive (TP): the amount of the attack detected when it is actually an attack.

• True negative (TN): the amount of the normal detected when it is actually normal.

• False positive (FP): the amount of the attack detected when it is actually normal, namely false alarm.

• False negative (FN): the amount of normal detected when it is actually attack, namely the attacks which can be detected by intrusion detection system.

The accuracy refers to the proportion of data classified an accurate type in total data, namely the situation TP and TN, the accuracy can be defined as follows:

$$Accuracy = \left(\frac{TP + TN}{TP + TN + FP + FN}\right) \times 100\%$$

During the exploration of this proposed method, we faced some drawbacks that caused disruption of the learning, like non-existence of the closest opposed pairs. To deal with this problem, we proposed an alternative which chooses from the majority class the centers closest to the minority class. By this way we ensure that the process select samples that are near the borderline in case of non-existence of the closest opposed pairs.

The ratio of imbalance of the explored data is about 0.2.

We present our result in the Figure 3, which sum up the variation of accuracy with respect to the number of clusters in minority and majority class.

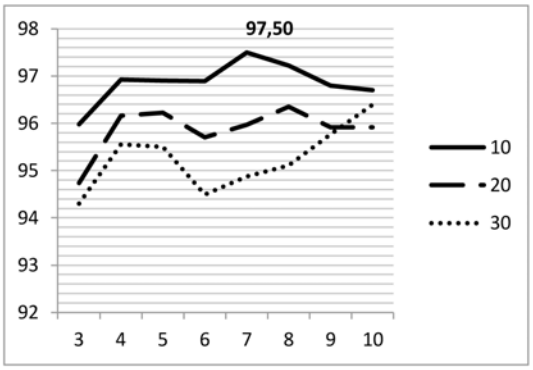

Figure 3. Variation of accuracy in function of numbre of centers of majority and minority class

According to those results we can see that the performance of learning is affected by the number of clusters of minority and majority class. We can say that the impact of number of centroïdes of majoratity class is much more important than minority class. Best values of accuracy are founded using 10 clusters in majority class.

The accuracy has increased to attend 97.5% for 10 clusters for majority class and 7 cluster of minority class.

By realizing a random Undersampling to balance the data base, we found that the accuracy of learning is about 95.5%

Finally, the application of Adaboost method on data base relives a precision of 97.7%.

We summarize in table IV, all results founded by realized experiments.

Data base	**Precision of learning**
Imbalanced data using MLP	93.7%
Imbalanced data using SVM	93.8%
Balanced data with Undersampling	95.5%
Balanced data with SS	97.5%
Learning using Adaboost	97.78%

Table 3. Comparaision etween Founded Results

Comparing those results, we can clearly see that techniques of sample selection and Adaboosting allow us to have the best results. We can say that the proposed method have the same precision as Adaboost techniques. The benefit of the proposed method is its simplicity and low computational cost, contrary to Adaboost which had taken a large time of running and a high computational cost.

8. Conclusion

In this paper, a novel method to deal with the problem of imbalanced data was proposed. A SS approach is used to create a balance between the classes. The main advantage of our method is its capacity to focus learning on the most important samples, which contribute positively in the improvement of the performance using clustering techniques. In the realized experiments we have shown how the application of SS on the majority class can lead performance of learning to reach level of performance of Adaboost method and that, by avoiding the selection of non-critical samples. Also, we have show the impact of the choice of the number of clusters on the performance of learning.

We consider this study as a beginning of a line of research in which we will explore more parameters that can improve the performance and introduce some theoretical bases on this work.

References

[1] Domingos, P. Metacost: A general method for making classifiers cost-sensitive, *In*: Proceedings of the 5th ACM SIGKDD *International Conference on Knowledge Discovery and Data Mining*, p. 155– 164., San Diego. ACM press.

[2] Haibo, H., Garcia, A. E. (2009). Learning from Imbalanced Data, *IEEE Transactions on Knowledge And Data Engineering*, 2 (9) September.

[3] Kubat, M., Matwin, S. (1997). Addressing the curse of imbalanced data sets: One-sided sampling, *In*: The 14th *International Conference on Machine Learning*, p. 179–186.

[4] Chen, C., Liaw, A., Breiman, L. Using Random Forest to Learn Imbalanced Data.

[5] El Jelali, S., Lyhyaoui, A., Anibal, R., Figueiras, V. An Emphasized Target Smoothing Procedure to Improve MLP Classifiers Performance, *In*: ESANN'2008 Proceeding, European symposium on Artificial Neural Networks.

[6] Lyhyaoui, M., Martinez, I., Mora, M., Vázquez, J. L., Sancho, Aníbal, R., Figueiras-Vidal. (1999). Sample Selection Via Clustering to Construct Support Vector-like Classifiers, *IEEE Transactions on Neural Networks* (6) Nov.

[7] Sung, A. H., Mukkamala, S. (2003). Identifying Important Features for Intrusion Detection Using Support Machines and Neural Networks, *In*: *International Symposium on Applications and the Internet Technology Proceedings*, IEEE Computer Society Press, p. 209-216.

[8] Ruck, W. D., Rogers, K. S., Kabristy, M. (1990). Feature selection using multilayer perception, *Journal of Neural Network Computing*, 2, p. 40-48.

[9] Wu, G., Chang, E. Y. (2003). Class-boundary alignment for imbalanced dataset learning, ICML'03 Workshop on Learning from Imbalanced Data Sets.

[10] Nitesh, V., Chawla, N. V., Kolcz, A. Editorial: Special Issue on Learning from Imbalanced Data Sets. *Sigkdd Explorations*, 6 (1).

[11] Chawla, N. V. (2003). C4.5 and Imbalanced Data Sets: Investigating the Effect of Sampling Method, Probabilistic Estimate, and Decision Tree Structure, *In*: Workshop on Learning from Imbalanced Data Sets II.

[12] Batista, G. E. A. P. A., Prati, R. C., Monard, M. C. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data, *Sigkdd Explorations,* 6 (1).

[13] Liu, X.-Y., Wu, J., Zhou, Z.-H. Exploratory Undersampling for Class-Imbalance Learning. *IEEE Transactions on Systems*, MAN AND CYBERNETICS – PART B.

[14] Denker, J., Schwartz, D., Wittner, B., Solla, S., Howard, R., Jackel, L. (1987). Large Automatic Learning, Rule Extraction, and Generalization, *In*: Complex Systems, 1, p.877-922. Complex Systems Publications, Inc.

[15] Huyer, K., Hororwitz, A. M. (1988). Generalization in Connectionist Networks that realize boolean Functions, in Touretzky, D., Hinton, G., Sejnowski, T. (eds), *In*: Proc. 1988 Connectionnist Models Summer School, Morgan Kaufman, Palo Alto, CA, p. 191-200.

[16] Ohnishi, N., Okamoto, A., Sugie, N. Selective Presentation of Learning Samples for Efficient Learning in Multi-Layer Perceptron, IJCNN'91.

[17] Munro, P. W. Repeat until bored: A pattern selection strategy, *In*: Advances in Neural Information Proc. Sys. 4, J. E. Moody et al., Eds. San Mateo, CA: Morgan

[18] Wann. T. M., Hediger, N., Greenbaun, N. (1990). The influence of Training Sets on Generalization in Feed-Forward Neural Netwoks, *In*: Proceeding of th International Joint Conference on Neural Networks. 3, 137-142.

[19] Zhang, B. T., Mühlenbein, H. (1993). Genetic Programming of Minimal Neural Nets Using Occam's Razor, *In*: Proc. Int. Conf. Genetic Algorithms, p. 342-349.

[20] Zhang, B. T. (1993). Self-development Learning: Constructing Optimal Size Neural Networks via Incremental Data Selection, Tech. Rep. (768), German National Research Center for Computer Science.

[21] B. T. Zhang, Accelerated Learning by Active Example Selection, *International Journal of Neural Networks*, 5 (1) 67 -75.

[22] Zhang, B. T. (1994). An Incremental Learning That Optimizes Network Size and Sample Size in One Trial, *In*: Proc. of th IEEE International Conference on Neural Networks, p. 215-220.

[23] Cachin, C. (1994). Pedagogical pattern selection strategies, *Neural Networks*, 7, 171–181.

[24] Lyhyaoui, A. (1999). RBF classifiers via clustering techniques and sample selection, July.

[25] Schapire, E., Singer, Y. (1999). Improved Boosting Algorithms Using Confidence-rated Predictions, *Machine Learning*, 37 (3) 297-336.

[26] Freund, Y., Schapire, E. (1999). A Short Introduction to Boosting, *Journal of Japanese Society for Artificial Intelligence*, 14 (5) 771-780, September.

[27] Gòmez-Verdejo, V., Ortega-Moral, M., Arenas-Garcìa, J., Figueiras-Vidal, A. R. (2006). Boosting by Weighting Critical and Erroneous Samples, *Neurocomputing*, 69, 679 685.

[28] Garcìa-Pedrajas, N. (2009). Constructing Ensembles of Classifiers by Means of Weighted Instance Selection, *IEEE Transactions on Neural Networks,* 20 (2) 258-277.

[29] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html