Resampling Methods versus Cost Functions for Training an MLP in the Class Imbalance Context

R. Alejo[1], P. Toribio[1], J.M. Sotoca[2], R.M. Valdovinos[3], and E. Gasca[4]

[1] Tecnológico de Estudios Superiores de Jocotitlán
Carretera Toluca-Atlacomulco KM. 44.8, col. Ejido de San Juan y San Agustín, Jocotitlán
[2] Institute of New Imaging Technologies, Universitat Jaume I
Av. Sos Baynat s/n, 12071 Castelló de la Plana, Spain
[3] Centro Universitario UAEM Valle de Chalco, Universidad Autónoma del Estado de México
Hermenegildo Galena No.3, Col. Ma. Isabel, 56615 Valle de Chalco, Mexico
[4] Lab. Reconocimiento de Patrones, Instituto Tecnológico de Toluca
Av. Tecnológico s/n, 52140 Metepec, México

Abstract. The class imbalance problem has been studied from different approaches, some of the most popular are based on resizing the data set or internally basing the discrimination-based process. Both methods try to compensate the class imbalance distribution, however, it is necessary to consider the effect that each method produces in the training process of the Multilayer Perceptron (MLP). The experimental results shows the negative and positive effects that each of these approaches has on the MLP behavior.

Keywords: MLP, random sampling, cost function, class imbalance problem.

1 Introduction

Recently, the class imbalance problem has been recognized as a crucial problem in data mining and machine learning [1]. This inconvenience occurs in real-world domains, where the decision system is aimed to detect a rare but important case. For instance, in detection of oil spills, in satellite radar images, fraudulent telephone calls or fraudulent credit cards [2].

Some of the most popular strategies for handling this problem are resampling techniques (over-sampling and under-sampling) [3, 2]. Over-sampling replicates samples in the minority classes and the under-sampling eliminates samples in the majority classes [4]. These two basic methods for resizing the training data set (TDS) produce a class distribution more balanced. However, both strategies have shown important drawbacks: Under-sampling may eliminate potentially useful data, while over-sampling increases the TDS size and hence the training time [5]. In the last years, research has been focused on improving these basic methods (for example see [6, 7]).

In the Multilayer perceptron (MLP) which applies these methods (over or under-sampling) has demonstrated notable improvements in the classifier performance [4, 1] and it has been verified that random under-sampling techniques provide better results than those obtained by means of random over-sampling [8]. However, in [1] is affirmed

D. Liu et al. (Eds.): ISNN 2011, Part II, LNCS 6676, pp. 19–26, 2011.

that the under-sampling methods produce negative effects when the TDS is seriously imbalanced.

Other popular strategy to deal with the class imbalance problem consists in internally biasing the discrimination-based process and compensate the class imbalance [9, 10]. With the MLP, this approach consists mainly in including a cost function in the training phase or in the test phase. Empirical studies have shown that the usage of the cost functions improves the classifier performance. Nonetheless, the effects of the cost function in the training process is similar than over-sampling techniques, i.e., it causes changes in probability distribution data [5].

This paper studies empirically the effects of the random sampling and the cost function in the training phase of the MLP. In this way, we could identify methods that are effective in the moderate and severe imbalance problems.

2 The MLP and the Class Imbalance Problem

The MLP neural network usually comprises one input layer, one or more hidden layers, and one output layer. Input nodes correspond to features, the hidden layers are used for computing, and output layers are related with the number of classes. A neuron is the elemental unit of each layer. It computes the weighted sum of its inputs, adds a bias term and drives the result through a generally nonlinear (commonly a sigmoid) activation function to produce a single output.

The most popular training algorithm for MLP is the back-propagation strategy, which uses a training set for the learning process. Given a feedforward network, the weights are initialized with small random numbers. Each training instance is sent through the network and the output from each unit is computed. The output target is compared with the estimated one by the network, computing the error which is fed-back through the network.

To adjust the weights, the back-propagation algorithm uses a descendant gradient to minimize the squared error. Each unit in the network starts from the output unit and it is moved to the hidden units. The error value is used to adjust the weights of its connections as well as to reduce the error. This process is repeated for a fixed number of times, or until the error is minimum or it cannot be reduced.

Empirical studies of the back-propagation algorithm [11] show that class imbalance problem generates unequal contributions to the mean square error (MSE) in the training phase. Clearly the major contribution to the MSE is produced by the majority class.

We can consider a TDS with two classes ($m = 2$) such that $N = \sum_i^m n_i$ and n_i is the number of samples from class i. Suppose that the MSE by class can be expressed as $E_i(U) = \frac{1}{N} \sum_{n=1}^{n_i} \sum_{p=1}^{L} (d_p^n - y_p^n)^2$, where d_p^n is the desired output and y_p^n is the actual output of the network for sample n. Then the overall MSE can be expressed as $E(U) = \sum_{i=1}^{m} E_i = E_1(U) + E_2(U)$.

If $n_1 << n_2$ then $E_1(U) << E_2(U)$ and $\|\nabla E_1(U)\| << \|\nabla E_2(U)\|$, consequently $\nabla E(U) \approx \nabla E_2(U)$. So, $-\nabla E(U)$ it is not always the best direction to minimize the MSE in both classes.

Regarding that the imbalance problem affects negatively in the back-propagation algorithm due to the disproportionate contributions in the MSE, it is possible to consider

two options: A) Resizing the training data set in order to cause that the class distribution of the TDS become more balanced (for instance, replicating samples in the minority classes or eliminating samples in the majority classes [4]). B) Including a cost function in the back-propagation algorithm for avoiding that the minority classes are ignored in the learning process, and it could accelerate the convergence of the neural network.

Consider a cost function (γ) that balance the TDS class imbalance as follows: $E(U) = \sum_{i=1}^{m} \gamma(i)\ E_i = \gamma(1)E_1(U) + \gamma(2)E_2(U) = \frac{1}{N}\sum_{i=1}^{m}\gamma(i)\sum_{n=1}^{n_i}\sum_{p=1}^{L}(d_p^n - y_p^n)^2$, where $\gamma(1)\|\nabla E_1(U)\| \approx \gamma(2)\|\nabla E_2(U)\|$ avoiding that the minority class be ignored in the learning process. In this work, the cost function is defined as $\gamma(i) = \|\nabla E_{max}(U)\|/\|\nabla E_i(U)\|$, where $\|\nabla E_{max}(U)\|$ corresponds to the largest majority class.

3 Methodology

The experiments were carried out on eleven severely imbalanced datasets. These datasets were obtained from the transformation of Cayo into two-class problems (reducing a m-class problem to a set of m two-class sub-problems). The main characteristics of these subsets have been summarized in the Table 1. To increase statistical significance of the results the k-fold cross validation technique (with k=10) has been applied. About 90% out of the total number of samples available has been used for the TDS and the rest for a test set.

The MLP used in this study was trained by the back-propagation algorithm in batch mode. The learning rate (η) was set 0.1. One hidden layer was used with four neurons. The stop criterion was established at 25000 iterations or an MSE below 0.001. The training MLP has been repeated ten times. The results here included correspond to the average of those achieved in the ten repetitions and of ten partitions.

A general criterion to measure the classifier performance is the overall accuracy (Acc). $Acc = 1 - n_e/n$ where n_e is the number of misclassified examples and n is the

Table 1. Main characteristics of the eleven subsets obtained from Cayo; notice that the distribution of data is presented in different forms to simplify their interpretation

Data	Samples	Features	Distribution	Ratio by class	Ratio
C01	6019	4	838/5181	0.14/0.86	0.16
C02	6019	4	293/5726	0.05/0.95	0.05
C03	6019	4	624/5395	0.10/0.90	0.12
C04	6019	4	322/5697	0.05/0.95	0.06
C05	6019	4	133/5886	0.02/0.98	0.02
C06	6019	4	369/5650	0.06/0.94	0.07
C07	6019	4	324/5695	0.05/0.95	0.06
C08	6019	4	722/5297	0.12/0.88	0.14
C09	6019	4	789/5230	0.13/0.87	0.15
C10	6019	4	833/5186	0.14/0.86	0.16
C11	6019	4	722/5247	0.13/0.87	0.15

total number of testing examples. Nevertheless, in the class imbalance problems this is not the most suitable measure [6]. The geometric mean (*g-mean*) is one of the most widely accepted criterion, and is defined as $g = \sqrt{a^+ \cdot a^-}$, where $a^+ = 1 - n_e^{cls^+}/n^{cls^+}$ is the accuracy on the minority class (cls^+) and $a^- = 1 - n_e^{cls^-}/n^{cls^-}$ is the accuracy on the majority class (cls^-). In this work, *g-mean*, Acc, a^- and a^+ were applied to measure classifier performance.

4 Experimental Results

The random over-sampling strategies in a severe-imbalance context cannot be considered suitable alternatives given the considerable increase in the computing cost, they would generate in the neural network a slow training process. Therefore, this paper is focused on analyze the random under-sampling and cost function strategies, (see section 2) and its convenience to be used on a context of a severe class imbalance problem.

Japkowicz [4, 1] observe that the random under-sampling method can improve considerably the classifier performance by compensating the class imbalance and by reducing the computational cost associated to the model. But, which is its performance when the TDS has severe multi-class imbalance problem?.

On the other hand, empirical studies have shown that using a cost function can improve the classifier performance [9]. Functionally, using a cost function is equivalent to apply random over-sampling, but it does not increase (significantly) the computational cost in the training process. However, as in the case of the random under-sampling technique is necessary to ask, what are the effects of the cost function when the TDS is severely imbalanced?.

In Table 2 the obtained results with the cost function and random under-sampling are shown. It is possible to observe that the performance of under-sampling is better than the obtained with the original dataset, but worse than the produced by the cost function. Moreover, the under-sampling performance is comparable to the cost function (except in C04, C05 and C06).

For more detail, in the Fig. 1 the accuracy by class is shown (a^+ and a^-). The Fill boxes symbolize the cost function and the not fill ones the random under-sampling strategy. Observe that in the corresponding image, the cls^+ (Fig. 1a), in most of the datasets, a^+, is almost the same. In the sets related to C05 and C06, random under-sampling presents a a^+ inferior than the obtained with the cost function. Nonetheless, these results are higher than those shown by the MLP with the original dataset (a^+=47.55% for C05 and a^+=0.0% for C06). In other words, the random under-sampling increases the a^+ in relation to the standard back-propagation algorithm, but it does not shows better results than those from the cost function.

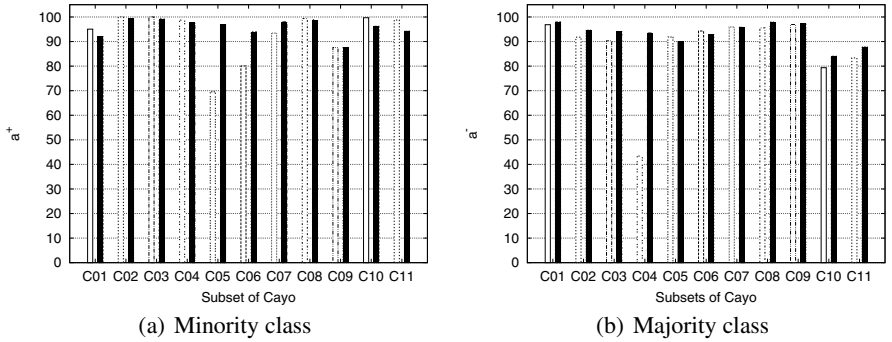
See Fig.1b (majority class) that in most of the subsets a^- both with cost function and random under-sampling is similar. Also, a tendency towards presenting better results is observed with cost function. In the particular case of the C04, the a^- obtained with under-sampling (a^-=43.87%) is very low regarding with original dataset (a^-=100%) and with the cost function (a^-=93.46%).

In summary, the results reported in the Table 2 suggest that on the severe class imbalance, random under-sampling can be a good choice. However, in some results were

Table 2. MLP: Average accuracy rate

Acc	Original	Cost function	Under-sampling
C01	97.81(0.03)	97.13(0.63)	96.61(0.74)
C02	97.42(0.36)	94.90(0.02)	92.18(0.15)
C03	96.40(0.04)	94.54(0.20)	91.30(0.42)
C04	94.65(0.00)	93.69(2.93)	46.36(0.92)
C05	97.81(0.00)	90.12(8.30)	91.34(6.17)
C06	96.79(0.04)	92.87(0.10)	93.41(0.36)
C07	94.58(0.00)	95.90(0.24)	95.80(3.43)
C08	98.54(1.01)	98.04(1.17)	96.03(2.75)
C09	98.12(0.06)	96.06(0.34)	95.71(0.71)
C10	92.33(0.72)	85.69(0.39)	82.17(0.06)
C11	89.32(0.97)	88.68(1.12)	85.31(0.46)
g-mean	Original	Cost function	Under-sampling
C01	93.60(0.03)	94.95(1.30)	95.95(1.47)
C02	68.25(6.34)	97.01(0.01)	95.85(0.08)
C03	94.32(0.18)	96.62(0.09)	95.07(0.25)
C04	0.00(0.00)	95.58(1.39)	65.50(0.72)
C05	0.00(0.00)	93.31(4.40)	79.95(4.80)
C06	68.99(0.47)	93.43(0.22)	86.91(0.93)
C07	0.00(0.00)	96.93(0.19)	94.71(2.37)
C08	96.67(3.66)	98.44(1.03)	97.49(1.66)
C09	93.37(0.03)	92.37(0.18)	92.17(0.39)
C10	75.14(3.99)	89.94(0.25)	88.99(0.05)
C11	53.34(13.22)	91.04(1.01)	90.76(0.21)

observed that the sampling methods produce a negative effect when the TDS is seriously imbalanced (confirm previous hypotheses [1]). So, the question here is, why?, or where reflected this effect?. To answer these questions, the outputs MLP of the C04 and C03 subsets are analyzed.

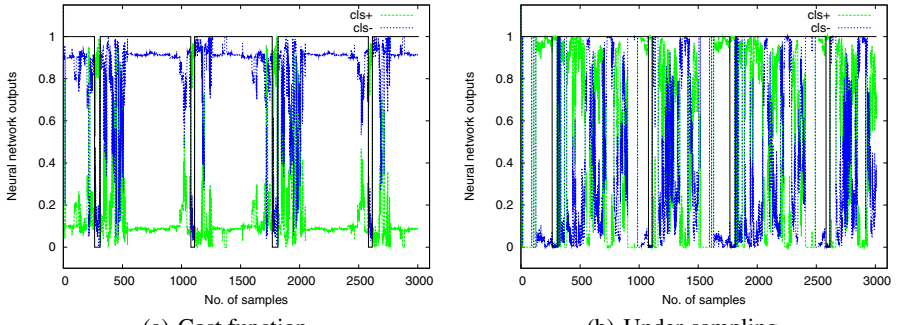


(a) Minority class (b) Majority class

Fig. 1. Accuracy of the minority class (a) and the majority class (b). Numbers in x axis represent the dataset used. The Fill boxes symbolize the cost function and the not fill boxes the random under-sampling strategy.

The Fig. 2 contains the graphic representation of MLP outputs for each sample utilized by evaluate the network performance with C04. Let us notice that x axis indicates the samples contained in the evaluation set, while y axis reflects the value of the network output (values between 0 and 1). The continuous line (black) establishes the separation between cls^+ and cls^- samples. Observe that in the largest intervals limited by this line, contain only samples of the most represented class and, in the smallest the elements of the cls^+. Green is associated to the neuron outputs corresponding to cls^+ and blue to the neuron associated to cls^-. For instance, in ideal conditions, i.e., when the network classify correctly, the behavior should be: when a sample is cls^+ the output value of the neuron associated to the cls^+ must be high (high values in green) and low for the neuron of cls^- (low values in blue) and viceversa when it is the case of cls^-.

Observe in Fig. 2a (cost function), the outputs for samples of the cls^- (blue) are high (values close to 1) when it corresponds of its class and low in other way. This behavior is constant in the Fig. 2a, which the cost function has a good performance in both cases, cls^+ and cls^-. An utterly different situation is shown in Fig. 2b (under-sampling) where the previous behavior is not observed. The Fig. 2b show a good performance on the cls^+, and very irregular on the cls^-.



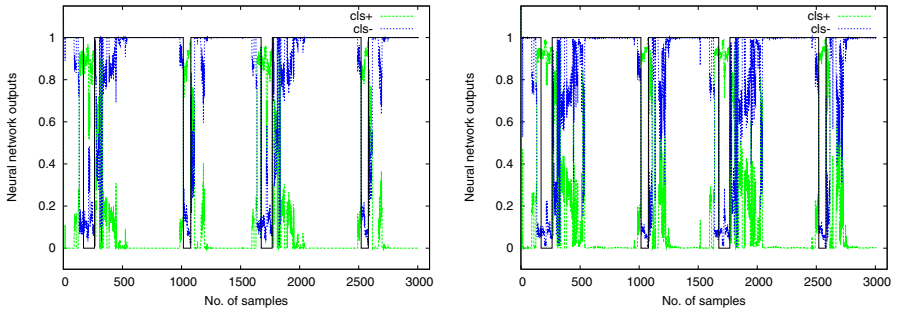
(a) Cost function (b) Under-sampling

Fig. 2. MLP outputs for the C04 subset, after to apply cost function and random under-sampling strategies. The line in black shows the separation between the outputs of both classes.

The results obtained with this subset suggest a weak learning on the cls^- (when random under-sampling is applied). Nevertheless, in the rest of the subsets it seemed as though this massive elimination of samples does not affect the cls^- (as it was observed in Fig. 1), but, what happen with this?.

The answer is in the sense that the class most represented (cls^-) is less learned when training samples are massively eliminated, i.e., the outputs present more irregular tendency than the cost function. To exemplify this, the subset C03 was utilized, i.e., this subset as the rest of the subsets does not present a significant difference related to random under-sampling and the cost function in their accuracy values, both for the cls^- and $cls+$ (Fig. 1).

In Fig. 3, the MLP outputs for the C03 subset are presented after to applying a cost function and random under-sampling technique. In Fig. 3a (cost function) a steadier



(a) Cost Function (b) Under-sampling

Fig. 3. MLP outputs for the C03 subset, after apply the cost function and random under-sampling strategies. The line in black shows the separation between the outputs of both classes.

tendency is observed in the MLP outputs, while in the Fig. 3b shows a more irregular behavior. However, the negative effects on the cls^- are not very severe than the case of C04 subset.

These results confirm that other problems, such as the overlap between classes or noise in the TDS, should be taken into account in classification tasks when the TDS is imbalanced [12, 13].

5 Conclusion

In this paper the suitable random under-sampling and cost functions for handling the severe class imbalance was empirically studied. The results suggest that when the imbalance is severe, the random under-sampling presents a tendency to have a weak learning on the cls^-. This situation becomes graver when there are very few elements in the cls^+, then a tendency toward over fitting (over fitting cls^+) appears. However, the main cause of this situation is the existence of overlap or noise in the TDS. Also, was observed that others subsets with same imbalance level their performance was high.

Finally, under a context of severe class imbalance the best alternative is to use a cost function for compensate the class imbalance. The application of cost functions improves two fundamental aspects: they prevent a weak learning on the cls^- (because avoids losing potentially useful data) and, they increase the cls^+ participation in the MLP learning.

Acknowledgment

This work has been partially supported by the Spanish Ministry of Science and Education under project CSD2007-00018, UAEMCA-114 and SBI112 from the Mexican SEP, the 2703/2008U from the UAEM project and SDMAIA-010 from the TESJO project.

References

1. Zhou, Z.-H., Liu, X.-Y.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Transactions on Knowledge and Data Engineering 18, 63–77 (2006)
2. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. 21(9), 1263–1284 (2009)
3. Visa, S.: Issues in mining imbalanced data sets - a review paper. In: Artificial Intelligence and Cognitive Science Conference, pp. 67–73 (2005)
4. Japkowicz, N., Stephen, S.: The class imbalance problem: a systematic study. Intelligent Data Analysis 6, 429–449 (2002)
5. Lawrence, S., Burns, I., Back, A., Tsoi, A.C., Giles, C.L.: Neural network classification and prior class probabilities. In: Orr, G., Müller, K.-R., Caruana, R. (eds.) NIPS-WS 1996. LNCS, vol. 1524, pp. 299–314. Springer, Heidelberg (1998)
6. Kubat, M., Matwin, S.: Detection of oil-spills in radar images of sea surface. Machine Learning (30), 195–215 (1998)
7. Chawla, N.V., Bowyer, K.W., Hall, L.O., Philip Kegelmeyer, W.: Smote: Synthetic minority over-sampling technique. J. Artif. Intell. Res. (JAIR) 16, 321–357 (2002)
8. Japkowicz, N., Myers, C., Gluck, M.: A novelty detection approach to classification. In: Proceedings of the Fourteenth Joint Conference on Artificial Intelligence, pp. 518–523 (1995)
9. Anand, R., Mehrotra, K., Mohan, C.K., Ranka, S.: Efficient classification for multiclass problems using modular neural networks. IEEE Transactions on Neural Networks 6(1), 117–124 (1995)
10. Bruzzone, L., Serpico, S.B.: Classification of imbalanced remote-sensing data by neural networks. Pattern Recognition Letters 18, 1323–1328 (1997)
11. Anand, R., Mehrotra, K.G., Mohan, C.K., Ranka, S.: An improved algorithm for neural network classification of imbalanced training sets. IEEE Transactions on Neural Networks 4, 962–969 (1993)
12. Visa, S., Ralescu, A.: Learning imbalanced and overlapping classes using fuzzy sets. In: Workshop on Learning from Imbalanced Datasets(ICML 2003), pp. 91–104 (2003)
13. Prati, R.C., Batista, G.E.A.P.A., Monard, M.C.: Class imbalances *versus* class overlapping: An analysis of a learning system behavior. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 312–321. Springer, Heidelberg (2004)