NEW PRUNING METHODS TO MLP NEURAL NETWORKS FOR UNBALANCED CLASSIFICATION PROBLEMS

MIRIAM R. SILVESTRE[1A], LEE L. LING[2].

1. *Depto. de Estatística, UNESP Univ. Estadual Paulista, Fac. de Ciências e Tecnologia Rua Roberto Simonen, 305, 19060-900 Presidente Prudente, SP, BRASIL A. PhD Student at FEEC/Unicamp, E-mail:* miriam@fct.unesp.br

2. *Depto. de Comunicações, UNICAMP Univ. Estadual de Campinas, Fac. Eng. Elétrica e Computação Av. Albert Ainstein, 400, 13083-852 Distrito de Barão Geraldo, Campinas, SP, BRASIL E-mail:* lee@decom.fee.unicamp.br

Abstract— This article deals with classification problems involving unequal probabilities in each class and discusses metrics that can be used in systems of multilayer perceptrons neural networks (MLP) developed for the task of classifying new patterns. We proposed three new methods of pruning that were compared with other existing methods in the literature for MLP networks. All pruning algorithms presented in this paper have been modified by the author to do pruning of neurons, in order to produce fully connected MLP networks and small in its intermediary layer. Experiments were carried out involving one unbalanced classification problem and ten pruning methods. The proposed methods showed good results; and had obtained better results than another pruning methods previously defined at the mlp neural network area.

Keywords— Artificial neural networks (theory and applications), pattern recognition, unbalanced data, pruning methods, mlp neural network.

Resumo— Este artigo trata de problemas de classificação envolvendo probabilidades desiguais em cada classe e são discutidas métricas que podem ser usadas em sistemas de redes neurais de múltiplas camadas (MLP) desenvolvidos para a tarefa de classificar novos padrões. São propostos três novos métodos de poda que foram comparados com outros métodos já existentes na literatura para redes MLP. Todos os algoritmos de poda apresentados nesse artigo foram modificados pela autora para realizar poda de neurônios, com o objetivo de produzir redes MLP completamente conectadas e pequenas em sua camada intermediária. Foram realizados experimentos envolvendo um problema de classificação desbalanceado e dez métodos de poda. Os métodos propostos mostraram bons resultados, tendo obtido resultados melhores que outros métodos de poda previamente definidos na área de redes neurais.

Palavras-chave— Redes neurais artificiais (teoria e aplicações), reconhecimento de padrões, dados desbalanceados, métodos de poda, redes neurais mlp

1 Introduction

Classification problems have received great emphasis in the area of computational intelligence, which has produced systems and programs that assist professionals in making decision when classifying a new eelement in one of several possible classes. However, not all classes have the same probability of occurrence in the population. In some cases, very rare events may be determined in the class, which makes the task of classifying a new pattern very difficult. An example presented in [1] related to banking transactions with credit cards believes that 1% of transactions are fraudulent, then a model that predicts every transaction as legitimate has an accuracy of 99%, although it fails to detect any fraudulent activity. Therefore, appropriate measures should be used to focus the rare class. Section 2 discusses different metrics, and an illustrative example is shown for their comparison in unbalanced classification problems.

The multilayer neural networks, known as MLP networks are largely used to construct classifiers. A major problem for the use of MLP networks is to define the size of the initial network about their hidden layers and number of neurons. One technique is to set the initial network with more neurons than necessary thereby it is possible to obtaining a faster classifier, and implement pruning methods, which are able to determine which neurons in the network could be removed without a severe impairment of the performance of classification network. However, the pruning algorithms are targeted to cut neurons displaying only the decrease of misclassification, which is not very suitable for problems with unbalanced data.

In this paper we present in Section 3 several pruning methods known to the MLP network, and are proposed three new pruning methods. However, the algorithms are modified so that other metrics are considered most appropriate to problems involving unequal probabilities in each class. In Section 4, the ten pruning methods described are applied to the Ecoli unbalanced problem obtained from the database [2] containing seven classes.

The Section 5 presents the final conclusions for the classification problems discussed and the techniques employed.

2 Metrics for assessment of the classification

Initially, start from the Bayesian decision theory, considering a classification problem with two classes: P_1 and P_2. For each object is observed a measurement vector **x** with px1 order on the **X** random variables vector of interest. Each class presents the density probability functions $f_1(\mathbf{x})$ e $f_2(\mathbf{x})$, respectively. Let Ω be the sample space, R_1 the region where the objects are classified in Class P_1, and $R_2=\Omega-R_1$ the region destined to objects belonging to the class P_2. Since every object must be assigned to only one of two classes, the sets R_1 e R_2 are mutually exclusive and exhaustive. Let p_1 and p_2 be the prior probabilities of the classes P_1 and P_2, respectively, where $p_1+p_2=1$. The performance of a classifier can be assessed by calculating the ratio of the current error (AER):

$$AER = p_1 \int_{\hat{R}_2} f_1(\mathbf{x})d\mathbf{x} + p_2 \int_{\hat{R}_1} f_2(\mathbf{x})d\mathbf{x} \qquad (1)$$

where $\hat{R}_1$ and $\hat{R}_2$ represent the classification regions determined by the sample sizes n_1 e n_2, and the functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are known.

More details on the Bayesian decision theory can be found in [3]. An estimate of AER, given in (1), may be obtained directly from the confusion matrix constructed using the classifier. This estimate is called the apparent error rate (APER).

Given n_{1c}=number of elements in the population P_1 correctly classified, n_{2c}=number of elements in the population P_2 correctly classified, n_{1m}=number of elements in the population P_1 incorrectly classified and n_{2m}=number of elements in the population P_2 incorrectly classified, as shown in Table 1.

Table 1. Classification Matrix or Confusion Matrix.

True Class	Classified as P_1	Classified as P_2	Total
P_1	n_{1c}	n_{1m}	n_1
P_2	n_{2m}	n_{2c}	n_2
Total	$n_{1c}+ n_{2m}$	$n_{1m}+ n_{2c}$	$N= n_1+ n_2$

The apparent error rate APER is given by (2):

$$APER = (n_{1m} + n_{2m})/(n_1 + n_2)\,. \qquad (2)$$

The value of APER is usually presented as a percentage, and is obtained by the proportion of the incorrectly classified items in the training dataset. However, the APER tends to underestimate the true value of AER. But, there are other ways to evaluate the error obtained by a classifier. One method is the Leave-one-out by Lachenbruch and Mickey [3], which also requires no assumptions about the distributions. This method consists in removing an element of the sample, build the classifier using the (n1+n2)-1 remaining elements and classify the element that was left out. Repeat the procedure until all elements have been removed, and the end, calculate the error APER obtained considering all the elements that were left out. The drawback of this procedure is that it must be built a classifier at each step, and for some types of binders, such as for MLP neural networks, may be impractical depending on the overall size of the sample. This method is more suitable for classifiers that do not demand a very high computational time in the construction of a model.

Another way is to divide the data set into two parts. One part is used for training on which the classifier is constructed, and the other to validate it, and, should consider the error APER obtained in the validation dataset, also known as test dataset, as an estimate of the AER more consistent with the true value.

The disadvantage of this method is that not all samples are used to build the classifier, which may end up not considering important features of problem that could be present in the test samples left and not used in the construction of the classifier. Moreover, an advantage of this method would be that the classifier would be tested in a dataset that has not been used for its construction, a fact that would allow the calculated error APER was much closer to the true value, and thus more suited to the capacity generalization of the model or classifier.

Reference [4] presents several evaluation measures to access the performance of a classification model. The following describes some evaluation measures, and propose another measure.

Below are given some definitions that can be obtained from the confusion matrix (Table 1), which metrics will be used in this section:

- True P_1 Rate: $TP1rate = n_{1c}/(n_{1c} + n_{1m}) = n_{1c}/n_1$
- True P_2 Rate: $TP2rate = n_{2c}/(n_{2m} + n_{2c}) = n_{2c}/n_2$
- False P_1 Rate: $FP1rate = n_{2m}/(n_{2m} + n_{2c}) = n_{2m}/n_2$
- False P_2 Rate: $FP2rate = n_{1m}/(n_{1c} + n_{1m}) = n_{1m}/n_1$
- P_1 Predictive Value: $P1Pvalue = n_{1c}/(n_{1c} + n_{2m})$
- P_2 Predictive Value: $P2Pvalue = n_{2c}/(n_{1m} + n_{2c})$.

2.1 Apparent Error Rate (APER)

The classification error APER defined in (2), and obtained by the classifier is a commonly used measure. The smaller the error APER, the better the classifier, however, as we shall see below, this is not an appropriate measure to find the best classifier in problems with unequal sample sizes of classes.

2.2 F-measure

The F-measure is considered when the class P_1 is considered the most important. And in this case, the following rates are used Recall (R=Recall=*TP1rate*) and Precision (P=Precision=*PP1value*), being the measure defined by *F-measure*= 2RP/(R+P).

The measure defined above may be reduced to the harmonic mean between R and P, such as *F-measure*= 2/[(1/R)+(1/P)].

2.3 G-mean

The *G-mean* measure is indicated when both cases are also of interest. In this case, the True P_1 Rate (*TP1rate*) and the True P_2 Rate (*TP2rate*) are expected to be high simultaneously. The measure *G-mean*=$(TP1rate \times TP2rate)^{1/2}$.

The disadvantage of this metric is that if a class has null classification rate, the measure will be zero, regardless of the result in another class.

2.4 Proposed Proportional Apparent Error Rate (APERP)

We propose another measure that is able to evaluate the classification ability of a classifier, which we call proportional error rate (APERP), that concerns in the average proportions of the errors committed in each class, and *FP2rate* and *FP1rate*. The reason is given in (3):

$$APERP = \frac{(n_{1m}/n_1) + (n_{2m}/n_2)}{2}. \quad (3)$$

As such given in (3), unlike the G-mean, if one of the classes provide zero error ratio still will be possible to calculate the capacity rating for the class remainder obtained.

Example 1

Consider as an example a two-class problem, for which were built two different models for classification: Model A and Model B. The rating results from each model are presented in Tables 2 and 3, respectively.

Table 2. Classification Matrix or Confusion Matrix of Model A.

True Class	Classified as P_1	Classified as P_2	Total
P_1	250	20	270
P_2	10	20	30
Total	260	40	300

Table 3. Classification Matrix or Confusion Matrix of Model B.

True Class	Classified as P_1	Classified as P_2	Total
P_1	240	30	270
P_2	0	30	30
Total	240	60	300

In this example it will be presented a classification problem with unbalanced classes with 90% of the data belonging to the class P_1 and 10% to the class P_2. It will be applied only the APER metrics and the APERP proposed measure due to unsuitability of the measure F-measure for dealing with problems where both classes are important, and due to the replacement of the metric G-mean by APERP for the reasons already mentioned in the definition of thereof.

The classification error apparent APER metric presented in (2) is estimated by the total obtained in the secondary diagonal, divided by the total sample. The classification error obtained by the Model A is:

$APER_A$ = $(n_{1m}+n_{2m})/(n_1+n_2)$ =(20+10)/(270+30) = 30/300=0.1.

The same error value is obtained by the Model B:

$APER_B$ = $(n_{1m}+n_{2m})/(n_1+n_2)$ =(30+0)/(270+30) = 30/300=0.1.

If we calculate the expected correctly classification rate, which is complementary to the error rate, we would have 0.9 or 90% accuracy for both models.

However, even with misclassification APER being equal, it is noted that Model B is more able to classify the elements of Class 2 than Model A, with a moderate decrease in class 1 correct classification. This can be observed when calculating the proportions of errors made in each class separately. For Model A, the errors would $APERP_A = 0.2037$ and for Model B, $APERP_B = 0.0555$.

Therefore the models A and B produced correct classification (1-APER) for 79.63% and 94.44%, respectively. It can be note clearly that the model B presented the best result for the two models, as it was expected to observe the data in Tables 2 and 3. Thus, it can be concluded that the classification measure APER as defined in (2) is not suitable because it results the same value for both models, while the model B is better than A.

Thus, the apparent proportional error ratio (APERP) proposed in (3), is more suitable for unbalanced problems where there is interest in both classes. A generalization to a classification problem of *c* classes is: APERP = $[(n_{1m}/n_1) + (n_{2m}/n_2) + ... +(n_{cm}/n_c)] / c$. This measure is the average of the error rate observed in each class, and it belongs to the range of 0 to 1, or, in terms of percentage range of 0 to 100%. Thus, when selecting a classifier based solely on the error APER produced we may be failing to choose the best classifier. And in that sense, all results involving this measure to select classifiers with unequal class sizes should be reviewed.

3 Pruning Methods to MLP Neural Networks

There are some neural network models [5], in this article we will consider a MLP network, with one hidden layer (SLFN). Let $[\mathbf{x}(n),\mathbf{t}(n)]$ denote the n^{th} (*n*

= 1, ..., N) training pattern where $\mathbf{x}(n)=[\mathbf{x}_1(n), \ldots, \mathbf{x}_d(n)]^T$ is the d-dimensional input feature vector and $\mathbf{t}(n)=[t_1(n), \ldots, t_c(n)]^T$ corresponds to the desired c-class output response. Let $\mathbf{W}^{(1)}=\{w_{ji}^{(1)}(n)\}$ and $\mathbf{W}^{(2)}=\{w_{kj}^{(2)}(n)\}$ be the matrix of synaptic weights which connect the input to hidden layers and the hidden to the output layers, respectively. The optimization procedure for classification consists of minimizing the sum of the mean square errors between the network output $y_k(n)$ and the desired output $t_k(n)$, defined as $E = \sum_{n=1}^{N} E(n)$, where:

$$E(n) = \frac{1}{2}\sum_{k=1}^{c}\left[y_k(n) - t_k(n)\right]^2 . \qquad (4)$$

Assuming that $g^{(1)}$ and $g^{(2)}$ are activation functions of sigmoid logistic type (5):

$$g^{(.)}(v_j(n)) = \frac{1}{1+\exp(-pv_j(n))} . \qquad (5)$$

The k^{th} network output $y_k(n)$ is given by:

$$y_k(n) = g^{(2)}\left(\sum_{j=0}^{M} w_{kj}^{(2)}(n) g^{(1)}\left(\sum_{i=0}^{d} w_{ji}^{(1)}(n) x_i(n)\right)\right), \qquad (6)$$

where M is the number of hidden neurons.

The pruning methods that will be described later, will use the output of the j^{th} neuron in the hidden layer (j=1,...,M), defined as:

$$z_j(n) = g^{(1)}\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i(n)\right) . \qquad (7)$$

The main goal of pruning methods is to reduce the network architecture in order to perform classification with speed and economy. When the pruning methods are constructed to remove the synaptic weights, the end result is the production of a large and sparsely connected network with multiple zero weights. However, these weights remain physically on the network. To prevent this from occurring, the pruning methods presented in this section have been adapted by the author to theoretically perform pruning of neurons, and be able to produce final fully connected networks because neurons that contribute little or nothing to the classification are excluded from the network. Thus, the network end will be much smaller and the classification of a new pattern will be much faster.

In [6], we performed a comparative experiment involving the initial network, called control method, for which there is no pruning method applied, and various methods of iterative pruning and non-iterative. Iterative methods remove a neuron in each step, and the non-iterative methods are executed in a single step. Through experimental results, it was found that only the iterative pruning methods were able to reduce the number of neurons. Following are the pruning methods that will be used in this work.

3.1 Pruning Methods previously defined at the mlp neural network area

3.1.1 Iterative Brutal Force (FBI)

The method called Brutal Force (FB), initially proposed by [7], was restructured theoretically in [8] to detect and remove neurons that have little influential. This method consists in defining a neuron with zero and evaluate the change caused the error E given in (4), using the measurement

$$S_j = \sum_{j=1}^{N} S_j(n) \text{, tal que, } S_j=E(z_j(n)=0)-E(z_j(n)).$$

When we define $E(z_j(n)=0)$, all synaptic weights connecting neuron j to the input and output layers are reset to zero, and an approximate value for E is obtained for a network without the presence of that neuron. And now $(z_j(n))$ is the value of E with all weights of the network associated with this neuron. If S_j is small, then the neuron j can be removed from the network. This will be done using the procedure searches backward (BPT), removing a neuron in each step of pruning. This method was called FBI.

3.1.2 Apparent Rate Error Training (APERT)

The method APERT also proposed in [8] uses the ratio of observations that were classified incorrectly by the network (due to the apparent error: APER), which will then be calculated on the training set. The variable S_j=APER(z_j=0)-APER(z_j) is adopted as a measure of the importance of each neuron. The error ratio is expressed as a percentage, which in general is easily calculated from the confusion matrix. The pruning process is iterative and is removed from the neuron with the smallest value S_j at each step of pruning.

3.1.3 Optimal Brain Damage (OBD)

In the OBD pruning method proposed by [9], the measure of importance of each neuron is given by $S_j = \sum_{j=1}^{N} \Delta E(z_j(n))$, where:

$$\Delta E(z_j(n)) = \frac{1}{2}\frac{\partial^2 E(n)}{(\partial z_j(n))^2}\left[\Delta(z_j(n))\right]^2 . \qquad (8)$$

The derivatives are obtained by the chain rule and it is defined $\Delta(z_j(n)) = -z_j(n)$.

3.1.4 Saliency (SAL)

The SAL method is defined in [10] and it adds the first derivative to (8), as defined in (9), such that:

$$\Delta E(z_j(n)) = \frac{\partial E(n)}{\partial z_j(n)}\Delta(z_j(n)) + \frac{1}{2}\frac{\partial^2 E(n)}{(\partial z_j(n))^2}\left[\Delta(z_j(n))\right]^2 . \qquad (9)$$

The method is iterative and removes of the network that neuron j with the smallest value of S_j at each step.

3.1.5 Dynamic Pruning (PD)

This method was first time proposed in [11], and uses the smaller measure:

$$S_j = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{c}\left(w_{kj}^{(2)}(n)\right)^2 z_j^2(n)\,, \qquad (10)$$

to detect the neuron that must to be removed in each iterative pruning pass.

3.1.6 Iterative Pruning (PI)

This method was defined in [12] and uses the same measure the PD, given in (10), to select the neuron that must be removed from the network. However, the adjustment of synaptic weights remaining is done by the conjugate gradient algorithm of the preconditioned normal equation.

3.1.7 Pruning based on Bayesian Decision Boundaries (PEB)

This method is proposed in [8], and the measure of salience used is similar to the measure used in the PD method given in (10), except for the exclusion of the quadratic terms in this way, the measure S_j represents all synaptic activity, excitatory and inhibitory related to neuron *j* in the exact form as it occurs:

$$S_j = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{c} w_{kj}^{(2)}(n) z_j(n)\,. \qquad (11)$$

3.2 Proposed Pruning Methods

In this section it will be presented tree new pruning methods: the APERTP, the KAPPA and the CHI method.

3.2.1 Apparent Training Proportional Error Rate (APERTP)

The APERTP method uses the metric APERP, equation (3), or its generalization to more than two classes, and calculated on the set of training data as a measure of S_j and removes the network neuron *j* with the lowest value S_j in each step of pruning.

3.2.2 KAPPA

This method uses the Kappa coefficient of agreement proposed by [13] as a measure of S_j. In [14], the Kappa measure was used to evaluate the errors obtained directly on the confusion matrix for image classification problems, but in this paper we propose to use it as a measure of importance of a neuron in the network on the task of classifying a pattern.

The Kappa coefficient is calculated from the values of the confusion matrix obtained by a classifier, and the higher the value the better the agreement between the rows and columns of the matrix, i.e., the better the performance of the classifier. A zero value of Kappa coefficient means that the successes of the classifier were by chance, while a positive value indicates that the Kappa hits were not by chance and a negative value means that the successes of the classifier are worse than classification by chance and a value of one means perfect classification.

To a problem with *c* classes the measure S_j is defined as:

$$S_j = \left(N\sum_{k=1}^{c} n_{kk} - \sum_{k=1}^{c} n_{k.}n_{.k}\right)\Bigg/\left(N^2 - \sum_{k=1}^{c} n_{k.}n_{.k}\right), \qquad (12)$$

where $n_{k.}$ is the sum of the k^{th} row of the confusion matrix, and on the other hand, $n_{.k}$ is the sum of the k^{th} column. The higher the value of S_j the better is the network performance in the classification.

To find which neuron has to be removed at each step of pruning, S_j is calculated without the neuron j to be present on the network. This is done by zeroing out all the synaptic weights connecting the neuron *j* to the neuron input layer and the network output layer. The measure S_j must be calculated for all network neurons intermediaries. The neuron *j* that present the greatest value S_j must be removed from the network, because it means that without his presence the classification performance of the network was best evaluated among neurons in that step.

3.2.3 Chi-Squared (CHI)

In this method, we suggest the use of χ^2 measure of the statistical test Chi-Squared Pearson [15], used for test the independence between the rows and columns of a contingency table, in our case, we consider the table as the confusion matrix. The higher the value of the statistic, the better the performance of classification, and the greater the frequency of elements on the main diagonal of the matrix, exactly where are positioned the correct classifications. We must calculate the value of the statistic for each one of the intermediary neuron in the network, and consider $E_{kl} = n_{k.}n_{.l}/n$ and S_j according to equation (13):

$$S_j = \chi^2 = \sum_{k=1}^{c}\sum_{l=1}^{c}\frac{(n_{kl} - E_{kl})^2}{E_{kl}}\,. \qquad (13)$$

4 Application for unbalanced classification problem

The Ecoli classification problem presented in this section are available in [2] and contain seven imbalance classes. They were defined as N = 336 sample units, d=7 features (variables) and c=8 classes: Class 1 (cytoplasm), Class 2 (inner membrane without the signal sequence), Class 3 (periplasm), Class 4 (inner membrane with no signal sequence attached), Class 5 (outer membrane), Class 6 (outer membrane lipoprotein), Class 7 (inner membrane lipoprotein) and Class 8 (with the inner membrane attached signal sequence). The experiment consisted on training 10 mlp neural networks containing 15 neurons at the intermediary layer, and it was applied all pruning methods described in the Section 3. The training dataset contain 252 observations and the test data has 84 patterns.

The dataset were divided into two parts: one containing approximately 75% of the observations for training and the remaining 25% for testing, respecting the internal percentage of each class. It was applied the z-transformation to the variables, i.e., all variables were transformed to have zero mean and unit variance.

We used the following parameters in the definition of MLP network: we used the logistic sigmoid activation function defined in (9), with the slope parameter p=0.8; the backpropagation algorithm for training without regularization parameter; the learning ratio used was equal to 0.9; the time constant was equal to zero and stopping criterion of training as the Euclidean norm of the gradient vector estimate of proposed [5] less than or equal to 0.05. It was generated 10 sets of uniform initial weights between the range [-0.1; 0.1].

Ten pruning methods were applied to MLP networks already trained and named as Control network. It was considered the APERP metric applied to the training set rather than APER when evaluating the best network, and noted the value of APERP in the test set. This is a differential analysis because studies involving pruning found in previously mentioned references, used the APER measurement error to assess the ability of generalization of MLP networks built, and as seen in Section 2 that is not the metric most suitable for unbalanced data.

The Figure 1 contains a boxplot graphic for each pruning method. Looking at Figure 1 we can see that the results of APERP for the test set are high, but between the proposed methods, the methods KAPPA and CHI show better results as the median (line inside the box) which are less than the method control. We can note that the PI method is very close to the values of the control network, which illustrates its characteristic not to favor the reduction of the error APERP, but keeping it at the same level of network Control.

5 Conclusions

In this article we presented reasons to use the metric proportional APERP instead of APER. We proposed three new methods for pruning MLP which can be easily programmable in computer languages. The proposed methods were applied to the Ecoli dataset classification problem containing seven unbalanced classes. From the results obtained in the experiments it can be concluded that the proposed methods showed good results compared to the others, however, the KAPPA and CHI methods were better than APERTP in relation to the new metrics APERP proposed.

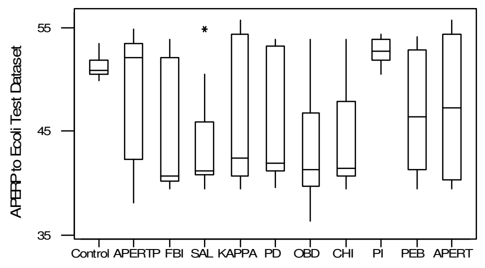

Figure 1. APERP of the pruning methods applied to Ecoli test dataset.

References

[1] P-N. Tan; M. Steinbach, V. Kumar (2009). Introdução ao Datamining Mineração de Dados. Ciência Moderna Ltda, Rio de Janeiro.

[2] Bache, K.; Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[3] Johnson, R. A.; Wichern, D. W. (2007). Applied multivariate statistical analysis. 6th ed, Pearson Education.

[4] Sun, Y.; Kamel, M. S.; Wong, A. K. C.; Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. Pattern Recognition, vol. 40, pp. 3358-3378..

[5] Haikin, S. (1994). Neural Networks: a comprehensive foundation. Wiley: New Jersey.

[6] Silvestre, M. R.; Ling, L. L. (2006). Avaliação estatística de métodos de poda aplicados em neurônios intermediários da rede neural MLP. IEEE Latin American Transactions, vol.4, n.4, pp.247-254.

[7] Karnin, E. D. (1990). A simple procedure for poda backpropagation trained neural networks. IEEE Trans. Neural Networks, vol.1, n.2, pp.239-242.

[8] Silvestre, M. R.; Ling, L. L. (2002). Optimization of neural classifiers based on Bayesian decision boundaries and idle neurons poda. Proc. International Conf. Pattern Recognition pp.387-390.

[9] Lecun, Y.; Denker, J. S.; Solla, S. A. (1990). Optimal brain damage. Advances in Neural Information Processing, vol.2, pp. 598-605.

[10] Mao J.; Mohiuddin, K.; Jain, A. K. (1994). Parsimonious network design and feature selection through node poda. Proc. International Conf. Pattern Recognition, pp. 622-624.

[11] Murase, K.; Matsunaga, Y.; Nakade, Y. (1991). A backpropagation algorithm which automatically determines the number of association units. Proc. International Conf. Neural Networks, pp. 783-788.

[12] Castellano, G.; Fanelli, A. M.; Pelillo, M. (1997). An iterative poda algorithm for feedforward neural networks. IEEE Trans. Neural Networks, vol.8, n.3, pp.519-531.

[13] Cohen, J. A. (1960). A coeficient of agreement for nominal scales, Journal of Educational and Measurement, Washington, vol. 20, n.1, pp. 37-46.

[14] Hudson, W. D.; Ramm, C. W. (1987). Correct formulation of the kappa coefficient of agreement. Photogrammetric Engineering & Remote Sensing, Maryland, vol. 53, n.4, pp. 421-422.

[15] Mood, A. M.; Graybill, F. A.; Boes, D. C. (1974). Introduction to the Theory of Statistics. 3ed. McGraw-Hill.