Classification of Binary Imbalanced Data Using A Bayesian Ensemble of Bayesian Neural Networks

Marcelino Lázaro[1(✉)], Francisco Herrera[2], and Aníbal R. Figueiras-Vidal[1]

[1] Department of Signal Theory and Communications,
Carlos III University of Madrid, 28911 Leganés, Spain
mlazaro@tsc.uc3m.es
[2] Department of Computer Science and Artificial Intelligence,
University of Granada, 18071 Granada, Spain

Abstract. This paper presents a new method to deal with classification of imbalanced data. A Bayesian ensemble of neural network classifiers is proposed. Several individual neural classifiers are trained to minimize a Bayesian cost function with different decision costs, thus working at different points of the Receiver Operating Characteristic (ROC). Decisions of the set of individual neural classifiers are fused using a Bayesian rule that introduces a "balancing" parameter allowing to compensate the imbalance of available data.

Keywords: Imbalanced data · Classification · Neural networks · Bayes risk

1 Introduction

Pattern classification is the act of taking in raw data and making a decision to assign it to a category or class [1]. There are many real world classification tasks in industry, business and science. In many cases, classification task has to be learned from an available labeled data set, containing samples of the objects to be classified along with their corresponding class labels. Machine learning methods, and specifically neural networks, have been extensively used to solve this kind of classification problems [2,3].

In last years, the classification of imbalanced data has attracted a great attention. Imbalanced data means that the number of instances of each class are very different. Focusing in a binary classification problem, learning from imbalanced data has the difficulty of representing the minority class, that can be shrouded in the thicker cloud of samples of the majority class. This is a serious concern in applications where the importance of detecting the minority class is high, such as medical or fraud detection applications.

This work has been partially supported by Research Grant S2013/ICE-2845 (CASI-CAM-CM), DGUI - Comunidad de Madrid

L. Iliadis and C. Jayne (Eds.): EANN 2015, CCIS 517, pp. 304–314, 2015.
DOI: 10.1007/978-3-319-23983-5_28

Different techniques have been used to face the problem of classification of imbalanced data: standard algorithms modified to compensate imbalance; preprocessing techniques that modify the data set to balance it (removing samples of the majority class, or introducing synthetic samples of the minority class); or ensembles of classifiers to improve the accuracy of individual classifiers. A detailed review of several of these methods can be found in [4,5]. In particular, ensembles of classifiers have shown excellent results [5,6].

Bayesian theory [7] allows to specify different costs for errors classifying each class, and at the same time to take into account the prior probability of each class. This formulation can be usefull to deal with imbalanced data.

In this paper, we will introduce a new ensemble method to deal with imbalanced binary classification problems. A "balancing" parameter is introduced in the design of the individual classifiers and in the design of the ensemble. This "balancing" parameter, based on the Bayesian formulation, allows to specify the relative importance of errors in the decision for each class.

2 Bayes Risk and Training of Neural Networks

A classification problem consists in assigning a D-dimensional pattern $\mathbf{x}$ (instance or sample) to one out of a known set $\mathcal{H}$ of possible classes or hypotheses. In a binary classification problem only two classes are possible, namely $\mathcal{H} = \{0, 1\}$. Bayesian formulation considers the a priori class probabilities along with the different costs of each possible decision for samples of every class. The goal of a Bayesian classifier is to minimize the Bayesian risk function, which includes the statistical average of these costs [7,8]

$$\mathcal{R} = \sum_{t \in \mathcal{H}} \sum_{d \in \mathcal{H}} \pi_t \; c_{d,t} \; p_{\hat{\mathcal{H}}|\mathcal{H}}(d|t) \tag{1}$$

where π_t denotes the probability of hypothesis t, $c_{d,t}$ is the cost of deciding hypothesis d when the true hypothesis is t, and $p_{\hat{\mathcal{H}}|\mathcal{H}}(d|t)$ denotes the conditional probability of this decision

$$p_{\hat{\mathcal{H}}|\mathcal{H}}(d|t) \equiv P(\text{Decide } d \mid t \text{ is true}) \tag{2}$$

The classifier minimizing the Bayesian risk is defined by the likelihoods (conditional distributions of input pattern under both hypothesis, $f_{\boldsymbol{\mathcal{X}}|\mathcal{H}}(\mathbf{x}|t)$, for $t \in \{0, 1\}$), as well as the a priori class probabilities and decision costs. The optimal decision rule minimizing Bayesian risk is

$$\Lambda(\mathbf{x}) = \frac{f_{\boldsymbol{\mathcal{X}}|\mathcal{H}}(\mathbf{x}|1)}{f_{\boldsymbol{\mathcal{X}}|\mathcal{H}}(\mathbf{x}|0)} \underset{\hat{\mathcal{H}}=0}{\overset{\hat{\mathcal{H}}=1}{\gtrless}} \frac{(c_{1,0} - c_{0,0})}{(c_{0,1} - c_{1,1})} \frac{\pi_0}{\pi_1} = \gamma \; (> 0) \tag{3}$$

i.e., a test comparing the likelihood ratio (LR) with a threshold γ given by costs $c_{d,t}$ and prior probabilities π_t [7,8].

A binary classifier is usually characterized by the false alarm and the miss probabilities (probabilities of erroneous decisions under both hypothesis)

$$p_{FA} = p_{\hat{\mathcal{H}}|\mathcal{H}}(1|0) \equiv P(\text{Decide } 1 \mid 0 \text{ is true}) \tag{4}$$

$$p_M = p_{\hat{\mathcal{H}}|\mathcal{H}}(0|1) \equiv P(\text{Decide } 0 \mid 1 \text{ is true}) \tag{5}$$

Obviously, different values of γ originate different pairs of values for p_{FA} and p_M, which show a compromise: reducing one of them means increasing the other. The Receiver Operating Characteristic (ROC) is a curve that represents $p_D = 1 - p_M$ vs. p_{FA} for $0 \leq \gamma < \infty$, i.e., draws the working points (p_{FA}, p_D) from $(0, 0)$ to $(1, 1)$ [9]. The Area Under this Curve (AUC) is a parameter that serves to compare different classifier designs when $c_{d,t}$ or π_t are not given [10,11].

In binary classification, Bayesian formulation can be simplified assuming $c_{1,1} = c_{0,0} = 0$, and parameterizing the other two costs by means of a single parameter α as follows

$$c_{1,0} = \frac{\alpha}{\pi_0},\ c_{0,1} = \frac{1-\alpha}{\pi_1} \tag{6}$$

Using this parameterization, the Bayesian risk becomes

$$\mathcal{R}(\alpha) = \alpha\, p_{FA} + (1-\alpha)\, p_M \tag{7}$$

and decision threshold is now given by $\gamma = \alpha/(1-\alpha)$. Parameter α establishes the relative importance given to errors for samples of both classes.

In practice, in many real problems distributions involved in (3) are unknown, and classifiers have to be designed from a set of labeled observations, $\{\mathbf{x}_k, y_k\}$, $k \in \{1, 2, \cdots, N\}$, with binary class labels y_k. In this case, a machine architecture with a number of trainable parameters, $\mathbf{w}$, can be used to obtain a classification rule. Machine learning methods, and in particular neural networks, can be used in several different ways to solve binary classification problems [3]. Here, architecture to implement the classifier is constrained to neural networks with a single output and a threshold based decision.

For these networks, typically labels $y_k = -1$ and $y_k = +1$ are associated to samples belonging to hyphotesis $\mathcal{H} = 0$ and $\mathcal{H} = 1$, respectively. The soft output of the network is

$$z_k = g(\mathbf{x}_k, \mathbf{w}) \tag{8}$$

where $g(\mathbf{x}, \mathbf{w})$ is the parameter-dependent non-linear transfer function of the neural network, which depends on the specific network architecture. Finally, the decision rule of the neural classifier, in terms of the given binary labels, is

$$\hat{y}_k = \text{sgn}(z_k) \tag{9}$$

where $\text{sgn}(\cdot)$ is the well known sign function.

The design of the neural classifier consists in obtaining the values of neural network parameters, $\mathbf{w}$. Frequently, $\mathbf{w}$ is obtained by imposing the minimization of some appropriate cost function $J(\mathbf{w})$ that measures the difference between z_k and y_k. If cost is conveniently selected, network output z_k can provide an

estimate of the posterior probabilities for each class and there is a relationship with Bayesian formulation (see [3,12] for more details). This is the case of the Mean Squared Error (MSE) cost function

$$J^{MSE}(\mathbf{w}) = \frac{1}{N}\sum_{k=1}^{N}(y_k - z_k)^2 \tag{10}$$

probably the most used cost function because it can be used to minimize the probability of error (Bayesian risk for $c_{0,1} = c_{1,0}$, or equivalently $\alpha = \pi_0$). This cost is useful for balanced problems, but for imbalanced problems, with $\pi_0 >> \pi_1$, reducing p_{FA} has a bigger impact in cost than reducing p_M, thus resulting in a poor performance for the minority class.

Typically, the cost function is iteratively minimized by using a gradient descent method to adapt parameters from iteration $(i-1)$ to iteration i

$$\mathbf{w}^{(i)} = \mathbf{w}^{(i-1)} - \mu\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} \tag{11}$$

It is interesting to remark that gradient expression in updating equation (11) can be written as

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \sum_{k=1}^{N}\frac{\partial J(\mathbf{w})}{\partial z_k}\,\frac{\partial z_k}{\partial \mathbf{w}} \tag{12}$$

Term $\frac{\partial z_k}{\partial \mathbf{w}}$ depends on network architecture, in particular on the dependence of neural network output on network parameters. Term $\frac{\partial J(\mathbf{w})}{\partial z_k}$ is the one including the dependence on the cost function. For MSE cost function

$$\frac{\partial J^{MSE}(\mathbf{w})}{\partial z_k} = -\frac{2}{N}(y_k - z_k) \tag{13}$$

3 Proposed Methods

In this communication we propose a classifier based on the ensemble of several binary neural network classifiers. The neural classifiers as well as the ensemble classifier will be based on the Bayes risk. In this section, first the individual classifiers will be presented, and then the ensemble classifier will be formulated.

3.1 Individual Bayesian Neural Network Classifiers

In the design of the individual classifiers that will compose the ensemble, the aim is to find the network parameters minimizing an estimate of Bayes risk (7) for a given value of parameter α when decision rule is subject to be provided by a neural network like the one presented in Section 2, i.e., given by (8) and (9).

Considering decision rule (9), false alarm (false positive) and miss (false negative) probabilities defining risk (7) for the neural classifier are

$$p_{FA} = \int_0^{\infty} f_{\mathcal{Z}|\mathcal{H}}(z|0)\,dz,\; p_M = \int_{-\infty}^{0} f_{\mathcal{Z}|\mathcal{H}}(z|1)\,dz \tag{14}$$

In general, conditional distributions on $\mathcal{Z}$, which models network output (8), are unknown. The proposed training method will estimate these distributions from available data using Parzen window estimator [13]. If sets $\mathcal{S}_0$ and $\mathcal{S}_1$ contain indexes for data corresponding to hypothesis $\mathcal{H} = 0$ and $\mathcal{H} = 1$, respectively

$$\mathcal{S}_0 = \{k : y_k = -1\} \text{ and } \mathcal{S}_1 = \{k : y_k = +1\} \tag{15}$$

and N_0 and N_1 denote the number of samples in each set, Parzen window estimate for conditional distribution on $\mathcal{Z}$ given $\mathcal{H} = t$ is

$$\hat{f}_{\mathcal{Z}|\mathcal{H}}(z|t) = \frac{1}{N_t} \sum_{k \in \mathcal{S}_t} K_\sigma(z - z_k),\ t \in \{0, 1\} \tag{16}$$

The window or kernel $K_\sigma(z)$ is any valid probability density function (PDF) with parameter σ controling its width. Typically, symmetric zero mean distributions, such as Gaussians, are used for estimation. Replacing $f_{\mathcal{Z}|\mathcal{H}}(z|t)$ in (14) by its Parzen estimation (16) to estimate p_{FA} and p_M

$$\hat{p}_{FA} = \int_0^\infty \hat{f}_{\mathcal{Z}|\mathcal{H}}(z|0)\ dz,\ \hat{p}_M = \int_{-\infty}^0 \hat{f}_{\mathcal{Z}|\mathcal{H}}(z|1)\ dz \tag{17}$$

and inserting now these estimates in (7), the proposed cost function is

$$J^{Bayes}(\mathbf{w}) = \alpha\ \hat{p}_{FA} + (1 - \alpha)\ \hat{p}_M \tag{18}$$

By defining function $L_\sigma(x)$ from integration of Parzen kernel function,

$$L_\sigma(x) = \int_x^{+\infty} K_\sigma(z)\ dz \tag{19}$$

and taking into account that for symmetric zero mean PDFs $K_\sigma(x)$

$$\int_0^{+\infty} K_\sigma(x - z)\ dx = L_\sigma(-z) \text{ and } \int_{-\infty}^0 K_\sigma(x - z)\ dx = L_\sigma(+z) \tag{20}$$

the proposed cost function can be written as

$$J^{Bayes}(\mathbf{w}) = \frac{\alpha}{N_0} \sum_{k \in \mathcal{S}_0} L_\sigma(-z_k) + \frac{1 - \alpha}{N_1} \sum_{k \in \mathcal{S}_1} L_\sigma(+z_k) \tag{21}$$

It is straightforward to obtain the derivative of this cost function with respect to network output

$$\frac{\partial J^{Bayes}(\mathbf{w})}{\partial z_k} = a_k\ K_\sigma(z_k),\ \text{with } \begin{cases} \alpha/N_0, & \text{if } y_k = -1 \\ (\alpha - 1)/N_1, & \text{if } y_k = +1 \end{cases} \tag{22}$$

Finally, gradient with respect to network parameters is given by

$$\frac{\partial J^{Bayes}(\mathbf{w})}{\partial \mathbf{w}} = \sum_{k=1}^{N} \frac{\partial J^{Bayes}(\mathbf{w})}{\partial z_k}\ \frac{\partial z_k}{\partial \mathbf{w}} \tag{23}$$

Again, note that $\frac{\partial z_k}{\partial \mathbf{w}}$ depends on the neural network specific architecture. From an implementation viewpoint, it is important to remark that with respect to the training of a neural network using MSE cost function, the only difference in the proposed training algorithm is that (13) has to be replaced by (22) in the computation of gradient for each iteration.

3.2 Ensemble Bayesian Classifier

The proposed classifier is a Bayesian ensemble of N_c individual neural network classifiers. Each individual classifier will be trained with the procedure presented in Section 3.1, for a different value for parameter α. The values for α used to train each individual classifier will be denoted as $\alpha^{(j)}$, with $j \in \{1, 2, \cdots, N_c\}$ This means that each individual classifier will try to work at a different point of the ROC of the classification problem, which is given by a different pair of probabilities of false alarm and detection, thus having N_c different points

$$\{(p_{FA}^{(j)}, p_D^{(j)})\} \text{ for } j \in \{1, 2, \cdots, N_c\} \tag{24}$$

This provides diversity in the decisions of individual classifiers. After training each individual neural classifier, these probabilities can be estimated from the training set, or from a validation set if one is available.

The proposed classifier will fuse the decisions of the previous N_c individual classifiers using a Bayesian rule. Therefore, to classify a given pattern $\mathbf{x}_k$, the input for this ensemble classifier is the vector containing the N_c decisions provided by the individual neural network classifiers for this pattern. If $\hat{y}_k^{(j)}$ denotes the binary decision of the j-th individual classifier for input pattern $\mathbf{x}_k$, the input of the ensemble for pattern $\mathbf{x}_k$ is

$$\mathbf{x}_k^E \equiv \mathbf{x}_k^E(\mathbf{x}_k) = \left[\hat{y}_k^{(1)}, \hat{y}_k^{(2)}, \cdots, \hat{y}_k^{(N_c)}\right] \tag{25}$$

A likelihood ratio can be defined for this input of the ensemble. Conditional distributions of the decisions of the individual classifiers are given by

$$p_{\hat{\mathcal{Y}}|\mathcal{H}}^{(j)}(\hat{y}_k^{(j)}|0) = \begin{cases} p_{FA}^{(j)}, & \text{if } \hat{y}_k^{(j)} = +1 \\ 1 - p_{FA}^{(j)}, & \text{if } \hat{y}_k^{(j)} = -1 \end{cases} \tag{26}$$

and

$$p_{\hat{\mathcal{Y}}|\mathcal{H}}^{(j)}(\hat{y}_k^{(j)}|1) = \begin{cases} p_D^{(j)}, & \text{if } \hat{y}_k^{(j)} = +1 \\ 1 - p_D^{(j)}, & \text{if } \hat{y}_k^{(j)} = -1 \end{cases} \tag{27}$$

Using these conditional distributions, and assuming conditional independence between the output of the N_c classifiers, the likelihood ratio for $\mathbf{x}_k^{(E)}$ is given by

$$\Lambda\left(\mathbf{x}_k^E\right) = \prod_{j=1}^{N_c} \frac{p_{\hat{\mathcal{Y}}|\mathcal{H}}^{(j)}(\hat{y}_k^{(j)}|1)}{p_{\hat{\mathcal{Y}}|\mathcal{H}}^{(j)}(\hat{y}_k^{(j)}|0)} = \prod_{j=1}^{N_c} \frac{p_D^{(j)}\ \delta[\hat{y}_k^{(j)} - 1] + \left(1 - p_D^{(j)}\right)\ \delta[\hat{y}_k^{(j)} + 1]}{p_{FA}^{(j)}\ \delta[\hat{y}_k^{(j)} - 1] + \left(1 - p_{FA}^{(j)}\right)\ \delta[\hat{y}_k^{(j)} + 1]} \tag{28}$$

where the discrete-time delta function $\delta[n]$ is used to provide a compact expression for (26) and (27). The final decision of the ensemble for input $\mathbf{x}^E$ will be obtained comparing the likelihood ratio with the threshold associated with the value of parameter α defined for the Bayesian risk of the ensemble, α^E

$$\Lambda(\mathbf{x}^E) \underset{\hat{\mathcal{H}}=0}{\overset{\hat{\mathcal{H}}=1}{\gtrless}} \gamma^E, \text{ with } \gamma^E = \frac{\alpha^E}{1-\alpha^E} \tag{29}$$

4 Experiments

This section presents the results obtained with the proposed method in the classification of several imbalanced databases.

4.1 Databases

We have tested the proposed method with several imbalanced real-world databases obtained from KEEL data-set repository [14]. Data and information about these data sets can be found in the Website http://www.keel.es/dataset.php. Data sets in [14] are organized in different k-fold partitions for training and test data. Here, we have worked with the 5-fold partition provided in KEEL-dataset repository, thus making easier to compare results. Results obtained for these 5 folds will be averaged. Table 1 shows the main characteristics of the tested databases.

4.2 Implementation Details

The goal of this paper is to demonstrate the intrinsic potential of the proposed method to work with different data. Therefore, instead of looking for the best configuration for each database, a generic setup has been used for all databases.

The only pre-processing for input data is normalization, forcing zero mean and unit variance for each dimension. Multilayer perceptrons (MLP) with a single hidden layer of N_n neurons, a single neuron in the output layer, and hyperbolic

Table 1. Description of the databases used for experiments.

Database	Number of patterns	Dimensionality	Imbalance Ratio
Yeast05679vs4	528	8	9.35
Ecoli067vs5	220	6	10.00
Glass2	214	9	10.39
Led7digit	443	7	10.97
Cleveland0vs4	177	13	12.62
Yeast4	1484	8	28.41
Yeast5	1484	8	32.78
Yeast6	1484	8	39.15

tangent activation functions, are used for the individual classifiers of the ensemble. Transfer function (8) and gradient expressions $\frac{\partial z_k}{\partial \mathbf{w}}$ in (23) are well known for this architecture [15,16], and have not been included due to lack of space. An adaptive step-size μ has been used in gradient updating equation (11). After each epoch, cost $J^{Bayes}(\mathbf{w}^{(i)})$ is evaluated and compared with $J^{Bayes}(\mathbf{w}^{(i-1)})$

- If $J^{Bayes}(\mathbf{w}^{(i)}) < J^{Bayes}(\mathbf{w}^{(i-1)})$: step size is increased, $\mu = c_I\ \mu$
- If $J^{Bayes}(\mathbf{w}^{(i)}) \geq J^{Bayes}(\mathbf{w}^{(i-1)})$: step size is decreased, $\mu = \mu/c_D$, and weights $\mathbf{w}^{(i)}$ are re-computed with the new step size.

$c_I = 1.05$ and $c_D = 2$ have been used in all experiments, with initial value $\mu = 1$. Relatively small networks have been used: MLP networks with $N_n = 4$ neurons in the hidden unit. Parzen window used in these experiments (to define cost and to compute (22)) has been, with $\sigma = 1$ in all cases

$$K_\sigma(z) = \begin{cases} \frac{1}{2\sigma}\left(1 + \cos\left(\frac{\pi}{\sigma}|z|\right)\right), & |z| \leq \sigma \\ 0, & |z| > \sigma \end{cases} \tag{30}$$

An ensemble of 9 individual neural network classifiers has been implemented, with the goal of showing advantage against ensembles with a higher number of elements. Parameters defining the Bayes risk objective for each one of the individual neural networks are $\alpha^{(j)} = 0.1 \times j$, for $j \in \{1, 2, \cdots, 9\}$. Network parameters $\mathbf{w}$ are randomly initialized, with values drawn from independent Gaussian distributions, with zero mean and variance 0.1.Not a single validation mechanism has been considered for training of individual neural classifiers, whereas 2000 epochs have been used for training every individual network. This generic setup will allow to assess the capability of the proposed method to deal with problems with a reduced number of samples, where a validation set can not be available.

4.3 Experimental Results

100 independent Monte Carlo simulations, starting with different initial parameters for individual neural classifiers, have been performed for each one of the 5 folds of every database. Average results obtained in the 5 folds, along with standard deviations, will be presented.

Table 2 compares the AUC of the ensemble with the AUC given by individual classifiers. AUC for ensemble has been obtained varying parameter γ^E from 0 to ∞ (in practice to maximum value for $\Lambda(\mathbf{x}_k^E)$ in samples of the test sets). AUC for individual classifiers has been computed by means of the trapezoidal integral of the 9 points (24) along with trivial ROC points $(0, 0)$ and $(1, 1)$. AUC for ensemble is higher than AUC obtained with individual classifiers in all databases, which shows the advantage of combining classifiers.

AUC gives an idea of the whole capability of the classifier to work at different tradeoffs for p_{FA} and p_M. Therefore, it is an appropriate figure of merit for a method designed to be able to work at any specified tradeoff. To evaluate the ability of the method to work at a specified tradeoff for errors of both classes,

Table 2. AUC for the ensemble and for the 9 working points (24) of the individual neural classifiers, along with trivial $(0, 0)$ and $(1, 1)$ ROC points.

Database	Individual classifiers	Ensemble
Yeast05679vs4	0.8286 (± 0.0609)	0.8618 (± 0.0546)
Ecoli067vs5	0.8935 (± 0.0602)	0.9226 (± 0.0712)
Glass2	0.7934 (± 0.1542)	0.9096 (± 0.0723)
Led7digit	0.9312 (± 0.0441)	0.9474 (± 0.0450)
Cleveland0vs4	0.8803 (± 0.1201)	0.9531 (± 0.0558)
Yeast4	0.8573 (± 0.0234)	0.8968 (± 0.0179)
Yeast5	0.9439 (± 0.0503)	0.9734 (± 0.0309)
Yeast6	0.8903 (± 0.0730)	0.9125 (± 0.0699)

Table 3. Average success probability $p_S = 1 - (p_{FA} + p_M)/2$ for best method in [5], individual neural classifier with $\alpha^{(j)} = 1/2$, and ensemble of 9 classifiers with $\alpha^E = 1/2$.

Database	Best in [5]	Individual	Ensemble
Yeast05679vs4	**0.8144**	0.7721 (± 0.0527)	0.7940 (± 0.0532)
Ecoli067vs5	0.8900	0.8736 (± 0.0539)	**0.8963** (± 0.0656)
Glass2	0.8045	0.7787 (± 0.1775)	**0.8675** (± 0.0917)
Led7digit	0.8880	0.8581 (± 0.0491)	**0.8911** (± 0.0838)
Cleveland0vs4	0.8280	0.8014 (± 0.1598)	**0.9164** (± 0.0712)
Yeast4	**0.8489**	0.7856 (± 0.0532)	0.8219 (± 0.0186)
Yeast5	**0.9661**	0.9343 (± 0.0548)	0.9506 (± 0.0481)
Yeast6	0.8678	0.8517 (± 0.0819)	**0.8771** (± 0.0712)

Table 3 compares the average probability of a successful classification of samples of both classes, measured as

$$p_S = 1 - \frac{p_{FA} + p_M}{2} \tag{31}$$

This means to assign the same importance to errors in both classes, which in the Bayesian risk corresponds to $\alpha = 1/2$. Therefore, $\alpha^E = 1/2$ will be used in the ensemble. Table also includes results obtained using a single individual neural classifier using $\alpha^{(j)} = 1/2$, and the best result provided by all the methods compared in [5]. This work compares performance obtained using ensembles of classifiers (10 to 40 classifiers) along with several preprocessing techniques used to balance data sets before constructing the ensemble. The proposed ensemble method provides better results than the individual neural classifier for all databases. Compared with the best method in [5], it provides better results in 5 of the 8 databases. We want to remark that the proposed method does not modify the data set before constructing the ensemble, to balance the number of samples of each class, as some methods in [5] do. The classifier is constructed using the original data.

5 Discussion

A new method, designed to classify imbalanced data, has been presented. The method combines several classifiers, each one designed to give a different relative importance to errors in majority and minority classes (p_{FA} and p_M) through parameters $\alpha^{(j)}$. This provides the ensemble with decisions for different working points in the ROC curve. Using these diverse decisions, the ensemble has the capability of working efficiently at different points of the ROC, allowing to select different compromise for both classes by varying parameter γ^E (or α^E).

The proposed method has shown competitive results in several imbalanced databases using a generic and simple setup, without validation nor an specific tuning for each database, and without data pre-processing to balance data sets. Lack of balance is handled with a proper choice for parameter α^E. This demonstrates its intrinsic potential to work with imbalanced data sets.

References

1. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification, 2nd edn. John Wiley & Sons (2001)
2. Widrow, B., Rumelhard, D.E., Lehr, M.A.: Neural networks: Applications in industry, business and science. Communications of the ACM **37**(3), 93–105 (1994)
3. Zhang, G.P.: Neural networks for classification: A survey. IEEE Transactions on Systems, Man, and Cybernetics **30**(4), 451–462 (2000)
4. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering **21**(9), 1263–1284 (2009)
5. Galar, M., Fernández, A., Barrenechea, E., Herrera, F.: EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. Pattern Recognition **46**, 3460–3471 (2013)
6. Galar, M., Fernández, A., Barrenechea, E., Bustince, H., Herrera, F.: A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews **42**(4), 463–484 (2012)
7. Kay, S.M.: Fundamentals of statistical signal processing: detection theory. Prentice-Hall Inc., Upper Saddle River (1998)
8. Scharf, L.S.: Statistical signal processing: detection, estimation, and time series analysis. Addison-Wesley (1991)
9. Van Trees, H.L.: Detection, Estimation, and Modulation Theory: Part I. John Wiley and Sons, New York (1968)
10. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters **27**(8), 861–874 (2006)
11. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition **30**(7), 1145–1159 (1997)
12. Cid-Sueiro, J., Arribas, J.I., Urbán-Muñoz, S., Figueiras-Vidal, A.R.: Cost functions to estimate a posteriori probabilities in multiclass problems. IEEE Transactions on Neural Networks **10**(3), 645–656 (1999)
13. Parzen, E.: On the estimation of a probability density function and the mode. Annals of Mathematical Statistics **33**, 1065–76 (1962)

14. Alcalá-Fdez, A., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F.: KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. Journal of Multiple-Valued Logic and Soft Computing **17**(2–3), 255–287 (2011)
15. Rumelhart, D.E., Hinton, G.E., Willians, R.J.: Learning representations by back-propagating errors. Nature (London) **323**, 533–536 (1986)
16. Widrow, B., Lehr, M.A.: 30 years of adaptive neural networks: perceptron, madaline and backpropagation. Proceedings of the IEEE **78**(9), 1415–1441 (1990)