HAL
open science

On multi-class learning through the minimization of the confusion matrix norm

Sokol Koço, Cécile Capponi

To cite this version:

Sokol Koço, Cécile Capponi. On multi-class learning through the minimization of the confusion matrix norm. 2013. hal-00801313v2

HAL Id: hal-00801313

https://hal.science/hal-00801313v2

Submitted on 1 Nov 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On multi-class learning through the minimization of the confusion matrix norm

Sokol Koço Cécile Capponi
Aix-Marseille Univ., LIF-QARMA, CNRS, UMR 7279, F-13013, Marseille, France
{*firstname.name*}*@lif.univ-mrs.fr*

November 1, 2013

Abstract

In imbalanced multi-class classification problems, the misclassification rate as an error measure may not be a relevant choice. Several methods have been developed where the performance measure retained richer information than the mere misclassification rate: misclassification costs, ROC-based information, etc. Following this idea of dealing with alternate measures of performance, we propose to address imbalanced classification problems by using a new measure to be optimized: the norm of the confusion matrix.

Indeed, recent results show that using the norm of the confusion matrix as an error measure can be quite interesting due to the fine-grain informations contained in the matrix, especially in the case of imbalanced classes. Our first contribution then consists in showing that optimizing criterion based on the confusion matrix gives rise to a common background for cost-sensitive methods aimed at dealing with imbalanced classes learning problems. As our second contribution, we propose an extension of a recent multi-class boosting method — namely AdaBoost.MM — to the imbalanced class problem, by greedily minimizing the empirical norm of the confusion matrix. A theoretical analysis of the properties of the proposed method is presented, while experimental results illustrate the behavior of the algorithm and show the relevancy of the approach compared to other methods.

Keywords: Multi-class Learning, Classification, Imbalanced Learning, Boosting, Confusion Matrix

1 Introdution

In multi-class classification, learning from imbalanced data concerns theory and algorithms that process a relevant learning task whenever data is not uniformly distributed among classes. When facing imbalanced classes, the classification accuracy is not a fair measure to be optimized (Fawcett, 2006). Accuracy may be artificially quite high in case of extremely imbalanced data: majority classes are favored, while minority classes are not recognized. Prediction is biased toward the classes with the highest priors. Such a bias gets stronger within the multi-class setting.

In the binary setting, learning from imbalanced data has been well-studied over the past years, leading to many algorithms and theoretical results (He & Garcia, 2009). It is mostly achieved by either resampling methods for rebalancing the data (e.g. Estabrooks et al. (2004)), and/or by using cost-sensitive methods (e.g. Ting (2000)), or with additional assumptions (e.g. active learning within kernel-based methods Bordes et al. (2005)).

However, learning from imbalanced data within a multi-class or multi-label setting is still an open research problem, which is sometimes adressed through the study of some specific measures of interest. Most of the time, generalizing the binary setting to the multi-class setting is based on the one-vs-all (or one-vs-one) usual trade-off (Abe, 2004). Recently, Zhou & Liu (2010) proposed a well-founded general rescaling approach of multi-class cost-sensitive learning, that pointed out the need of separating misclassification costs from class imbalance rates during the learning algorithm.

It is worth noticing that specific learning tasks other than classification have been addressed through the optimization of relevant measures within the multi-class imbalanced setting (Chapelle & Chang., 2011; Yue et al., 2007; Wang et al., 2012; Tang et al., 2011). Although related to accuracy, these measures are intended to better model what one would expect to be a a relevant performance measure in such a setting. Meanwhile, the correlations between these alternative measures and accuracy have been partly studied (Cortes & Mohri, 2004) without any theoretical result so far (He & Garcia, 2009).

Cost-sensitive methods are usually based on a cost matrix which embeds misclassification costs carrying various meanings. These methods weight the error measure according to class-based costs of each misclassification, computed from the confusion matrix results (Elkan, 2001). Indeed, the *confusion matrix* is one of the most informative performance measures a multi-class learning system can rely on. Among other information, it contains how much the classifier is accurate on one class, and the way it tends to mistake each class for other ones (confusion among classes).

In the binary classification setting, cost-sensitive approaches provide algorithms that somehow optimize a metric computed from the raw confusion matrix where the entries of a row sum up to the number of examples of the class corresponding to the row. Basically, a raw confusion matrix is a square matrix that represents the count of a classifier's class predictions with respect to the actual outcome on some labeled learning set. Computed from the raw matrix, the *probabilistic confusion matrix* (section 3) exhibits an interesting property: the entries of a row sum up to 1, independently from the actual number of examples of the class corresponding to the row. With such property, the confusion matrix constitutes a great equalizer tool that can be used to overcome the class-imbalance problem. Moreover, if one only considers the non-diagonal elements in the matrix, then summing over a row gets quite informing about how the corresponding class is correctly handled by the predictor at hand. This paper explores one way to capitalize on this property: we advocate that directly minimizing the norm of the confusion matrix is helpful for smoothing the accuracy among imbalanced classes, thus giving more importance to minority classes.

Based on recent works on the confusion matrix (Morvant et al., 2012; Ralaivola, 2012) and helped by a multi-class classification theoretical setting (Mukherjee & Schapire, 2011), the aim of this paper is to sketch up a computationally and theoretically consistent classification algorithm dubbed CoMBo (section 4) that is ensured to minimize the norm of the confusion matrix; while also minimizing the classification error as proven in section 3. Based on Adaboost, this algorithm greedily processes a sort-of regularization on imbalanced classes, in such a way that poorly represented classes are performed as well as majority classes within the overall learning process, independently from any prior misclassification cost. Section 5 summarizes the experimental performances of this algorithm, compared to Adaboost.MM (Mukherjee & Schapire, 2011) and other boosting-based related approaches. Sections 6 and 7 propose a discussion on the contributions of this paper and the future works.

2 General framework and notation

2.1 General notation

Matrices are denoted with bold capital letters like $\mathbf{C}$, while vectors are denoted by bold small letters like $\mathbf{x}$. The entry of the l-th row and the j-th column of $\mathbf{C}$ is denoted $\mathbf{C}(l,j)$, or simply $\mathbf{c}_{l,j}$. $\lambda_{max}(\mathbf{C})$ and $Tr(\mathbf{C})$ respectively correspond to the largest eigenvalue and the trace of $\mathbf{C}$. The *spectral* or *operator norm* $\|\mathbf{C}\|$ of $\mathbf{C}$ is defined as:

$$\|\mathbf{C}\| \stackrel{\text{def}}{=} \max_{\mathbf{v}\neq 0} \frac{\|\mathbf{C}\mathbf{v}\|_2}{\|\mathbf{v}\|_2} \stackrel{\text{def}}{=} \sqrt{\lambda_{max}(\mathbf{C}^*\mathbf{C})},$$

where $\|\cdot\|_2$ is the Euclidian norm and $\mathbf{C}^*$ is the conjugate transpose of $\mathbf{C}$. Let $\mathbf{A}$ and $\mathbf{B}$ be two matrices, then $\mathbf{AB}$ and $\mathbf{A}\cdot\mathbf{B}$ respectively refer to the inner product and the Frobenius inner product of $\mathbf{A}$ and $\mathbf{B}$.

The indicator function is denoted by $\mathbb{I}$; K is the number of classes, m the number of examples and m_y is the number of examples of class y, where $y \in \{1,...,K\}$. $(\mathbf{x}_i, y_i), \mathbf{x}_i$ or simply i are interchangeably used to denote the ith training example.

2.2 Multi-class boosting framework

In this paper we use the boosting framework for multi-class classification introduced in Mukherjee & Schapire (2011), and more precisely the one defined for AdaBoost.MM. Algorithms based on the AdaBoost family maintain a distribution over the training samples in order to identify hard-to-classify examples: the greater the weight of an example, the greater the effort devoted to classify these data correctly. In the considered setting, the distribution over the training examples is replaced by a cost matrix. Let $S = \{(\mathbf{x}_i, y_{,i})\}_{i=1}^m$ be a training sample, where $\mathbf{x}_i \in X$ and $y_i \in \{1,...,K\}$. The cost matrix $\mathbf{D} \in \mathbb{R}^{m\times K}$ is constructed so that for a given example $(\mathbf{x}_i, y_i)$, $\forall l \neq y_i$: $\mathbf{D}(i, y_i) \leq \mathbf{D}(i, l)$, where i is the row of $\mathbf{D}$ corresponding to $(\mathbf{x}_i, y_i)$.

This cost matrix is a particular case of cost matrices used in cost sensitive methods (for example, Sun et al. (2006)), where classification costs are given for each example and each class. However, contrary to those methods, the matrix is not given prior to the learning process, but it is updated after each

iteration of AdaBoost.MM so that the misclassification cost reflects the difficulty of correctly classifying an example. That is, the costs are increased for examples that are hard to classify, and they are decreased for easier ones.

In the case of AdaBoost.MM, the cost matrix $\mathbf{D}$ at iteration T, is defined as follows:

$$\mathbf{D}_T(i,l) \stackrel{\text{def}}{=} \begin{cases} \exp(f_T(i,l) - f_T(i,y_i)) & \text{if } l \neq y_i \\ -\sum\limits_{j \neq y_i} \exp(f_T(i,j) - f_T(i,y_i)) & \text{otherwise,} \end{cases}$$

where $f_T(i,l)$ is the score function computed as: $f_T(i,l) = \sum_{t=1}^{T} \alpha_t \mathbb{I}(h_t(i) = l)$.

At each iteration t, AdaBoost.MM selects the classifier h and its weight α that minimize the exponential loss:

$$h_t, \alpha_t = \underset{h,\alpha}{\operatorname{argmin}} \sum_{i=1}^{m} \sum_{j \neq y_i} e^{f_t(i,j) - f_t(i,y_i)}, \text{ where } f_t(i,j) = \sum_{s=1}^{t-1} \alpha_s \mathbb{I}(h_s(i) = j) + \alpha \mathbb{I}(h(i) = j).$$

Finally, the output hypothesis of AdaBoost.MM is a simple majority vote:

$$H(x) = \underset{l=1\ldots K}{\operatorname{argmax}}\ f_T(i,l).$$

3 Boosting the confusion matrix

3.1 The confusion matrix as an error measure

When building predictors, most Empirical Risk Minimization based methods optimize loss functions defined over a training sample and depending on the number of misclassified data. This reveals to be a poor strategy when tackling problems with class-imbalance, since minimizing the estimated error may result in overfitting the majority classes.

A common tool used for estimating the goodness of a classifier in the imbalanced classes scenario is the confusion matrix. We give here the probabilistic definitions of the *true confusion matrix* and the *empirical confusion matrix*.

Definition 1 (True and empirical confusion matrices)**.** *The true confusion matrix* $\mathbf{A} = (a_{l,j}) \in \mathbb{R}^{K \times K}$ *of a classifier* $h : X \leftrightarrow \{1, \ldots, K\}$ *over a distribution* $\mathcal{D}$ *is defined as follows:*

$$\forall l, j \in \{1, ..., K\}, \mathbf{a}_{l,j} \stackrel{def}{=} \mathbb{E}_{\mathbf{x}|y=l} \mathbb{I}\big(h(\mathbf{x}) = j\big) = \mathbb{P}_{(\mathbf{x},y)\sim\mathfrak{D}}(h(\mathbf{x}) = j | y = l).$$

For a given classifier h *and a sample* $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m} \sim \mathcal{D}$*, the empirical confusion matrix* $\mathbf{A}_S = (\hat{a}_{l,j}) \in \mathbb{R}^{K \times K}$ *of* h *is defined as :*

$$\forall l, j \in \{1, ..., K\},\ \hat{\mathbf{a}}_{l,j} \stackrel{def}{=} \sum_{i=1}^{m} \frac{1}{m_l} \mathbb{I}(h(\mathbf{x}_i) = j) \mathbb{I}(y_i = l).$$

An interesting property of these matrices is that the entries of a row sum up to 1, independently from the number of examples contained in the corresponding class. Moreover, the diagonal entries represent the capability of the classifier h to recognize the different classes, while the non-diagonal entries represent the mistakes of the classifier.

In this paper, and in the following sections, we propose a framework which makes use of the confusion matrix as an error measure in order to learn predictors for the imbalanced classes setting. As such, we need to redefine the confusion matrices, so that only the mistakes of the classifier are taken into consideration. This is done by keeping the non-diagonal entries and zeroing the diagonal ones:

Definition 2 (Error-focused true and empirical confusion matrices)**.** *For a given classifier* h*, we define the empirical and true confusion matrices of* h *by respectively* $\mathbf{C}_S = (\hat{\mathbf{c}}_{l,j})_{1 \leq l,j \leq K}$ *and* $\mathbf{C} = (\mathbf{c}_{l,j})_{1 \leq l,j \leq K}$ *such that for all* (l, j)*:*

$$\hat{\mathbf{c}}_{l,j} \stackrel{def}{=} \begin{cases} 0 & \textit{if } l = j \\ \hat{\mathbf{a}}_{l,j} & \textit{otherwise,} \end{cases} \quad \mathbf{c}_{l,j} \stackrel{def}{=} \begin{cases} 0 & \textit{if } l = j \\ \mathbf{a}_{l,j} & \textit{otherwise.} \end{cases} \tag{1}$$

Let $\mathbf{p} = [P(y=1), ..., P(y=K)]$ be the vector of class priors distribution, then we have:

$$R(h) \stackrel{\text{def}}{=} P_{(x,y)\sim\mathcal{D}}(h(x) \neq y) = \|\mathbf{pC}\|_1, \tag{2}$$

where $\|\cdot\|_1$ is the $l1$-norm. This result means that it is possible to retrieve the true error rate of h from its confusion matrix. Using this result and the equivalence between norms in finite dimensions, we obtain equation (3) between the operator norm and the true risk:

$$R(h) \leq \sqrt{K}\|\mathbf{C}\| \tag{3}$$

Equations (2) and (3) together imply that minimizing the norm of the confusion matrix can be a good strategy for dealing with imbalanced classes, while at the same time it would ensure a small risk. Our aim is thus to find a classifier $\hat{h}$ that verifies the following criterion:

$$\hat{h} = \underset{h\in\mathcal{H}}{\operatorname{argmin}}\|\mathbf{C}\| \tag{4}$$

3.2 Bounding the confusion matrix

The result given in equation (4) minimizes the operator norm of the true confusion matrix, but it is difficult to use in a practical case, since the underlying distribution $\mathcal{D}$ is unknown. A popular way to overcome this difficulty is to use the empirical estimation of the confusion matrix. Theorem 1 in Ralaivola (2012) gives the relation between the *true* and the *estimated norm* of the confusion matrix. [1] We recall here a particular formulation of this theorem applied to the supervised setting, where the considered loss is the indicator function $\mathbb{I}$.

Corollary 1. *For any $\delta \in (0;1]$, it holds with probability $1-\delta$ over a sample $S_{(\mathbf{x},y)\sim\mathcal{D}}$ that:*

$$\|\mathbf{C}\| \leq \|\mathbf{C}_S\| + \sqrt{2K\sum_{k=1}^{K}\frac{1}{m_k}\log\frac{K}{\delta}},$$

where $\mathbf{C}_S$ is the empirical confusion matrix computed for a classifier h over S.

The corollary suggests that our goal may boil down to minimizing the empirical norm of the confusion matrix, which is fairly similar to other optimization problems, where empirical error measures are considered. Unfortunately, due to the nature of the considered confusion matrix, finding an analytical expression for the norm reveals to be quite challenging. This is why we propose to optimize an upper bound of $\|\mathbf{C}_S\|^2$ instead.

$$\|\mathbf{C}_S\|^2 = \lambda_{max}(\mathbf{C}_S^*\mathbf{C}_S) \leq Tr(\mathbf{C}_S^*\mathbf{C}_S) \tag{5}$$

The matrix $\mathbf{C}_S^*\mathbf{C}_S$ is positive semi-definite, meaning that all its eigenvalues are positive. Thus we choose to upper-bound the norm of the confusion matrix by the trace of $\mathbf{C}_S^*\mathbf{C}_S$ as in equation (5). We can now focus on the value of $Tr(\mathbf{C}_S^*\mathbf{C}_S)$.

$$Tr(\mathbf{C}_S^*\mathbf{C}_S) = \sum_{l=1}^{K}\mathbf{C}_S^*\mathbf{C}_S(l,l) \leq \sum_{l=1}^{K}\sum_{j\neq l}\hat{\mathbf{c}}_{l,j} = \sum_{l=1}^{K}\sum_{j\neq l}\frac{1}{m_l}\sum_{i=1}^{m}\mathbb{I}(y_i=l)\mathbb{I}(H(i)=j) \tag{6}$$

Most of the previous equalities are simple rewritings of the involved terms, while the inequality comes from the fact that the entries of the confusion matrix $\mathbf{C}_S$ are at most 1. We now have an upper bound on the norm that depends only on the entries of the confusion matrix (equation (6)).

The drawback of this bound is the presence of the indicator function, which is not optimization friendly. One way to handle this difficulty is to replace the indicator function with loss functions defined over the two classes of indices l and j in the bound.

For a given classifier h and a sample S, let $\ell_{l,j}(h,\mathbf{x})$ be the loss of h choosing class j instead of class l for the example $\mathbf{x}$, such that $\forall(\mathbf{x},y)\in S, 1\leq l,j\leq K$ and $0\leq\mathbb{I}(h(\mathbf{x})\neq y)\leq\ell_{l,j}(h,\mathbf{x})$. The bound on the confusion matrix can be now expressed using loss functions:

$$Tr(\mathbf{C}_S^*\mathbf{C}_S) \leq \sum_{l=1}^{K}\sum_{j\neq l}\frac{1}{m_l}\sum_{i=1}^{m}\mathbb{I}(y_i=l)\ell_{l,j}(h,\mathbf{x}_i) = \sum_{i=1}^{m}\sum_{j\neq y_i}\frac{1}{m_{y_i}}\ell_{y_i,j}(h,\mathbf{x}_i). \tag{7}$$

[1] In the original paper, the theorem is applied to the online setting, but it holds also for the batch setting.

The resulting bound is a sum over two penalization terms for all the learning examples: the first term is $\frac{1}{m_{y_i}}$ and the second one represents the newly introduced loss function. The first term is often used in the imbalanced classes problems in order to simulate the resampling effect over the training sample. Indeed, by multiplying the weight of an example with the inverse of the number of examples having the same class, the distribution over the classes becomes more uniform, thus promoting all the classes the same way. Resampling over the examples achieves the same effect, since it aims to retain the same number of examples for each class.

The second term is common in the imbalanced classes setting, where cost-sensitive methods — either per-class based or per-couple of classes based — have been developed. In these methods this term depends on some misclassification cost computed, or simply given, *prior* to the learning phase.

Based on these observations, it follows that the last term of the bound given in equation 7 is quite similar to usual terms in optimization problems for the imbalanced classes setting. As such it can be seen as a common base for most imbalanced classes learning problems, since it encompasses both resampling and penalization, as advocated in Zhou & Liu (2010). Motivated by these results, and as an intermediate conclusion, we strongly think that minimizing the operator norm of the confusion matrix defined as in equation 1, yields a good strategy for the imbalanced classes problem.

3.3 Towards a boosting method

Let $\mathcal{H}$ be an ensemble of classifiers and $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ a learning sample. Our goal is to find a classifier $\hat{H}$ which can be written as a weighted combination of all the classifiers of $\mathcal{H}$ and at the same time, which minimizes the norm of the confusion matrix. That is:

$$\hat{H} = \operatorname*{argmin}_{H \in \mathcal{H}} \|\mathbf{C}_S\|^2, \tag{8}$$

where $H(\cdot) = \operatorname{argmax}_{l \in \{1,\cdots,K\}} \sum_{h \in \mathcal{H}} \alpha_h \mathbb{I}(h(\cdot) = l)$, and $\alpha_h \geq 0, \forall h$.

In order to make use of the bound given in equation (7), we need to choose loss functions adapted to our problem. We start off by defining score functions for each example i and each class l as follows:

$$f_H(i, l) = \sum_{h \in \mathcal{H}} \alpha_h \mathbb{I}(h(\mathbf{x}_i) = l), \forall i \in \{1, \cdots, m\}, l \in \{1, \cdots, K\}. \tag{9}$$

These simple functions measure the importance that the classifier H gives to class l for example i. Since the prediction rule is based on the argmax (eq. (8)), H returns the class with the highest score. If H correctly classifies an example, then the score given to its true class is higher than the score given to any of the other classes. Hence, let the loss functions be the difference between the score given to a class and the one given to the true class:

$$\ell_{y_i,j}(H, \mathbf{x}_i) = f_H(i, j) - f_H(i, y_i), \forall j \neq y_i. \tag{10}$$

If an example is correctly classified then the loss is negative, while if it is misclassified, then the loss is positive for at least one of the classes. This observation reveals to be the main downside of these losses, since it implies that the requirements for the bound in equation (7) might not be met. This is why we propose to consider the exponential losses instead, and to rewrite our goal (8) as:

$$\hat{H} = \operatorname*{argmin}_{H \in \mathcal{H}} \sum_{i=1}^{m} \sum_{j \neq y_i} \frac{1}{m_{y_i}} \exp\Big(f_H(i, j) - f_H(i, y_i)\Big). \tag{11}$$

Finding the optimal solution for the last equation can be quite challenging, since it depends on the size of $\mathcal{H}$ and the weights. A popular approach when dealing with ensembles of classifiers is to use an iterative greedy method based on the boosting framework. The goal of such methods is to consider, at each iteration, the best classifier $h \in \mathcal{H}$ — and its weight α — that minimizes the loss. The main advantage of boosting methods resides in the fact that the selected classifier h, and generally all the classifiers in $\mathcal{H}$, only need to perform slightly better than random guessing, a notion that is formalized later on. They are also known as weak classifiers.

Suppose that at iteration t, the classifiers $h_1, \ldots, h_t \in \mathcal{H}$ have been chosen in order to minimize the loss of $H_t(\cdot) = \operatorname{argmax}_{l \in \{1,\cdots,K\}} \sum_{s=1\cdots t} \alpha_s \mathbb{I}(h_s(\cdot) = l)$. In the following iteration, the chosen classifier would be the one verifying:

$$h_{t+1}, \alpha_{t+1} = \operatorname*{argmin}_{h \in \mathcal{H}, \alpha} \sum_{i=1}^{m} \sum_{j \neq y_i} \frac{1}{m_{y_i}} \exp\Big(f_{t+1}(i, j) - f_{t+1}(i, y_i)\Big), \tag{12}$$

where $f_{t+1}(i,j) = \sum_{s=1\cdots t} \alpha_s \mathbb{I}(h_s(\mathbf{x}_i) = j) + \alpha \mathbb{I}(h(\mathbf{x}_i) = j)$.

This last optimization problem is quite similar to the one defined for AdaBoost.MM by Mukherjee & Schapire (2011). As such, the method proposed in this paper is an extension of AdaBoost.MM to the imbalanced classes setting.

In order to take advantage of the boosting framework recalled in section 2, we need to define a cost matrix $\mathbf{D}$ so that $\forall i$, and $\forall l \neq y_i, \mathbf{D}(i,l) \leq \mathbf{D}(i,y_i)$. Due to the similarity of our minimization problem (12) and AdaBoost.MM's, the most straightforward choice for $\mathbf{D}$ is the following :

$$\mathbf{D}_t(i,l) \stackrel{\text{def}}{=} \begin{cases} \frac{1}{m_{y_i}} \exp(f_t(i,l) - f_t(i,y_i)) & \text{if } l \neq y_i \\ -\sum\limits_{j \neq y_i} \frac{1}{m_{y_i}} \exp(f_t(i,j) - f_t(i,y_i)) & \text{otherwise.} \end{cases} \tag{13}$$

4 Algorithm and theoretical properties

In the previous section, we showed why it is interesting to consider a boosting-based method for minimizing the norm of the empirical confusion matrix. This section introduces CoMBo, the actual algorithm that performs this minimization, where the loss decreases at each step of the algorithm.

4.1 The Confusion Matrix Boosting Algorithm

The pseudo-code of the proposed method named CoMBo is given in Algorithm 1. The inputs : a training sample S, the total number of iterations T and a weak learner $\mathcal{W}$. During the initialization step, the score functions f are set to zero and the cost matrix $\mathbf{D}$ is initialized accordingly.

The training phase consists of two steps: the weak learner $\mathcal{W}$ is used in order to build the weak classifiers, and the predictions of h_t are used to update the cost matrix $\mathbf{D}_t$. At each round t, $\mathcal{W}$ takes as

Algorithm 1 CoMBo : Confusion Matrix BOosting

Given

- S $= \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$ where $\mathbf{x}_i \in X$, $y_i \in \{1, \cdots, K\}$
- T the number of iterations, $\mathcal{W}$ a weak learner
- $\forall i \in \{1, \ldots, m\}$, $\forall l \in \{1, \ldots, K\}$ $f_1(i,l) = 0$
- $\mathbf{D}_1(i,l) = \begin{cases} \frac{1}{m_{y_i}} & \text{if } y_i \neq l \\ \frac{-(K-1)}{m_{y_i}} & \text{if } y_i = l \end{cases}$

for $t = 1$ **to** T **do**

Use $\mathcal{W}$ to learn h_t with edge δ_t on $\mathbf{D}_t$, and $\alpha_t = \frac{1}{2} \ln \frac{1+\delta_t}{1-\delta_t}$

where :

$$\delta_t = \frac{-\sum_{i=1}^m \mathbf{D}_t(i, h_t(\mathbf{x}_i))}{\sum_{i=1}^m \sum_{l \neq y_i} \mathbf{D}_t(i,l)}$$

Update $\mathbf{D}$:

$$\mathbf{D}_{t+1}(i,l) = \begin{cases} \frac{1}{m_{y_i}} \exp(f_{t+1}(i,l) - f_{t+1}(i,y_i)) & \text{if } l \neq y_i \\ -\frac{1}{m_{y_i}} \sum\limits_{j \neq y_i}^{k} \exp(f_{t+1}(i,j) - f_{t+1}(i,y_i)) & \text{if } l = y_i \end{cases}$$

where $f_{t+1}(i,l) = \sum\limits_{z=1}^{t} \mathbb{I}[h_z(i) = l]\alpha_z$

end for

Output final hypothesis :

$$H(\mathbf{x}) = \underset{l \in 1,\ldots,k}{\text{argmax}} f_T(\mathbf{x}, l), \text{ where } f_T(\mathbf{x}, l) = \sum_{t=1}^{T} \mathbb{I}[h_t(\mathbf{x}) = l]\alpha_t$$

input the cost matrix $\mathbf{D}_t$ and returns a weak classifier h_t. The cost matrix is then used to compute the weight α_t for h_t, which can be seen as the importance given to h_t. α_t depends on the edge δ_t obtained by h_t over the cost matrix $\mathbf{D}_t$. For boosting methods based on AdaBoost, the edge measures the difference of performances of the classifier and of random guessing (Mukherjee & Schapire, 2011). The underlying idea is that the better h_t performs over $\mathbf{D}_t$, the greater the edge δ_t and α_t.

The update rule for the cost matrix is designed so that the misclassification cost is increased for the examples that are misclassified by h_t, while it is decreased for the correctly classified ones. This forces the weak learner $\mathcal{W}$ to focus on the most difficult examples. Then, using the term $1/m_{y_i}$ in the update rule, allows the misclassification cost of an example to depend not only on the ability to correctly classify a hard example, as in AdaBoost.MM, but also on the number of examples of S having the same class y_i.

The output hypothesis is a simple weighted majority vote over the whole set of weak classifiers. So, for a given example, the final prediction is the class that obtains the highest score.

4.2 Bounding the loss

Let us recall the minimal weak learning condition as given by Mukherjee & Schapire (2011).

Definition 3. *Let D^{eor} be the space of all cost matrices $\mathbf{D}$ that put the least cost on the correct label, that is $\forall(\mathbf{x}_i, y_i), l, \mathbf{D}(i, y_i) \leq \mathbf{D}(i, l)$. Let B^{eor}_γ be the space of baselines $\mathbf{B}$ which are γ more likely to predict the correct label for every example $(\mathbf{x}_i, y_i)$, i.e. $\forall l \neq y_i, \mathbf{B}(i, y_i) \geq \mathbf{B}(i, l) + \gamma$. Then, the minimal weak learning condition is given by:*

$$\forall \mathbf{D} \in D^{eor}, \exists h \in \mathcal{H} : \mathbf{D} \cdot \mathbf{1}_h \leq \max_{\mathbf{B} \in B^{eor}} \mathbf{D} \cdot \mathbf{B}, \tag{14}$$

where $\mathcal{H}$ is a classifier space, and $\mathbf{1}_h$ is the matrix defined as $\mathbf{1}_h(i, l) = \mathbb{I}[h(i) = l]$.

In the rest of this paper, we consider baselines that are close to uniform random guessing. Noted $\mathbf{U}_\gamma$, these baselines have weights $(1-\gamma)/k$ on incorrect labels and $(1-\gamma)/k + \gamma$ on correct ones. The weak learning condition is given by :

$$\mathbf{D} \cdot \mathbf{1}_h \leq \mathbf{D} \cdot \mathbf{U}_\gamma \tag{15}$$

All of the weak classifiers returned by $\mathcal{W}$ during the training phase are assumed to verify that weak learner condition.

Lemma 1 shows that the general loss decreases with each iteration, provided that the weak classifier h_t satisfies the weak learning condition (15). This result and its proof are similar to the ones given for AdaBoost.MM.

Lemma 1. *Suppose the cost matrix $\mathbf{D}_t$ is chosen as in the algorithm 1, and the returned classifier h_t satisfies the edge condition for the baseline $\mathbf{U}_{\delta_t}$ and cost matrix $\mathbf{D}_t$,* i.e. $\mathbf{D}_t \cdot \mathbf{1}_{h_t} \leq \mathbf{D}_t \cdot \mathbf{U}_{\delta_t}$.

Then choosing a weight $\alpha_t > 0$ for h_t makes the loss at round t at most a factor

$$1 - \frac{1}{2}\left(e^{\alpha_t} - e^{-\alpha_t}\right)\delta_t + \frac{1}{2}\left(e^{\alpha_t} + e^{-\alpha_t} - 2\right)$$

of the loss before choosing α_t, where δ_t is the edge of h_t.

Proof. For readability reasons, let us note L_t and $L_t(i)$ the following losses:

$$L_t = \sum_{i=1}^{m} L_t(i), \text{ where } L_t(i) = \sum_{l \neq y_i} \frac{1}{m_{y_i}} \exp(f_t(i, l) - f_t(i, y_i))$$

The weak classifier h_t returned by $\mathcal{W}$ satisfies the edge condition, that is:

$$\mathbf{D_t} \cdot \mathbf{1}_{h_t} \leq \mathbf{D_t} \cdot \mathbf{U}_{\delta_t}, \tag{16}$$

with δ_t being the edge of h_t on $\mathbf{D_t}$.

Denote S_+ (resp. S_-) the set of examples of S correctly classified (resp. misclassified) by h_t. Using the definitions of $\mathbf{D_t}$, $\mathbf{1}_{h_t}$ and $\mathbf{U}_{\delta_t}$, the classification costs of h_t and $\mathbf{U}_{\delta_t}$ are given by:

$$\begin{aligned}
\mathbf{D}_t \cdot \mathbf{1}_{h_t} &= -\sum_{i \in S_+} L_{t-1}(i) + \sum_{i \in S_-} \tfrac{1}{m_{y_i}} \exp\left(f_{t-1}(i, h_t(i)) - f_{t-1}(i, y_i)\right) = -A^t_+ + A^t_- \\
\mathbf{D}_t \cdot \mathbf{U}_{\delta_t} &= -\delta_t \textstyle\sum_{i=1}^{m} L_{t-1}(i) = -\delta_t L_{t-1}
\end{aligned}$$

Inserting these two costs in (16), we have :

$$A_+^t - A_-^t \geq \delta_t L_{t-1}. \tag{17}$$

Taking a closer look at the drop of the loss after choosing h_t and α_t, we have:

$$\begin{aligned} L_{t-1} - L_t &= \sum_{i \in S_+} L_{t-1}(i)(1 - e^{-\alpha_t}) + \sum_{i \in S_-} \frac{1}{m_{y_i}} \exp(\Delta f_{t-1}(i, h_t, y_i))(1 - e^{\alpha_t}) \\ &= (1 - e^{-\alpha_t})A_+^t - (e^{\alpha_t} - 1)A_-^t \\ &= \left(\frac{e^{\alpha_t} - e^{-\alpha_t}}{2}\right)(A_+^t - A_-^t) - \left(\frac{e^{\alpha_t} + e^{-\alpha_t} - 2}{2}\right)(A_+^t + A_-^t) \end{aligned}$$

where $\Delta f_{t-1}(i, h_t, y_i)) = f_{t-1}(i, h_t(i)) - f_{t-1}(i, y_i)$.

The result in (17) gives a lower bound for $A_+^t - A_-^t$, while $A_+^t + A_-^t$ is upper-bounded by L_{t-1}. Hence,

$$L_{t-1} - L_t = \sum_{i \in S_+} L_{t-1}(i)(1 - e^{-\alpha_t}) \geq (\frac{e^{\alpha_t} - e^{-\alpha_t}}{2})\delta_t L_{t-1} - (\frac{e^{\alpha_t} + e^{-\alpha_t} - 2}{2})L_{t-1}.$$

Therefore, the result of the lemma:

$$L_t \leq \left(1 - \frac{e^{\alpha_t} - e^{-\alpha_t}}{2}\delta_t + \frac{e^{\alpha_t} + e^{-\alpha_t} - 2}{2}\right) L_{t-1} = \left(\frac{((1-\delta_t)}{2}e^{\alpha_t} + \frac{(1+\delta_t)}{2}e^{-\alpha_t})\right) L_{t-1}. \tag{18}$$

□

The expression of the loss drop given in Lemma 1 can be further simplified. Indeed, if we choose the value of α_t as given in the pseudo-code of Algorithm 1, then the loss drop is simply equal to $\sqrt{1 - \delta_t^2}$. Since the value of δ_t^2 is always positive, $\sqrt{1 - \delta_t^2}$ is smaller than 1, thus the loss L_t is always smaller than L_{t-1}. The following theorem summarizes this result.

Theorem 1. *Let $\delta_1, \cdots, \delta_T$ be the edges of the classifiers $h_1, \cdots, h_T$ returned by $\mathcal{W}$ at each round of the learning phase. Then the error after T rounds is $K(K-1)\prod_{t=1}^T \sqrt{1 - \delta_t^2} \leq K(K-1)\exp\left\{-(1/2)\sum_{t=1}^T \delta_t^2\right\}$.*

Moreover, if there exists a γ so that $\forall t, \delta_t \geq \gamma$, then the error after T rounds is exponentially small, $K(K-1)e^{-T\gamma^2/2}$.

5 Experimental results

5.1 Datasets and experimental setup

In order to compare CoMBo with other approaches, it was run on 9 imbalanced multi-class classification datasets. They are all from the UCI Machine Learning Repository (Frank & Asuncion, 2010). They exhibit various degrees of imbalance, as well as various numbers of instances and attributes. Table 1 summarizes information about these datasets.

Dataset	# classes	# examples	distribution of classes	Imb. ratio
New-Thyroid	3	215	150/35/30	5.00
Balance	3	625	49/288/288	5.88
Car	4	1728	1210/384/69/65	18.62
Connect	3	67557	44473/6449/16635	6.90
Nursery-s	4	12958	4320/328/4266/4044	13.17
Glass	6	214	70/76/17/13/9/29	8.44
E.coli	5	327	143/77/52/35/20	7.15
Yeast	10	1484	463/429/244/163/51/44/35/30/20/5	92.60
Satimage	6	6435	1533/703/1358/626/707/1508	2.45

Table 1: Class distributions of considered UCI datasets (the last column reports the ratio between the # of instances of the majority class and the # of instances of the minority class (imbalance ratio).)

For each dataset, we performed 10 runs of 5-fold cross-validation of CoMBo and Adaboost.MM with a low-size decision tree as a base learner ($\mathcal{W}$) and $T = 200$, and we averaged the results. Two kinds of results are reported: advanced accuracy performances are first compiled (MAUC, Gmean), then the behavior of CoMBo is compared to AdaBoost.MM with a deeper insight on the norm of the confusion matrix.

G-mean	CoMBo	AdaBoost.NC	SmoteBoost
Car	**0.967** $\pm$ 0.023	0.924 $\pm$ 0.024	0.944 $\pm$ 0.031
Balance	**0.675** $\pm$ 0.088	0.321 $\pm$ 0.173	0.000 $\pm$ 0.000
New-Thyroid	0.914 $\pm$ 0.081	0.927 $\pm$ 0.056	0.940 $\pm$ 0.057
Nursery-s	**1.000** $\pm$ 0.001	0.967 $\pm$ 0.006	0.996 $\pm$ 0.003
Ecoli	0.784 $\pm$ 0.066	0.790 $\pm$ 0.062	0.803 $\pm$ 0.059
Glass	0.431 $\pm$ 0.0.375	0.578 $\pm$ 0.249	0.561 $\pm$ 0.343
Satimage	†0.825 $\pm$ 0.012	0.881 $\pm$ 0.009	**0.898** $\pm$ 0.010
Yeast	0.107 $\pm$ 0.216	0.237 $\pm$ 0.270	0.140 $\pm$ 0.240
MAUC	CoMBo	AdaBoost.NC	SmoteBoost
Car	0.993 $\pm$ 0.003	0.982 $\pm$ 0.005	**0.997** $\pm$ **0.000**
Balance	**0.884** $\pm$ 0.032	0.704 $\pm$ 0.037	0.703 $\pm$ 0.027
New-Thyroid	**0.996** $\pm$ 0.010	0.983 $\pm$ 0.013	0.988 $\pm$ 0.003
Nursery-s	**1.000** $\pm$ 0.000	0.998 $\pm$ 0.000	0.999 $\pm$ 0.000
Ecoli	0.961 $\pm$ 0.015	0.957 $\pm$ 0.002	0.963 $\pm$ 0.004
Glass	**0.947** $\pm$ 0.027	0.881 $\pm$ 0.009	0.925 $\pm$ 0.009
Satimage	†0.976 $\pm$ 0.003	0.990 $\pm$ 0.000	**0.992** $\pm$ 0.000
Yeast	0.861 $\pm$ 0.025	0.857 $\pm$ 0.004	0.847 $\pm$ 0.003

Table 2: Comparison with algorithms addressing classification within imbalanced datasets. Means and standard deviations are given over 10 runs of 5-fold cross-validations. Except for CoMBo, the results are the *best* retrieved from Wang & Yao (2012), which analysed four algorithms. Results in boldface indicate a significantly best measure of the algorithm over all the others, according to the sudent T-test with a confidence of 95%; the † attached to the result on `Satimage` indicates that CoMBo is significantly worse than all other reported methods.

5.2 Performance results

Usual performance measures of multi-class classification include MAUC (an extension of the AUC to multi-class problems, (Hand & Till, 2001)), and G-mean (the geometric mean of recall values over all classes (Sun et al., 2006)).

Table 2 reports these measures for CoMBo, as well as results from two boosting-based algorithms relying on resampling: AdaBoost.NC with oversampling (Wang & Yao, 2012), and SmoteBoost (Chawla et al., 2003). These algorithms were chosen for comparison because they showed the best results on the considered datasets (Wang & Yao, 2012).

Looking at results in Table 2, CoMBo is quite promising w.r.t. current literature, without any tuning nor trying to optimize the reported measures. Only the best results of AdaBoost.NC and SmoteBoost from Wang & Yao (2012) are reported. Except for the dataset `Satimage` (the less imbalanced dataset), CoMBo challenges the other boosting-based approaches. The advantage of preferring CoMBo over the reported methods seems to be related to the imbalance ratio: the higher this ratio is, the more CoMBo makes the difference, which is consistent with the aim of CoMBo. It is worth noticing that CoMBo does not need any prior initialisation of the misclassification cost matrix: it computes and adjusts the cost matrix along the learning stages, based on the confusion matrix norm computed at each step. In other words, CoMBo gets these good results without any other parameters than the weak algorithm (actually a 2 to 3-depth decision tree), and the number T of rounds.

5.3 Improvements from Adaboost.MM

One may think that the good results of CoMBo are partly due to the fact that it is based on the theory underlying Adaboost.MM. Therefore, it is worth exploring the way CoMBo differs from Adaboost.MM in the processing of imbalanced datasets through the norm of the confusion matrix. Results are presented Table 3: the confusion matrix norms are reported, together with the accuracies, the MAUC and the G-mean.

These results confirm the good results of CoMBo on imbalanced datasets as evidenced by the low values of the norms. They illustrate the impact of minimizing the norm of the confusion matrix. Let us note that the accuracy with CoMBO tends to be a bit worse than the one of Adaboost.MM. Meanwhile, as expected, the norm of the confusion matrix is always smaller with CoMBo. The performances of the

dataset	CoMBo $\|\mathbf{C}\|$	AdaMM $\|\mathbf{C}\|$	CoMBo accuracy	AdaMM accuracy
Balance	$\mathbf{0.460} \pm 0.097$	0.559 ± 0.112	0.856 ± 0.039	0.875 ± 0.032
Car	0.082 ± 0.064	0.116 ± 0.080	0.974 ± 0.009	$\mathbf{0.977} \pm 0.012$
Connect	$\mathbf{0.308} \pm 0.015$	0.670 ± 0.040	0.728 ± 0.015	$\mathbf{0.805} \pm 0.009$
New-Thyroid	0.194 ± 0.158	0.186 ± 0.157	0.949 ± 0.033	0.952 ± 0.033
Nursery-s	0.002 ± 0.004	0.003 ± 0.006	1.000 ± 0.000	1.000 ± 0.001
Yeast	$\mathbf{0.815} \pm 0.150$	1.101 ± 0.194	0.572 ± 0.025	0.577 ± 0.025
	G-mean	G-mean	MAUC	MAUC
Balance	$\mathbf{0.675} \pm 0.088$	0.566 ± 0.190	$\mathbf{0.884} \pm 0.032$	0.855 ± 0.039
Car	0.967 ± 0.023	0.954 ± 0.034	$\mathbf{0.993} \pm 0.003$	0.988 ± 0.004
Connect	$\mathbf{0.703} \pm 0.013$	0.497 ± 0.035	$\mathbf{0.863} \pm 0.008$	0.852 ± 0.009
New-Thyroid	0.914 ± 0.081	0.915 ± 0.075	0.996 ± 0.010	0.996 ± 0.005
Nursery-s	$\mathbf{1.000} \pm 0.001$	0.999 ± 0.002	1.000 ± 0.000	1.000 ± 0.000
Yeast	0.107 ± 0.216	0.000 ± 0.000	$\mathbf{0.861} \pm 0.025$	0.847 ± 0.003

Table 3: Adaboost.MM vs. CoMBo. In non-reported datasets, results of both algorithms are equivalent). Means and standard deviations are given over 10 runs of 5-fold cross-validations. Results in boldface indicate a significantly best measure according to the sudent T-test with a confidence of 95%.

classifier trained with CoMBo is smoothed throughout all the classes, whatever the number of examples they feature in the dataset. That way, majority classes are not as favored as they usually are in multi-class approaches.

On non-reported datasets, computed measures for both algorithms are very close. On reported results, we observe that the higher the decrease of the confusion matrix norm, the higher the gain on MAUC and G-mean. However, we still have to investigate the relation between MAUC/G-mean and the norm of the confusion matrix. These preliminary results let us think that there might be no gain using CoMBo instead of Adaboost.MM in other cases, but there is no loss either (the computational times are the same).

$$\begin{pmatrix} 0.000 & 0.147 & 0.068 \\ 0.146 & 0.000 & 0.169 \\ 0.056 & 0.203 & 0.000 \end{pmatrix} \quad \begin{pmatrix} 0.000 & 0.011 & 0.055 \\ 0.656 & 0.000 & 0.179 \\ 0.246 & 0.048 & 0.000 \end{pmatrix}$$

CoMBo Adaboost.MM

Table 4: Confusion matrices obtained with Adaboost.MM and CoMBo, on the dataset `Connect`.

Finally, Table 4 illustrates what actually occurs on the dataset `Connect`[2]. The errors on minority classes 2 (83.5%) and 3 (29.4%) are dramatically high with Adaboost.MM which promotes the majority class (only 6.6% of errors). These differences are reduced with CoMBo: the error on the majority class reaches 20.2% while errors on minority classes decrease respectively to 31.5% and 25.9%. However, the real error is still higher with CoMBo: misclassified examples of the majority classes getting more numerous, it directly impacts the overall error rate. Such a behavior of CoMBo points out that it equally considers each class during the learning process, independently from any tricky misclassification cost.

These experiments acknowledge the smoothed learning processed by CoMBo over imbalanced classes. We think that this smoothing effect is related to the link between the minimization of the confusion matrix's norm and the MAUC/G-mean, since it allows CoMBo to perform well on all the classes, contrary to AdaBoost.MM, who performs excellently on majority classes and poorly on minority ones.

6 Discussion and future works

The framework proposed in this paper raises several questions and prospect works, some of which are discussed in this section.

Some potential extensions consist in using the result obtained in equation (7) in order to derive other cost sensitive algorithms. As briefly discussed in section 3, the optimization term depends on the different

[2]The dataset `Connect` was chosen for illustration because of readability (it only features 3 classes) and because it is much imbalanced.

loss functions defined over the classes. The only restriction on these losses is that their value should be greater than the one returned by the indicator function. This implies that it is possible to embed various informations into the loss functions, as long as they respect this condition. For instance, it is easy to include in the proposed method penalization terms defined over two classes simply by replacing the loss functions considered in section 3 with $\ell_{y_i,j}(h, \mathbf{x}_i) = c_{y_i,j} \exp(f_H(i,j) - f_H(i,y_i))$, where $c_{y_i,j}$ is a penalization term given as a prior or coming from a cost matrix.

Motivated by the theoretical and experimental results — think of the difference between the estimated error of the proposed method and AdaBoost.MM's — other works should focus on finding tighter bounds for the operator norm of the confusion matrix. Depending on the singular values computed from the confusion matrix, the bound given in equation (7) can be quite loose. Finding a tighter bound would then be a first step towards a more effective method in minimizing both the risk and the norm of the confusion matrix alike.

Continuing on the same line of thought, an important question that arises from the proposed framework is: *is it possible to obtain similar results for other norms*? Even though minimizing the different norms of the confusion matrix may be the same thing due to the equivalence between the norms, the main difference consists in whether it is possible to find an analytical expression of the norm. Some of the considered norms can be the $l1-$norm, entry-wise norms or even more exotic ones.

A second question that naturally comes to mind is: *is it possible to consider other definitions for the confusion matrix, such as loss-based matrices*, where the entries are replaced by loss functions. More generally, it would be interesting to see if there exists a relation between the choice of the confusion matrix and choice of the norm to be minimized.

Finally, based on the empirical results, we think that the norm of the confusion matrix is quite an useful tool for measuring the performance of a model in the multi-class and/or imbalanced classes setting, alongside classical measures as MAUC, G-mean, F-measure, etc.

7 Conclusion

The contributions of this paper are two-fold: first we make use of the norm of the confusion matrix as an error measure in order to derive a common bound for most cost sensitive methods; secondly, using the bound as a starting point, we show step by step how to obtain an extension of AdaBoost.MM for the imbalanced classes framework. Empirical results show that the proposed method compares favorably to other cost-based boosting methods. These performances are both due to the inherited boosting framework — which is more adapted to the multi-class problems than other frameworks — and to the weighting scheme obtained from the choice of the loss functions in section 3. Although much work remains to be done, as discussed in section 6, the proposed method is, to the best of our knowledge, the first boosting based approach that aims to actively minimize the norm of the confusion matrix, by including it in the learning process.

Acknowledgements

The authors would like to thank Liva Ralaivola for his inputs and suggestions on the work presented in this paper.

References

Abe, Naoki. An iterative method for multi-class cost-sensitive learning. In *In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 3–11, 2004.

Bordes, A., Ertekin, S., Weston, J., and Bottou, L. Fast kernel classifiers with online and active learning. *J. Machine Learning Research*, 6:1579–1619, 2005.

Chapelle, O. and Chang., Y. Yahoo! learning to rank challenge overview. In *JMLR Workshop and Conference Proceedings*, volume 14, pp. 1–24, 2011.

Chawla, Nitesh V., Lazarevic, Aleksandar, Hall, Lawrence O., and Bowyer, Kevin W. Smoteboost: Improving prediction of the minority class in boosting. In *PKDD'03*, pp. 107–119, 2003.

Cortes, Corinna and Mohri, Mehryar. Auc optimization vs. error rate minimization. In Press, MIT (ed.), *Advances in Neural Information Processing Systems (NIPS 2003)*, volume 16, Vancouver, Canada, 2004.

Elkan, Charles. The foundations of cost-sensitive learning. In *In Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pp. 973–978, 2001.

Estabrooks, Andrew, Jo, Taeho, and Japkowicz, Nathalie. A multiple resampling method for learning from imbalanced data sets. *Computational Intelligence*, 20(1):18–36, 2004.

Fawcett, Tom. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

Frank, A. and Asuncion, A. UCI machine learning repository, 2010. URL `http://archive.ics.uci.edu/ml`.

Hand, David J. and Till, Robert J. A simple generalisation of the area under the roc curve for multiple class classification problems. *Mach. Learn.*, 45(2):171–186, October 2001.

He, Haibo and Garcia, Edwardo A. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284, September 2009.

Morvant, Emilie, Koço, Sokol, and Ralaivola, Liva. PAC-Bayesian Generalization Bound on Confusion Matrix for Multi-Class Classification. In *International Conference on Machine Learning*, pp. 815–822, 2012.

Mukherjee, Indraneel and Schapire, Robert E. A theory of multiclass boosting. *CoRR*, abs/1108.2989, 2011.

Ralaivola, Liva. Confusion-based online learning and a passive-aggressive scheme. In *Neural Information Processing Systems Conference*, 2012.

Sun, Y., Kamel, M.S., and Wang, Y. Boosting for learning multiple classes with imbalanced class distribution. In *In 2006 IEEE International Conference on Data Mining, HongKong*, pp. 592–602, 2006.

Tang, K., Wang, R., and Chen, T. Towards maximizing the area under the roc curve for multi-class classification problems. In *Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, 2011.

Ting, K.M. A comparative study of cost-sensitive boosting algorithms. In *Int'l Conf. Machine Learning*, pp. 983–990, 2000.

Wang, Huanjing, Khoshgoftaar, Taghi M., and Napolitano, Amri. Software measurement data reduction using ensemble techniques. *Neurocomputing*, 92:124 – 132, 2012.

Wang, Shuo and Yao, Xin. Multiclass imbalance problems: Analysis and potential solutions. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 42(4):1119–1130, 2012.

Yue, Yisong, Finley, Thomas, Radlinski, Filip, and Joachims, Thorsten. A support vector method for optimizing average precision. In *SIGIR*, pp. 271–278, 2007.

Zhou, Zhi-Hua and Liu, Xu-Ying. On multi-class cost-sensitive learning. *Computational Intelligence*, 26 (3):232–257, 2010.