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

Sokol Koço, Cécile Capponi

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*

March 15, 2013

Abstract

In many multi-class classification problems, the misclassification rate as an error measure is not the relevant choice, think of the imbalanced classes problems. In order to overcome this shortcoming, several methods have been proposed where the error measure embeds richer informations than the mere misclassification rate. Yet, to the best of our knowledge, none of these methods makes use of one of the most natural tools in the multi-class setting: the confusion matrix.

Recent results show that using the norm of the confusion matrix as an error measure can be quite interesting due to the additional informations contained in the matrix, especially in the case of imbalanced classes. In this paper, we show step by step how to obtain a boosting-based method which minimizes the norm of the confusion matrix. The experimental results point out that the proposed method performs better that AdaBoost.MM on imbalanced datasets, while both methods are equivalent on balanced datasets.

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

1 Introdution

Learning from imbalanced data concerns theory and algorithms that process a relevant learning task whenever data is not uniformely distributed among classes. When facing imbalanced classes, the classification accuracy is not the fair measure to be optimized Fawcett (2006). Accuracy can be quite high in case of extreme imbalanced data: majority classes are promoted, while minority classes are not recognized. Such a bias gets stronger within the multi-class setting

In the binary setting, learning from imbalanced data has been quite 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 over classes (for example Estabrooks et al. (2004)), or/and by dealing with cost-sensitive methods (for example Ting (2000)), or with additional assumptions such as active learning within kernel-based methods (for example: Bordes & Bottou (2005)).

Despite the famous words "it is easy to generalize to more than two classes", learning 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 alternate measures of interest. Most of times, generalizing the binary setting to the multi-class setting is based on the one-vs-all (or one-vs-one) usual trade-off. It is worth to notice that some specific learning tasks have been addressed through the optimization of relevant measures within the multi-class imbalanced setting, whatever the average accuracy could be. For example, let us cite: Chapelle & Chang. studies some ranking measures, Yue et al. (2007) focused on the maximization of the Mean Average Precision (MAP) in the multi-label setting, Wang et al. (2012) addresses imbalanced feature selection through the maximization of the MAUC, the multi-class extension of the Area Under the ROC Curve, while K. Tang & 2011. maximizes the MAUC for improving classification. 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).

Furthermore, the *confusion matrix* is one of the most informative measure a multi-class learning system can rely on. Among other information, it contains the ways :

- the classifier gets right or wrong on one class,
- and the amount of confusion among (imbalanced) classes.

The sum of the entries of a row of the confusion matrix is equal to 1 independently from the number of examples having the class corresponding to the row. As such the confusion matrix constitutes a great tool that can be used to overcome the imbalanced classes problem. Moreover, if we consider the matrix containing only the non-diagonal elements, than summing over a row of this new matrix can be quite informing of how the corresponding class is recognized over the learning problem. Surprisingly, as far as we know, no-one proposes an algorithm that would optimize a metric computed from the confusion matrix.

In this work, we advocate that minimizing the norm of the confusion matrix is helpful for smoothing the accuracy among imbalanced classes, so that minority classes are considered as important as majority classes. We thus work on a multi-class learning framework, based on the confusion matrix. As far as we know, this work presents the first multi-class learning algorithm that minimizes the norm of the confusion matrix.

Starting from a strong multi-class classification theoretical setting Mukherjee & Schapire (2011), and helped by previous recent works on the confusion matrix Morvant et al. (2012); Ralaivola (2012), the aim of this paper is to sketch up a computationally and theoretically fair classification algorithm (section 4) that is ensured to minimize the norm of the confusion matrix, minimizing the classification error as proven in section 3. Boosting based, this algorithm greedly processes a sort-of regularization on imbalanced classes, in such a way that poorly represented classes are still of interest within the overall learning process, independently from any prior misclassification cost. Section 5 summerizes the experimental resultats of this algorithm, compared to Adaboost.MM Mukherjee & Schapire (2011). Section 6 wraps it all up with a discussion on the contributions of this paper and the future works.

2 General framework and Notations

The method proposed in this paper uses the confusion matrix as a performance measure in order to build a multi-class classifier from a boosting-based process. Before attacking the core of the problem and our main contribution, we introduce the different notations used throughout the paper.

The first part of this section contains the notations on the matrices and some of the tools that will be used to transform a confusion matrix towards an error measure. The second part introduces the boosting framework used to obtain the method that constitutes the main contribution of this paper.

2.1 General notations

The matrices are noted with bold capital letters like $\mathbf{C}$ and $\mathbf{C}(l, j)$, or simply $\mathbf{c}_{l,j}$, corresponds to the entry of the lth row and the jth column of $\mathbf{C}$. $\lambda_{max}(C)$ and $Tr(\mathbf{C})$ correspond respectively to the largest eigenvalue and the trace of $\mathbf{C}$, while $\|\mathbf{C}\|$ is its *spectral* or *operator norm*. $\|\mathbf{C}\|$ is defined as the square root of the largest eigenvalue of $\mathbf{C}^*\mathbf{C}$, where $\mathbf{C}^*$ is the conjugate transpose of $\mathbf{C}$. Let $\mathbf{A}$ and $\mathbf{B}$ be two matrices, then $\mathbf{AB}$ and $\mathbf{A} \cdot \mathbf{B}$ refer respectively to the inner product and the Frobenius inner product of $\mathbf{A}$ and $\mathbf{B}$.

The indicator function is denoted by $\mathbb{I}$ and, unless stated otherwise, K is the number of classes, m the number of examples and m_y if the number of examples of class y, where $y \in \{1, ..., K\}$.

2.2 Multi-class boosting framework

In this paper we make use of the boosting framework for multiclass classification introduced in Mukherjee & Schapire (2011), and more precisely the one defined for AdaBoost.MM. In this setting the distribution on the training examples is replaced by a cost matrix. Let $S = \{(x_i, y_{,i})\}$ be a training sample, where $x_i \in X$ and $y_i \in \{1, ..., K\}$. The cost matrix $\mathbf{D}$ is constructed so that for a given example (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 (x_i, y_i).

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

$$\mathbf{D}_T(i,l) \stackrel{def}{=} \begin{cases} \exp(f_T(i,l) - f_T(i,y_i)) & \text{if } l \neq y_i \\ -\sum\limits_{j \neq y_i} \exp(f_T(i,j) - f_T(i,y_i)) & \text{otherwise,} \end{cases} \quad (1)$$

where $f_T(i,l)$ is the score function computed as:

$$f_T(i,l) = \sum_{t=1}^{T} \alpha_t \mathbb{I}[h_t(i) = l]. \tag{2}$$

When needed, $f_T(i,l)$ will be noted as $f_{T,i,l}$ for readability reasons.

The output hypothesis of AdaBoost.MM is given by the following expression :

$$H(x) = \underset{l=1\cdots K}{argmax}\ f_T(i,l). \tag{3}$$

3 Boosting the confusion matrix

3.1 The confusion matrix as an error measure

Boosting algorithms, such as the AdaBoost family (AdaBoost, AdaBoost.M1, AdaBoost.MM,$\cdots$), are designed in order to greedily minimize the empirical error computed on the training sample. Their goal is therefore to iteratively construct a classifier H which minimizes :

$$\frac{1}{m}\sum_{i=1}^{m} \mathbb{I}[H(i) \neq y_i].$$

The loss functions considered in these methods reflect this goal, that is they take into consideration only the number of examples misclassified by H, independently from their class. For example, in the case of AdaBoost, the exponential loss forces the weak learners to focus on the most difficult examples. However, as mentioned in the introduction, in the case of imbalanced classes, this may not be optimal, since the weak learner could possibly be focused only on the examples of one class. Take for example, a binary classification problem where one of the classes makes up 99% of the training sample. The simple "majority class" classifier would have an error of at most 1%. Nevertheless its generalization capabilities would be catastrophic, since it can only recognize one class.

It would therefore be more preferable to have another error measure, and another loss function, which can take into considerations richer informations. In this paper, we propose to use a particular form of the confusion matrix of a classifier as an error measure. We start off by defining the *true confusion matrix* and *the empirical confusion matrix* for a classifier h.

Definition 1. *(True confusion matrix) The true confusion matrix* $\mathbf{A}$ *of a classifier* h *over a distribution* $\mathcal{D}$ *is defined as :*

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

Definition 2. *(Empirical confusion matrix) For a given classifier* h *and a sample* $S = \{(x_i, y_i)\}_{i=1}^{m} \sim \mathcal{D}$*, the empirical confusion matrix* $\mathbf{A}_S$ *of* h *is defined as :*

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

One may notice that the entries of a row of the confusion matrix sum up to 1, independently from the number of examples contained in the corresponding class. The diagonal entries of this matrix correspond to the correctly classified examples. Since our aim is to use the confusion matrix as an error measure, we zero these diagonal elements. The following definition gives the general terms of the new confusion matrices:

Definition 3. *For all* $h \in \mathcal{H}$ *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)*:*

$$\hat{\mathbf{c}}_{l,j} \overset{def}{=} \begin{cases} 0 & \text{if } l = j \\ \hat{a}_{l,j} & \text{otherwise,} \end{cases} \tag{4}$$

$$\mathbf{c}_{l,j} \overset{def}{=} \begin{cases} 0 & \text{if } j = j \\ a_{l,j} = \mathbb{P}_{(\mathbf{x},y)\sim\mathfrak{D}}(f(\mathbf{x}) = j | y = l) & \text{otherwise.} \end{cases} \tag{5}$$

Let $\mathbf{p} = [P(y = 1), ..., P(y = K)]$ be the vector of class priors distribution, then using the new definition of the confusion matrix, it is easy to see that, for a given classifier h:

$$R(h) \stackrel{def}{=} P_{(x,y)\sim\mathcal{D}}(h(x) \neq y) = \|\mathbf{C}\|_1 \tag{6}$$

where $\|\mathbf{C}\|_1$ is the $l1$-norm of matrix $\mathbf{C}$. This simple, yet beautiful, result means that it is possible to retrieve the true error rate of h from its confusion matrix.

In this paper we focus on the operator norm of the confusion matrix, which is given by the square root of the largest eigenvalue of the matrix. Using the result in equation 6 and the equivalency between the norms, we have the following relation between the operator norm and the true risk:

$$R(h) \leq \sqrt{K}\|\mathbf{C}\| \tag{7}$$

Both equation 6 and equation 7 imply that minimizing the norm of the confusion matrix can be a good strategy in order to have a small risk.

3.2 Bounding the confusion matrix

The result given in equation 7 bounds 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. In order to overcome this difficulty, we make use of a particular case of Theorem 1 given in Ralaivola (2012). This particular case consist in choosing the indicator function $\mathbb{I}$ as the loss function. The following corollary is the direct consequence of this choice applied to the theorem.

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 computer for a classifier h over the training sample S.

Instead of minimizing the operator norm of the empirical confusion matrix $\mathbf{C}_S$, we propose to minimize an upper bound of $\|\mathbf{C}_S\|$.

$$\begin{aligned} \|\mathbf{C}_S\| &= \sqrt{\lambda_{max}(\mathbf{C}_S^*\mathbf{C}_S)} \\ &\leq \sqrt{Tr(\mathbf{C}_S^*\mathbf{C}_S)} \end{aligned}$$

The matrix $\mathbf{C}_S^*\mathbf{C}_S$ is positive semi-definite, hence all its eigenvalues are positives. The equality is simply the rewrite of the operator norm of $\mathbf{C}_S$, while the inequality comes from the fact that the trace is equal to the sum of the eigenvalues. We focus now on the value of $Tr(\mathbf{C}_S^*\mathbf{C}_S)$.

$$\begin{aligned} Tr(\mathbf{C}_S^*\mathbf{C}_S) &= \textstyle\sum_{l=1}^{K} \mathbf{C}_S^*\mathbf{C}_S(l,l) \\ &= \textstyle\sum_{l=1}^{K} \sum_{j=1}^{K} \hat{\mathbf{c}}_{l,j}\hat{\mathbf{c}}_{j,l} \\ &= \textstyle\sum_{l=1}^{K} \sum_{j\neq l} \hat{\mathbf{c}}_{l,j}\hat{\mathbf{c}}_{j,l} \\ &\leq \textstyle\sum_{l=1}^{K} \sum_{j\neq l} \hat{\mathbf{c}}_{l,j} \\ &= \textstyle\sum_{l=1}^{K} \sum_{j\neq l} \frac{1}{m_l} \sum_{i=1}^{m} \mathbb{I}[y_i = l]\mathbb{I}[H(i) = j] \end{aligned}$$

The first and the second equality come from the definition of the trace and the confusion matrix, while the third equality comes from the fact that the diagonal entries of the confusion matrix $\mathbf{C}_S$ are 0.

As for the inequality, it is simply the consequence of the fact that all the entries of the confusion matrix are smaller than 1. Finally, the last equality is obtained using the definition of the entries of $\mathbf{C}_S$.

In the next step, we make use of the boosting framework given in Mukherjee & Schapire (2011), and more precisely of the definitions of f_T and H. For readability reasons, $f_T(i,j) - f_T(i,y_i)$ is noted as $\Delta f_T(i,j,y_i)$.

$$\begin{aligned} Tr(\mathbf{C}_S^* \mathbf{C}_S) &= \sum_{l=1}^{K} \sum_{j \neq l} \frac{1}{m_l} \sum_{i=1}^{m} \mathbb{I}[y_i = l] \mathbb{I}[H(i) = j] \\ &= \sum_{i=1}^{m} \sum_{j \neq y_i} \frac{1}{m_{y_i}} \mathbb{I}[H(i) = j] \\ &\leq \sum_{i=1}^{m} \sum_{j \neq y_i} \frac{1}{m_{y_i}} e^{\Delta f_T(i,j,y_i)} \mathbb{I}[H(i) = j] \\ &\leq \sum_{i=1}^{m} \sum_{j \neq y_i} \frac{1}{m_{y_i}} e^{\Delta f_T(i,j,y_i)}. \end{aligned}$$

In order to obtain the first inequality, note that the term $\sum_{j \neq y_i} \mathbb{I}[H(i) = j]$ is non zero (and equal to 1) only for those examples on which the classifier H errs. That is to say that there exists at least one $j \neq y_i$ such as $f_T(i,j) \geq f_T(i,y_i)$. Hence the term $\sum_{j \neq y_i} \exp(f_T(i,j) - f_T(i,y_i))$ is at least equal to 1. The last inequality follows quite naturally.

Taking a step back and looking at what we've obtained so far, gives:

$$\|\mathbf{C}\| \leq \sqrt{Tr(\mathbf{C}^* \mathbf{C})} \; and \; Tr(\mathbf{C}_S^* \mathbf{C}_S) \leq \sum_{i=1}^{m} L_T(i), \tag{8}$$

where $L_T(i) = \sum_{j \neq y_i} \frac{1}{m_{y_i}} \exp(f_{T,i,j} - f_{T,i,y_i})$, is the loss computed for the example (x_i, y_i). The final step consists in putting together these results.

The different losses $L_T(\cdot)$ on the examples of the training sample are computed after round T. Since all the different parameters, such as α_t and h_t $(\forall t)$, are known, we can redefine the score function f_T as follows:

$$f_t^{norm}(i,l) = \frac{\sum_{t=1}^{T} \alpha_t \mathbb{I}[h_t(i) = l]}{\sum_{t=1}^{T} \alpha_t},$$

which is simply a normalized version of the original score function given in equation 2. The second result obtained in equation 8 is still correct, since the new score function does not change the predictions returned by H.

The term $\sum_{i=1}^{m} L_T(i)$ takes its minimal value when all the classifiers h_t correctly classify all the examples of S, that is $\forall i, f_T(i,y_i) = \sum_{t=1}^{T} \alpha_t$ or, replacing f_T by f_T^{norm}, $\forall i, f_T(i,y_i) = 1$. Rewriting the loss with f_T^{norm}, we have:

$$\begin{aligned} \sum_{i=1}^{m} L_T(i) &= \sum_{i=1}^{m} \sum_{j \neq y_i} \frac{1}{m_{y_i}} \exp(f_{T,i,j}^{norm} - f_{T,i,y_i}^{norm}) \\ &= \frac{1}{m_1} \sum_{i \in S_1} \sum_{j \neq 1} \exp(f_{T,i,j}^{norm} - f_{T,i,y_1}^{norm}) + \cdots \\ &\quad + \frac{1}{m_K} \sum_{i \in S_K} \sum_{j \neq K} \exp(f_{T,i,j}^{norm} - f_{T,i,y_k}^{norm}) \\ &\geq \frac{1}{m_{y_1}} \sum_{i \in S_1} \frac{(K-1)}{e} + ... + \frac{1}{m_K} \sum_{i \in S_K} \frac{(K-1)}{e} \\ &= \frac{K(K-1)}{e} \end{aligned}$$

This result shows that if $K > 2$ than the loss $\sum_{i=1}^{m} L_T(i)$ after iteration T is strictly greater than 1. Since we are in a multiclass setting, $K > 2$ is not too much of a limitation. The direct consequence of this result is the following inequality:

$$\sqrt{Tr(\mathbf{C}_S^* \mathbf{C}_S)} \leq \sqrt{\sum_{i=1}^{m} L_T(i)} \leq \sum_{i=1}^{m} L_T(i). \tag{9}$$

Combining equation 9 and the first result of equation 8, we finally obtain:

$$\|\mathbf{C}_S\| \leq \sum_{i=1}^{m} L_T(i). \tag{10}$$

3.3 Choosing the confusion matrix

The result obtained in equation 10 gives an upper bound on the norm of the confusion matrix, which is the general loss $L_T = \sum_{i=1}^{m} L_T(i)$. Taking a closer look to this loss, one may notice that it is merely the sum of simpler loss functions, each one defined on an example from the training set S, that is $L_T(i) = \sum_{l \neq y_i} \frac{1}{m_{y_i}} \exp f_T(i,l) - f_T(i,y_i)$.

In order to use the boosting framework presented in Mukherjee & Schapire (2011) we need to define a cost matrix $\mathbf{D}$ so that $\forall i$, and $\forall l \neq y_i, \mathbf{D}(i,l) \leq \mathbf{D}(i,y_i)$. Moreover $\mathbf{D}$ should be such that the total loss computed on $\mathbf{D}$ should be equal to $L_T = \sum_{i=1}^{m} L_T(i)$.

Taking into account the fact that the different singular losses are fairly similar to the one used in AdaBoost.MM, the most straightforward choice for $\mathbf{D}$ is the following :

$$\mathbf{D}_T(i,l) \stackrel{def}{=} \begin{cases} \frac{1}{m_{y_i}} \exp(f_{T,i,l} - f_{T,i,y_i}) & \text{if } l \neq y_i \\ -\sum\limits_{j \neq y_i} \frac{1}{m_{y_i}} \exp(f_{T,i,j} - f_{T,i,y_i}) & \text{otherwise,} \end{cases} \tag{11}$$

4 The core of CoMBo

In this section, we introduce a new boosting method based on the results obtained in the previous sections and we show that the loss decreases after every step of the algorithm, similar to AdaBoost.MM.

4.1 The Confusion Matrix Boosting Algorithm

The pseudo-code of the proposed method is given in algorithm 1. The inputs for this algorithm are the classical inputs for all boosting methods similar to the AdaBoost family, that is, a training sample S, the total number of iterations T and a weak learner WL. 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: using the weak learner WL in order to build the set of weak classifiers and using the predictions of h_t to update the cost matrix $\mathbf{D}_t$. At each round t, WL takes as 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 $\mathbf{D}_t$. The underlying idea is that the better h_t performs over $\mathbf{D}_t$, the greater the edge δ_t and the importance given to h_t.

The update rule for the cost matrix is designed so that the misclassification cost is increased for the examples misclassified by h_t and is decreased for the correctly classified ones. This forces the weak learner WL to focus on the most difficult examples. The main difference between our method and AdaBoost.MM is the use of the term $\frac{1}{m_{y_i}}$ in the update rule, where m_{y_i} is the number of examples having the same class y_i. The direct consequence of this, is that the misclassification cost on an example (x_i, y_i) depends also from 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 outputted prediction is the class that obtains the biggest score.

4.2 Bounding the loss

First off we recall the minimal weak learning condition as given in Mukherjee & Schapire (2011).

Definition 4. *(Minimal weak learning condition) Let D^{eor} be the space of all cost matrices $\mathbf{D}$ which put the least cost on the correct label, that is $\forall (x_i, y_i), l, \mathbf{D}(i,y_i) \leq \mathbf{D}(i,l)$. Let B^{eor}_{γ} be the space of baselines $\mathbf{B}$ which are γ more likely to predict the correct label for every example (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 :*

$$\forall \mathbf{D} \in D^{eor}, \exists h \in \mathcal{H} : \mathbf{D} \cdot \mathbf{1}_h \leq \max_{\mathbf{B} \in B^{eor}} \mathbf{D} \cdot \mathbf{B}, \tag{12}$$

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

Algorithm 1 CoMBo : Confusion Matrix BOosting

Given

- $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ where $x_i \in X$, $y_i \in \{1, \cdots, K\}$
- T the number of iterations, WL a weak learner
- $\forall i \in \{1, \cdots, m\}$, $\forall l \in \{1, \cdots, K\}$ $f_1(i, l) = 0$
- $\mathbf{D}_1(i,l) = \begin{cases} \frac{1}{m_{y_i}} & \text{if } y_i \neq l \\ \frac{-(K-1)}{m_{y_i}} & \text{if } y_i = l \end{cases}$

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

Get h_t with edge δ_t on $\mathbf{D}_t$, where:

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

Compute α_t as:

$$\alpha_t = \frac{1}{2} \ln \frac{1+\delta_t}{1-\delta_t}.$$

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

end for

Output final hypothesis :

$$H(x) = \underset{l \in 1,...,k}{\operatorname{argmax}} f_T(x,l),$$

$$\text{where } \; f_T(x,l) = \sum_{t=1}^{T} \mathbb{I}[h_t(x) = l]\alpha_t$$

In the rest of this paper, we will consider a particular case of baselines, which are the closest to the uniform. These baselines, noted $\mathbf{U}_\gamma$, have weights $(1-\gamma)/k$ on incorrect labels and $(1-\gamma)/k + \gamma$ on the correct ones. The weak learning condition is given by :

$$\mathbf{D} \cdot \mathbf{1}_h \leq \mathbf{D} \cdot \mathbf{U}_\gamma \tag{13}$$

All of the weak classifiers returned by WL during the training phase verify this weak learner condition.

The following result shows that the general loss L_t decreases with each iteration, if the weak classifier h_t satisfies the weak learning condition. This result and its proof are fairly 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 $h_{t,m}$ 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 $\sum\limits_{i=1}^{m} \sum\limits_{l \neq y_i} \exp(f_t(i,l) - f_t(i,y_i))$, 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 = edge of h_t.

Proof. Recall that the loss function L_t, and $L_t(i)$ are defined as

$$L_t = \sum_{i=1}^{m} \sum_{l \neq y_i} \mathbf{D}_t(i,l) = \sum_{i=1}^{m} \sum_{l \neq y_i} \frac{1}{m_{y_i}} \exp(f_{t,i,l} - f_{t,i,y_i})$$

$$L_t(i) = \sum_{l \neq y_i} \mathbf{D}_t(i,l) = \sum_{l \neq y_i} \frac{1}{m_{y_i}} \exp(f_t(i,l) - f_t(i,y_i))$$

The weak classifier h_t returned by WL satisfies the edge condition, that is:

$$\mathbf{D_t} \cdot \mathbf{1}_{h_t} \leq \mathbf{D_t} \cdot \mathbf{U}_{\delta_t}, \tag{14}$$

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 different definitions of $\mathbf{D_t}$, $\mathbf{1}_{h_t}$ and $\mathbf{U}_{\delta_t}$, the classification cost of h_t (left side of 14) is given by:

$$\begin{aligned} \mathbf{D}_t \cdot \mathbf{1}_{h_t} &= -\sum_{i \in S_+} L_{t-1}(i) \\ &\quad + \sum_{i \in S_-} \tfrac{1}{m_{y_i}} \exp(f_{t-1,i,h_t(i)} - f_{t-1,i,y_i}) \\ &= -A_+^t + A_-^t \end{aligned}$$

while the cost of $\mathbf{U}_{\delta_t}$ (right side of 14) is given by:

$$\mathbf{D}_t \cdot \mathbf{U}_{\delta_{t,m}} = -\delta_t \sum_{i=1}^{m} L_{t-1}(i) = -\delta_t L_{t-1}$$

Injecting these two costs in 14, we have :

$$A_+^t - A_-^t \geq \delta_t L_{t-1}. \tag{15}$$

If we take a closer look at the drop of the loss after choosing h_t and its weight α_t, we have:

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

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 15 gives a lower bound for $A_+^t - A_-^t$, while $A_+^t + A_-^t$ is upper-bounded by L_{t-1}. Hence,

$$\begin{aligned} L_{t-1} - L_t &= \sum_{i \in S_+} L_{t-1}(i)(1 - e^{-\alpha_t}) \\ &\geq \left(\tfrac{e^{\alpha_t} - e^{-\alpha_t}}{2}\right) \delta_t L_{t-1} - \left(\tfrac{e^{\alpha_t} + e^{-\alpha_t} - 2}{2}\right) L_{t-1}. \end{aligned}$$

Therefore, the result of the lemma:

$$\begin{aligned} L_t &\leq \left(1 - (\tfrac{e^{\alpha_t} - e^{-\alpha_t}}{2})\delta_t + \left(\tfrac{e^{\alpha_t} + e^{-\alpha_t} - 2}{2}\right)\right) L_{t-1} \\ &= \left(\tfrac{1}{2}((1 - \delta_t)e^{\alpha_t} + (1 + \delta_t)e^{-\alpha_t})\right) L_{t-1}. \end{aligned} \tag{16}$$

□

The expression of the loss drop given in the Lemma 1 can be further simplified. Indeed, if we choose the value of α_t as given in the pseudo-code of Algorithm 1, than 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 resumes this result.

Theorem 1. *Let $\delta_1, \cdots, \delta_T$ be the edges of the classifiers $h_1, \cdots, h_T$ returned by WL 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, $K(K-1)e^{-T\gamma^2/2}$.

dataset	AdaBoost.MM $\|\mathbf{C}\|$	CoMBo $\|\mathbf{C}\|$	AdaBoost.MM error	CoMBo error
Abalone	2.939	1.373	0.076	0.086
Car	0.328	0.055	0.002	0.005
Connect-4	1.066	0.437	0.024	0.036
Nursery	0.923	0.201	0.007	0.012
Poker	1.778	0.471	0.033	0.050
Pendigits	0.004	0.011	0.000	0.000
Im. Segm.	0.138	0.257	0.004	0.006
Letter	0.187	0.194	0.013	0.013

Table 1: Adaboost.MM vs. CoMBo on error and norm of the confusion matrix. The last three datasets are balanced.

5 Experimental results

5.1 Datasets and experimental setup

8 datasets were used in the experiments, the same used in Mukherjee & Schapire (2011). They are all from the UCI Machine Learning Repository Frank & Asuncion (2010), and are all related to multi-class learning tasks, mainly classification. They exhibit various degrees of imbalanced data, as well as various number of instances and attributes. Since the work is concerned with multi-class imbalanced datasets, class distributions must be specified:

- datasets `Pendigits` (10 classes), `Letter` (26 classes) and `Image segmentation` (7 classes) are balanced.
- `Abalone` is fairly imbalanced. It features 28 classes: 13 classes represent less than 1% of the total number of instances each, while 4 classes represent more than 10% out of the total number of instances each. Each of the 11 remaining classes represents between 1 and 10% of the dataset.
- The dataset `Car` features 4 classes, which contains respectively 70.023%, 22.222%, 3.993%, and 3.762% of the instances: the two last classes are much less represented than the two first ones.
- The dataset `Connect-4` features 3 classes, which contains respectively 65.83%, 24.62%, and 9.55% of the instances.
- The dataset `Nursery` has 5 classes, which contain respectively 33.333%, 0.015%, 2.531%, 32.917%, and 31.204% of the population.
- The dataset `PokerHand` is the most imbalanced. It features 10 classes, where the four first contains respectively around 50%, 42%, 5% and 2% of the dataset. Each of the six other classes represents less than 0.5% of the dataset.

For each dataset, we performed 10-folds cross-validations and averaged the results. Two measures are reported: estimations of the error and of the confusion matrix norm. CoMBo and Adaboost.MM ran for 200 iterations.

5.2 Results

Results are presented in table 1: the estimated confusion matrix norms are reported, together with the estimated generalization errors.

The results on balanced datasets (Letter, Pendigits and Im. segmentation) are similar with Adaboost.MM and CoMBo: estimated errors and norms of the confusion matrix are deeply close. These preliminar results let us think that, in case of multi-class balanced datasets, there is no gain using CoMBo instead of Adaboost.MM, but there is no loss either (the computational times are quite the same).

Concerning imbalanced datasets, CoMBo turns out challenging. The estimated real error with CoMBO leans to be a bit worse than the one of Adaboost.MM. Meanwhile, as expected, the estimated norm of the confusion matrix is much smaller with CoMBo. Having a closer look at results on the `PokerHand` fairly imbalanced dataset, the decreasing of the norm is drastic. It confirms that using CoMBo, the accuracy

$\begin{pmatrix} 0.000 & 0.006 & 0.000 & 0.001 \\ 0.000 & 0.000 & 0.000 & 0.015 \\ 0.000 & 0.167 & 0.000 & 0.000 \\ 0.000 & 0.250 & 0.013 & 0.000 \end{pmatrix}$	$\begin{pmatrix} 0.000 & 0.045 & 0.004 & 0.007 \\ 0.000 & 0.000 & 0.021 & 0.010 \\ 0.000 & 0.000 & 0.000 & 0.013 \\ 0.000 & 0.000 & 0.000 & 0.000 \end{pmatrix}$
Adaboost.MM on Car	CoMBo on Car
$\begin{pmatrix} 0.000 & 0.000 & 0.048 \\ 0.890 & 0.000 & 0.110 \\ 0.582 & 0.000 & 0.000 \end{pmatrix}$	$\begin{pmatrix} 0.000 & 0.232 & 0.137 \\ 0.181 & 0.000 & 0.212 \\ 0.079 & 0.266 & 0.000 \end{pmatrix}$
Adaboost.MM on Connect	CoMBo on Connect

Table 2: Confusion matrices obtained with Adaboost.MM and CoMBo, on datasets Car and Connect-4.

of the classifier on minority classes is improved, whereas it turns sour on majority classes. Somehow, we could say that the performances of the classifier is smoothed among all the classes, whatever their representation are within the dataset. That way, majority classes are not as promoted as usually in multi-class approaches.

Let us illustrate on table 2 what actually occurs on the two smallest imbalanced datasets, where the norm of the confusion matrix decreases with CoMBo: `Car` and `Connect-4`. Except for the diagonal entries[1], each (l, j) represents the probability that instances of class l are classified as j by the classifier. Hence, the accuracy of class l is estimated as $1 - \sum_{j \neq l} C(l, j)$.

With CoMBo on the `Car` dataset, the errors on minority classes 3 and 4 get smaller, while the error on the first (majority) class increases w.r.t. Adabost.MM. This the smoothing effect of CoMBo.

On the `Connect-4` dataset, the misclassification rates on classes 2 (100%) and 3 (58.2%) are dramatically high with Adaboost.MM which promotes the majority class. With CoMBo, classes 2 and 3 are as well recognized as class 1, although class 1 is the majority class.

In both datasets, the estimated real error is higher with CoMBo: this is explained by the fact that misclassified examples of the majority classes getting more numerous, it directly impacts the overall error.

Such a behavior of CoMBo points out that it equally considers each class during the learning process. These experiments acknowledge the smoothed learning processed by CoMBo over imbalanced classes. Then, the integration of cost-sensitive errors could be easily performed during the minimization process on $\|\mathbf{C}\|$.

6 Discussion

The method proposed in this paper aims at minimizing the operator norm of the confusion matrix, which is used as a performance measure, instead of the classical misclassification error. In order to do so, we proposed in section 3 to minimize an upper-bound of the norm of the empirical confusion matrix. Indeed, it is regrettable to make use of an upper-bound, since what we wish to minimize is the norm itself, in the same fashion as the COPA algorithm Ralaivola (2012). The first natural follow up of this work is to obtain a novel method which greedily minimizes the norm of the confusion matrix. We think that using the same multi-class boosting framework as the one used here, is be the best way to tackle this problem.

In section 3.2 we mentioned that the confusion matrix used in this paper is just a particular case of a larger family of confusion matrices. The matrices in this family are obtained by replacing the indicator function in equations 4 and 5 by a loss function. Doing so allows us to consider special cases of matrices, such as the fact that confusing class a with class b is worse than confusing a and c (think about automated diagnosis system). The second perspective of this work is thus to extend the results obtained here, to the more general family of loss-based confusion matrices. It then could easily integrate any prior on cost-sensitive misclassifications, hopefully as a constant within the norm of the generalized confusion matrix.

Last but not least, we would like to mention another possible extension of the confusion matrix as a performance measure framework. Confusion matrices are generally defined for learning samples, but they can also be defined for an ensemble of classifiers. Hence future work will also be focused on using confusion matrices for the estimation of the performances for ensembles of classifiers and, hopefully, obtaining new bound for methods based on ensemble learning.

[1]Recall that the diagonal is set to zero for it must be as high as possible, thus it is not taken into account during the norm minimization process.

7 Conclusion

We proposed a novel method based on a multi-class boosting framework that uses the operator norm of the confusion matrix as an alternate performance measure, taking advantages of the richer informations embedded in this matrix. Our main contribution lies in the fact that, to the best of our knowledge, this is the first boosting method based on the idea of minimizing the norm of the confusion matrix. We prowed in section 3 how to obtain a loss function which upper-bounds the operator norm of the confusion matrix, and how to obtain a boosting method that minimizes this loss function. Our method is given in section 4 and we showed in the same section that the loss decreases with each round. Finally, the experimental results given in section 5 show that our method performs better than AdaBoost.MM when the norm of the confusion matrix is considered as a performance measure. This method fairs better than AdaBoost.MM in the case of imbalanced samples.

References

Bordes, S. Ertekin, J. Weston 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*.

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.

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`.

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

K. Tang, R. Wang and 2011., T. Chen 7-11 August. Towards maximizing the area under the roc curve for multi-class classification problems. In *Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*.

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.

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.

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