Loss Factors for Learning Boosting Ensembles from Imbalanced Data

Roghayeh Soleymani*, Eric Granger*, Giorgio Fumera†
*Laboratoire d'imagerie, de vision et d'intelligence artificielle, École de technologie supérieure,
Université du Québec, Montreal, Canada,
Email: rSoleymani@livia.etsmtl.ca, Eric.Granger@etsmtl.ca
† Dept. of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy,
Email: Fumera@diee.unica.it

Abstract—**Class imbalance is an issue in many real world applications because classification algorithms tend to misclassify instances from the class of interest when its training samples are outnumbered by those of other classes. Several variations of AdaBoost ensemble method have been proposed in literature to learn from imbalanced data based on re-sampling. However, their loss factor is based on standard accuracy, which still biases performance towards the majority class. This problem is mitigated using cost-sensitive Boosting algorithms, although it can be avoided at the outset by modifying the loss factor calculation. In this paper, two loss factors, based on F-measure and G-mean are proposed that are more suitable to deal with imbalanced data during the Boosting learning process. The performance of standard AdaBoost and of three specialized versions for class imbalance (SMOTEBoost, RUSBoost, and RB-Boost) are empirically evaluated using the proposed loss factors, both on synthetic data and on a real-world face re-identification task. Experimental results show a significant performance improvement on AdaBoost and RUSBoost with the proposed loss factors.**

I. Introduction

Many pattern recognition applications suffer from imbalanced class distributions because the performance of the classifiers designed on unequal proportions from each class becomes biased towards the class with higher number of samples. This is due in part to the fact that standard accuracy is typically used as the criterion to optimize classification systems during design.

Boosting ensemble learning algorithms, first introduced in AdaBoost [1], can effectively promote a weak learner that performs slightly better than random guessing into a stronger ensemble. Base classifiers are trained iteratively while samples are assigned weights that increase when the corresponding samples are misclassified, and decrease when they are correctly classified. Depending on the type of base classifier, the weights are either used directly by the classifier, or to re-sample data. In the latter case, the weights are proportional to the probability of the corresponding samples to be selected for training the classifier in each iteration.

The performance of Boosting ensembles degrades when data is imbalanced for two reasons. First, since the majority class constitutes a great proportion of the dataset, its samples are more likely to be selected for training classifiers. The second reason is related to the calculation of loss factor which is used to update sample weights, and to set the contribution of base classifiers in class prediction function. The standard loss factor is calculated based on the proportion of misclassified samples (considering their weights) to the overall number of training samples. However, when the classes are imbalanced, this loss factor depends on the ability of the classifiers to recognize the majority class more than the minority class.

Several approaches in literature aim to compensate for the weaknesses of AdaBoost in learning from imbalanced data. Some approaches re-balance training data by undersampling the majority class [2], [3], by upsampling the minority class through interpolation with neighbors [4], by duplicating minority samples [5], or by combining upsampling and undersampling [6], [7]. However, these approaches still suffer from the unsuitable loss factor calculation and may be affected by high imbalance levels between classes.

Other approaches introduce different cost factors for minority and majority classes into loss factor calculation or weight update formula, e.g., AdaC [8], AdaCost [9], and CSB [10] algorithms. The disadvantage is that these cost factors must be set among the possible costs for a particular dataset. In [11] a different weight update factor is defined based on error rate with regard to both classes using geometrical mean of these error rates.

In this paper, a new loss factor is proposed to improve the performance of Boosting algorithms on imbalanced data. To this aim, an imbalance-compatible accuracy measure based on either the F_β-measure or G-mean is exploited, instead of standard accuracy. The modified loss factor is integrated into the weight update formula and to set the contribution of classifiers in final class prediction, to avoid a bias towards the majority class. The proposed loss factor does not need calibration of any cost parameters. Nevertheless, the value of β when using F_β-measure is an application-dependent parameter.

II. Boosting Ensembles from Imbalanced Data

Analogous to most of the classifier systems, AdaBoost and its variants (AdaBoost.M1 and AdaBoost.M2 [12]) process the samples in the same way regardless of the class to which they belong. Accordingly, when data is imbalanced the performance of these ensembles becomes biased towards majority classes.

We focus on two-class problems, and denote the design samples as $\mathbf{S} = \{(\mathbf{x}_i, y_i)\}$, $i = 1, ..., N$, where $y_i \in \{-1, 1\}$ are the class labels, $\mathbf{x}_i \in \mathbb{R}^n$ are data samples and $N = N^+ + N^-$, where N^+ and N^- are the number of minority and majority class samples, respectively. With AdaBoost.M1 (see Algorithm 1), all training samples are initially weighted with $1/N$. Given a classifier that does not use sample weights, $\mathbf{S}$ is resampled based on such weights and a new set $\mathbf{S}'$ with a new weight distribution $\mathbf{W}'$ is used to train a classifier C_t. This classifier is then tested on $\mathbf{S}$, and a loss factor ϵ_t is calculated as the sum of the weights of misclassified samples:

$$\epsilon_t = \sum_{(i,Y_i):y_i \neq Y_i} W_t(i) \ , \tag{1}$$

where Y_i is the label predicted by C_t for $\mathbf{x}_i$. If the classifier is too weak ($\epsilon_t > 0.5$, where 0.5 corresponds to random guessing), it is discarded and $\mathbf{S}$ is resampled to train a new classifier. Then, ϵ_t is used to define a weight update factor:

$$\alpha_t = \frac{\epsilon_t}{1 - \epsilon_t} \ , \tag{2}$$

to update the weights of samples for the next iteration:

$$\mathbf{W}_{\mathrm{t+1}}(i) = \mathbf{W}_{\mathrm{t}}(i)\alpha_t^{\frac{1}{2}|\mathrm{y_i - Y_i}|} \ . \tag{3}$$

The weights of the misclassified samples increase while the weights of the correctly classified samples decrease. Factor α_t is also used to weigh the contribution of C_t in the overall class predictions of the ensemble by either majority voting or score averaging function:

$$H(\mathbf{x}) = \sum_{t=1}^{T} h_t(\mathbf{x}) \log \frac{1}{\alpha_t} \ , \tag{4}$$

where $h_t(\mathbf{x}_i)$ denotes the output of C_t (either a classification score or a label) for an input sample $\mathbf{x}$. This process is repeated to design a predefined number of T classifiers.

In line (2.i) of AdaBoost.M1, $\mathbf{S}$ is resampled based on the weights $\mathbf{W}_t$. Even with equal weights of minority and majority class samples, the majority class samples are more likely to be selected. If the minority class samples are included in $\mathbf{S}'$, majority class samples contribute more to the loss factor calculation in line (2.iv). Therefore, the ensemble performance is biased toward the majority class. In the subsequent steps (line 2.vii), the weights of misclassified majority samples increase and these samples have a higher chance to stay in $\mathbf{S}'$ during next iteration(s).

Internal and external techniques including data upsampling, data undersampling, combined sampling (both upsampling and undersampling) and cost-sensitive methods have been integrated into AdaBoost by many researchers to handle imbalance-related issues [13]. In particular, SMOTEBoost [4] modifies AdaBoost.M2 by generating artificial minority samples using the Synthetic Minority Over-sampling Technique (SMOTE). RUSBoost [2] is similar to SMOTEBoost, with the difference that instead of upsampling the minority class, the majority class of the imbalanced data is undersampled using random under-sampling (RUS) to create a balanced subset before training the new classifier in each iteration. Random Balance Boosting (RB-Boost) [7] combines SMOTE and RUS to create subsets with random and different skew levels in AdaBoost.M2 to increase diversity among them.

Algorithm 1: AdaBoost.M1 Algorithm.

```
Input: Training set: $\mathbf{S} = \{(\mathbf{x}_i, y_i); i = 1, ..., N\}$, $y_i \in \{-1, 1\}$,
       $N = N^+ + N^-$, # of iterations: $T$
1 Initialize $\mathbf{W}_1(i) = \frac{1}{N}$ for $i = 1, ..., N$.
2 for $t = 1, .., T$ do
     i    Create new training set $\mathbf{S}_t^{'}$ with weight distribution $\mathbf{W}_t^{'}$.
     ii   Train classifier $C_t$ on $\mathbf{S}_t^{'}$ with $\mathbf{W}_t^{'}$.
     iii  Test $C_t$ on $\mathbf{S}$ and produce labels $\{Y_i, i = 1, ..., N\}$.
     iv   Compute the pseudo-loss $\epsilon_t$ for $\mathbf{S}$ with Eq. (1).
     v    If $\epsilon_t > 0.5$ go to step i
     vi   Calculate the weight update parameter $\alpha_t$ with Eq. (2).
     vii  Compute $\mathbf{W}_{\mathrm{t+1}}(i)$ with Eq. (3).
     viii Normalize $\mathbf{W}_{\mathrm{t+1}}$ such that: $\sum \mathbf{W}_{\mathrm{t+1}} = 1$.
```

A comprehensive review of cost-sensitive Boosting ensembles can be found in [13]. The most representative ones are summarized here. Given m_i as the cost of misclassification of sample $\mathbf{x}_i$, in AdaCost [9], two cost adjustment functions are defined for each sample as $\phi_+ = -0.5m_i + 0.5$ and $\phi_- = 0.5m_i + 0.5$. They are integrated into the weight update factor as well as weight update formula of AdaBoost (Eqs. 2 and 3) and consequently the weight of misclassified minority samples increase more sharply than the weight of misclassified majority samples. In CSB [10] a cost factor is embedded only in weight update formula (Eq. 3). AdaC [8] also introduces two different cost factors m_i for two classes into the weight update factor and weight update formula to balance the classification performance. In RareBoost [25] two different α_ts for minority and majority classes are defined as $\frac{1}{2}ln(^{TP_t}/_{FP_t})$ and $\frac{1}{2}ln(^{TN_t}/_{FN_t})$, respectively. In [11], the authors define two pseudo errors e_t^+ and e_t^- for minority and majority classes as the corresponding percentage of misclassified samples. Then α_t is defined as:

$$\alpha_t = \ln(\sqrt{m_i \alpha_t^+ \alpha_t^-}) \ , \tag{5}$$

where $\alpha_t^+ = \frac{1-e_t^+}{e_t^+}$, $\alpha_t^- = \frac{1-e_t^-}{e_t^-}$ and m_i is a multiplier to control the weight of each sample. The issue with this loss factor is that, if there are no misclassified samples in one class or in both classes, α_t is undefined.

A drawback of the cost-sensitive Boosting approaches is that they require ad hoc tuning of the corresponding cost factor to each sample, on a given data set, e.g., by a searching in the cost space. The alternative proposed in this paper avoids biasing Boosting ensembles toward the majority class by using an imbalance-compatible accuracy measure to compute the error.

III. Performance Metrics for Imbalanced Data

The performance of a classifier can be measured to guide the classifier modelling during design or to evaluate discrimination during testing. Accuracy, calculated as the proportion of correctly classified samples to overall number of samples, is not a suitable performance measure when data is imbalanced because it over-values correct classification of majority samples. Metrics focused on a single class like true positive rate (TPR) or recall (Re), false positive rate (FPR), false negative rate (FNR) and true negative rate (TNR) are inter-related, so increasing one of them decreases the others. Therefore, a trade-off between them must be considered for evaluating classifiers performance. Receiver Operating Characteristic (ROC) curves allow to visualize the trade-off between TPR and FPR (or $1-$TNR) over different operating points. A global scalar metric, the area under ROC curve (AUC) is also used to evaluate the performance of a classification system over all possible operating points. However, ROC space does not reflect the effect of imbalance because large variations in the number of misclassified majority class samples (FP) results in a small change in FPR. In contrast, precision (Pr) considers the number of correct classification of minority class (TP) with regard to the number of misclassified majority class samples (FP). Precision declines severely when correct classification of minority class is increased at the expense of misclassification of majority class. Therefore, precision-recall curve is preferred to ROC curve to visualize the performance of system under different operating settings when data is imbalanced [26]. Inspired from ROC curve, area under precision-recall curves (AUPR) may be employed to assess the overall classifier performance under class imbalance.

For a fixed operating setting, G-mean evaluates the classification performance on both classes:

$$\text{G-mean} = \sqrt{\text{TNR} \cdot \text{TPR}} \; . \tag{6}$$

In addition, precision and recall can be weighted and combined into a single performance metric as the F_β-measure:

$$F_\beta = \frac{(1+\beta^2)\text{Pr} \cdot \text{Re}}{\beta^2 \text{Pr} + \text{Re}} = \frac{(1+\beta^2)\text{TP}}{(1+\beta^2)\text{TP} + \text{FP} + \beta^2 \text{FN}} \; , \tag{7}$$

where $\beta \geq 1$ and $\beta < 1$ allows to give a higher importance to correct classification of the minority and majority class, respectively. Therefore, β can be set based on the application in hand.

For applications in which data is imbalanced and the cost of misclassification of both classes is equal ($c_{\text{FN}} = c_{\text{FP}}$), G-mean is a suitable performance metric. In applications with highly imbalanced data, the cost of incorrect classification of minority class is usually higher: $c_{\text{FN}} \gg c_{\text{FP}}$. Therefore, F$_\beta$ with $\beta \geq 1$ is a desirable metric to evaluate classifiers in such applications.

Some authors account for the imbalance level of data by defining variants of the above performance metrics [14]–[16]. For example a measure of expected accuracy in terms of AUC is defined as $\pi(1-\pi)(2AUC-1)+1/2$ where $\pi = {}^{N^+}\!/_{N}$ is the data skew level [15]. Precision-recall gain curves [16] is proposed to make up for the weaknesses of precision-recall curve by normalizing precision and recall in terms of π as: $precG = \frac{Pr-\pi}{(1-\pi)Pr}$, and $recG = \frac{Re-\pi}{(1-\pi)Re}$.

Masking the effect of imbalance in performance metrics can misguide the learning process of Boosting ensembles. Other performance measures in literature like cost and Brier curves are usually used to evaluate classifiers in cost analysis to select a proper operating point for the desired conditions. ROC, precision-recall, cost and Brier curves are usually produced by varying decision threshold, over real-valued scores produced by a classifier. Consequently, using global metrics like AUC, AUPR, Brier scores in Boosting ensembles may limit the choice of the type of base classifier. Therefore, F_β and G-mean metrics are found to be the best choices. They are considered in this paper to modify the loss factor calculation in Boosting ensembles, to avoid bias of performance towards majority class.

IV. Unbiased Loss Factors for Boosting from Imbalanced Data

The loss factor calculation in AdaBoost and its variants biases the performance of Boosting ensembles towards correct classification of majority class, even with data rebalancing. In this paper, a new loss factor is proposed to avoid this issue by evaluating the performance of each classifier in terms of either F_β-measure or G-mean. This loss factor can be integrated into any Boosting ensemble technique.

To this aim, the weight vector is separated into two vectors for minority and majority classes:

$$\mathbf{W}_t^+ = \{\mathbf{W}_t(i), i = 1, ..., N | y_i = 1\} \; , \tag{8}$$
$$\mathbf{W}_t^- = \{\mathbf{W}_t(i), i = 1, ..., N | y_i = -1\} \; . \tag{9}$$

Then, weighted versions of true positive, false positive, true negative and false negative counts are defined as:

$$\text{TP}_t = \sum_{(j,Y_j):Y_j=1} \mathbf{W}_t^+(j), j = 1, ..., N^+ \; , \tag{10}$$
$$\text{FP}_t = \sum_{(j,Y_j):Y_j=1} \mathbf{W}_t^-(j), j = 1, ..., N^- \; , \tag{11}$$
$$\text{TN}_t = \sum_{(j,Y_j):Y_j=-1} \mathbf{W}_t^-(j), j = 1, ..., N^- \; , \tag{12}$$
$$\text{FN}_t = \sum_{(j,Y_j):Y_j=-1} \mathbf{W}_t^+(j), j = 1, ..., N^+ \; . \tag{13}$$

Based on these values, the accuracy of a classifier is computed in terms of F_β-measure and G-mean respectively as:

$$A_F = \frac{(1+\beta^2)\text{TP}_t}{(1+\beta^2)\text{TP}_t + \text{FP}_t + \beta^2\text{FN}_t} \; , \tag{14}$$
$$A_G = \sqrt{\frac{\text{TP}_t \times \text{TN}_t}{(\text{TP}_t + \text{FN}_t) \times (\text{TN}_t + \text{FP}_t)}} \; . \tag{15}$$

To measure the error of the classifiers, the corresponding loss factor are defined respectively as:

$$L_t = 1 - A_F = \frac{\text{FP}_t + \beta^2\text{FN}_t}{(1+\beta^2)\text{TP}_t + \text{FP}_t + \beta^2\text{FN}_t} \; , \tag{16}$$
$$L_t = 1 - A_G \; . \tag{17}$$

An advantage of the loss factor based on F-measure (16) is that it allows one to bias the Boosting ensemble to correctly

Algorithm 2: Proposed Boosting with F$_\beta$ loss.

```
Input: Training set: $\mathbf{S} = \{(\mathbf{x}_i, y_i); i = 1, ..., N\}$, $y_i \in \{-1, 1\}$,
       $N = N^+ + N^-$, # of iterations: $T$, weighting
       parameter: $\beta$, re-sampling technique: RS.
1 Initialize $\mathbf{W}_1(i) = \frac{1}{N}$ for $i = 1, ..., N$.
2 for $t = 1, .., T$ do
      i    Create new training set $\mathbf{S}_t^{'}$ with weight distribution $\mathbf{W}_t^{'}$
           using RS.
      ii   Train classifier $C_t$ on $\mathbf{S}_t^{'}$ with $\mathbf{W}_t^{'}$.
      iii  Test $C_t$ on $\mathbf{S}$ and produce labels $\{Y_i, i = 1, ..., N\}$.
      iv   Define weight vectors $\mathbf{W}_t^{+}$ and $\mathbf{W}_t^{-}$ as in Eqs. (8) and (9).
      v    Compute loss factor $L_t$ with Eq. (16)
      vi   If $L_t > \frac{N^-}{(1+\beta^2)N^+ + N^-}$ go to step i
      vii  Calculate the weight update parameter: $\alpha_t = \frac{L_t}{1-L_t}$
      viii Update $\mathbf{W}_{t+1}(i)$ as in Eq. (3).
      ix   Normalize $\mathbf{W}_{t+1}$ such that: $\sum \mathbf{W}_{t+1} = 1$.
```

classify the minority or the majority classes by tuning β. The loss factor based on G-mean (17) is suitable in applications where the correct classification of both classes is equally important.

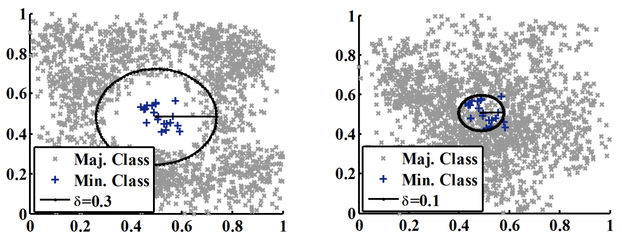
The condition $\epsilon_t > 0.5$ in line (v) of AdaBoost.M1 (Algorithm 1) means that classifiers in a Boosting ensemble should perform better than random guessing. A random classifier that predicts the positive label with probability p results in G-mean $= \sqrt{p(1-p)}$. For $p = 0$ and $p = 1$, G-mean $= 0$. When $p = 0.5$, then G-mean become maximum and equals 0.5. Thus, when the loss factor is calculated using Eq. (17), the accuracy criterion of Boosting ensemble remains as 0.5.

When F$_\beta$-measure is used as the evaluation metric, the worst random classifier is the one that predicts everything as positive [16]. Therefore, when the loss factor is calculated using Eq. (16), the accuracy criterion of 0.5 in line (3.v) of AdaBoost.M1 should be replaced by $l_r = \frac{N^-}{(1+\beta^2)N^+ + N^-}$. In Algorithm 2, the proposed loss factor (16) is integrated into a data re-sampling Boosting ensemble. After calculation of α_t based on the proposed loss factor, $\mathbf{W}_t$ is updated using Eq. (3) and the Boosting algorithm continues for the next iterations.

V. Experiments and Results

The performance of a classifier tends to degrade when data is imbalanced. In particular, imbalance becomes an issue when it increases the level of overlap between classes [17]. To investigate this issue, the first experiments in this section focus on a synthetic 2D data set with a controllable level of overlap between classes. Then, experiments are performed with a real-world video surveillance dataset for face re-identification application. It can be characterized by a significant level of class imbalance. For both data sets, ensembles are trained on skew level of $\lambda_{\text{tr}} = 1 : 100$ and classification performance is evaluated for two different imbalance levels of testing data ($\lambda_{\text{te}} = 1 : 1$,and $1 : 100$).

Four well-known Boosting algorithms are compared in this paper: AdaBoost, SMOTEBoost, RUSBoost and RB-Boost (in Algorithm 2, RS $\in$ {Ada, SMOTE, RUS, RB}), with ensemble sizes fixed to $T = 100$. Their original version is compared



(a) Minimum overlap. (b) Maximum overlap.

Fig. 1: Examples of synthetic data.

with three modified versions obtained by using the loss factor of [11] with $\mu_i = 1$ (see Sect. II), and using the loss factors proposed in this work, (16) and (17). β in (16) is set as 2 in experiments with both synthetic and real-world datasets. Other values have also been considered on RUSBoost for completeness. SVM with RBF kernel [18] is used as the base classifier (parameters are set similarly to [23]). Classification performance is evaluate using the F_2-measure and G-mean.

A. Results with synthetic data:

The synthetic data used for experiments is generated in 2D space. First, $N^+ = 20$ minority class samples are generated with a normal distribution as $N(m_+, \sigma_+)$, where m_+ and σ_+ indicate the mean and standard deviation, respectively. Then, $N^- = 100$ normal distributions, whose means, denoted by $m_{-,j}, j = 1, ..., N^-$, are in turn generated from a uniform distribution around m_+ such that they keep a margin distance δ from m_+. This margin is used to control the level of overlap between minority and majority classes. For each $m_{-,j}$, $N^+ = 20$ samples are randomly generated from its corresponding normal distribution $N(m_{-,j}, \sigma_-)$. For each overlap level, one set is generated for training and another for testing. The maximum imbalance level between classes in these datasets is 1:N^-/N^+ (1:100). For $m_+ = (0.5, 0.5), \sigma_+ = \sigma_- = 0.11$, such that $(0, 0) \leq x_i \leq (1, 1), i = 1, ..., 2020$, we varied the parameter δ between 0.1 (maximum overlap) and 0.3 (minimum overlap) with the step of 0.05. For example, data generated with two different δ's are presented in Fig. 1.

The average performance of Boosting ensembles are compared in Tables I in terms of F_2-measure and G-mean. As expected, all these ensembles perform significantly better when the overlap is small. RB-Boost and SMOTEBoost often exhibit the best performance, especially when the overlap between classes is larger.

The loss factor of [11] improves AdaBoost performance, but worsens the performance of RB-Boost in terms of G-mean. For other Boosting ensembles using this loss factor, the results in the tables are presented by (-). The reason is that these ensembles could not be generated since α in Eq. (5) is undefined if $FP = 0$ or $FN = 0$. AdaBoost also failed in high overlap level, where $\delta = 0.1$.

When F-measure (16) is used, the performance of AdaBoost significantly improves in all cases except from $\delta = 0.25$. The performance of RUSBoost improves over all levels of overlap

TABLE I: Average of F_2-measure (a) and G-mean (b) performance of proposed and baseline techniques on synthetic data over 5 replications over different levels of skew and overlap of test data.

Classification Systems		δ 0.3		0.25		0.2		0.15		0.1	
		λ_{te} 1:1	1:100	1:1	1:100	1:1	1:100	1:1	1:100	1:1	1:100
(a) F_2-MEASURE											
A1:	**AdaBoost [12]**	1.00 ± 0.00	0.97 ± 0.00	0.99 ± 0.00	0.96 ± 0.00	0.77 ± 0.02	0.68 ± 0.02	0.59 ± 0.04	0.56 ± 0.03	-	-
A2:	**AdaBoost-F**	1.00 ± 0.00	0.97 ± 0.00	0.95 ± 0.01	0.92 ± 0.01	0.96 ± 0.01	0.82 ± 0.01	0.86 ± 0.02	0.59 ± 0.01	0.98 ± 0.00	0.63 ± 0.02
S1:	**SMOTEBoost [4]**	1.00 ± 0.00	0.92 ± 0.01	1.00 ± 0.00	0.85 ± 0.00	1.00 ± 0.00	0.79 ± 0.01	1.00 ± 0.00	0.70 ± 0.01	0.99 ± 0.00	0.60 ± 0.01
S2:	**SMOTEBoost-F**	1.00 ± 0.00	0.92 ± 0.00	1.00 ± 0.00	0.85 ± 0.00	1.00 ± 0.00	0.79 ± 0.01	1.00 ± 0.00	0.69 ± 0.01	0.99 ± 0.00	0.62 ± 0.01
R1:	**RUSBoost [2]**	0.99 ± 0.00	0.30 ± 0.01	0.98 ± 0.01	0.26 ± 0.01	0.96 ± 0.01	0.23 ± 0.00	1.00 ± 0.00	0.18 ± 0.00	0.92 ± 0.01	0.21 ± 0.01
R2:	**RUSBoost-F**	0.97 ± 0.01	0.54 ± 0.03	1.00 ± 0.00	0.46 ± 0.01	0.99 ± 0.00	0.47 ± 0.01	1.00 ± 0.00	0.35 ± 0.01	0.95 ± 0.01	0.34 ± 0.01
RB1:	**RB-Boost [7]**	1.00 ± 0.00	0.92 ± 0.00	1.00 ± 0.00	0.93 ± 0.00	0.99 ± 0.00	0.84 ± 0.01	0.99 ± 0.00	0.77 ± 0.01	0.99 ± 0.00	0.69 ± 0.01
RB2:	**RB-Boost-F**	1.00 ± 0.00	0.93 ± 0.00	1.00 ± 0.00	0.90 ± 0.00	1.00 ± 0.00	0.82 ± 0.01	1.00 ± 0.00	0.75 ± 0.01	1.00 ± 0.00	0.67 ± 0.01
(b) G-MEAN											
A1:	**AdaBoost [12]**	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.85 ± 0.02	0.84 ± 0.02	0.71 ± 0.03	0.71 ± 0.03	-	-
A3:	**AdaBoost-G [11]**	-	-	-	-	0.97 ± 0.00	0.97 ± 0.00	0.79 ± 0.02	0.79 ± 0.02	0.86 ± 0.01	0.86 ± 0.01
A4:	**AdaBoost-G**	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.85 ± 0.02	0.85 ± 0.02	0.67 ± 0.01	0.67 ± 0.01	0.57 ± 0.02	0.56 ± 0.02
S1:	**SMOTEBoost [4]**	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.96 ± 0.01	0.98 ± 0.00
S3:	**SMOTEBoost-G [11]**	-	-	-	-	-	-	-	-	-	-
S4:	**SMOTEBoost-G**	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.97 ± 0.01	0.98 ± 0.00
R1:	**RUSBoost [2]**	0.95 ± 0.02	0.94 ± 0.00	0.91 ± 0.02	0.92 ± 0.00	0.81 ± 0.03	0.91 ± 0.00	0.99 ± 0.00	0.87 ± 0.01	0.94 ± 0.01	0.89 ± 0.00
R3:	**RUSBoost-G [11]**	-	-	-	-	-	-	-	-	-	-
R4:	**RUSBoost-G**	0.97 ± 0.01	0.93 ± 0.00	0.97 ± 0.01	0.91 ± 0.00	1.00 ± 0.00	0.91 ± 0.00	1.00 ± 0.00	0.89 ± 0.00	0.87 ± 0.02	0.87 ± 0.01
RB1:	**RB-Boost [7]**	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.98 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.98 ± 0.00
RB3:	**RB-Boost-G [11]**	-	-	1.00 ± 0.00	1.00 ± 0.00	0.96 ± 0.00	0.95 ± 0.00	0.96 ± 0.00	0.95 ± 0.00	0.96 ± 0.01	0.96 ± 0.01
RB4:	**RB-Boost-G**	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	1.00 ± 0.00	0.99 ± 0.00	0.98 ± 0.00	0.99 ± 0.00

and skew of test data. For SMOTEBoost, our loss factor has no impact in most overlap and skew levels. Its performance improves when $\delta = 0.1$ ($\lambda_{\mathrm{te}} = 1:100$), and worsen when $\delta = 0.1$ ($\lambda_{\mathrm{te}} = 1:100$). The performance of RB-Boost improves when $\lambda_{\mathrm{te}} = 1:1$, and decrease when $\lambda_{\mathrm{te}} = 1:100$.

When G-mean (17) is used, the performance of AdaBoost improves only when $\delta = 0.1$. The performance of RUSBoost improves when $\delta = 0.2$ and 0.1. In most of the other cases, our loss factor provides the same performance.

For AdaBoost and RUSBoost, using the F-measure to compute the loss factor is significantly more effective than using G-mean due to the fact that G-mean is less sensitive to imbalance than F-measure. In some cases the performance decrease after using G-mean (17). Interestingly, the proposed loss factor allows for an AdaBoost performance comparable to that of the other Boosting algorithms, even tough it does not involve any data rebalancing procedure. Using the proposed loss factors guides the sample selection and classifiers weight selection in AdaBoost such that bias of performance is reduced. The classification performance evaluated by the F-measure decreases by a considerable amount as the imbalance level of testing data increases (see Table I (a)), whereas G-mean reflects only a slight effect of imbalance (Table I (b)).

However, the improvements on the performance of Boosting ensembles after using the proposed loss factors is greater for higher imbalance levels than for balanced data.

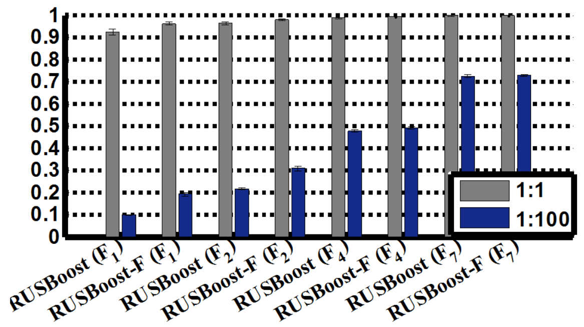


Fig. 2: Performance of RUSBoost for different values of β.

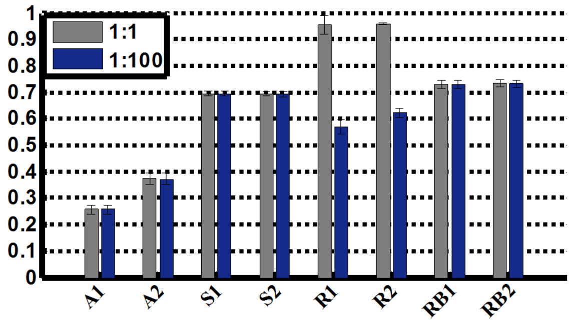
In Figure 2, the performance of RUSBoost and RUSBoost-F for different values of β is compared when $\delta = 0.2$. Evaluation is done in terms of the same F_β-measure that is used in loss factor calculation. The performance of RUSBoost improves for $\beta = 1$ and 2, and the improvement tends to decrease for higher β values.

B. Results with video surveillance data:

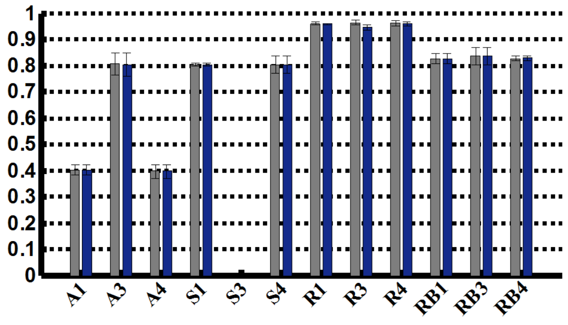
Face re-identification is a video surveillance application where the faces of appearing in in videos are recognized at different time instants and locations over a distributed network of surveillance cameras. An important challenge is that the number of faces captured from the individuals of interest is greatly outnumbered by those faces of other people.

This imbalance problem is addressed in literature using an architecture of individual-specific ensembles of classifiers [19]–[23]. In our experiments results are produced on the Face in Action (FIA) dataset [24]. It contains 5-seconds video sequences from individuals recorded in both indoor and outdoor environments during three sessions that are three months apart, and with 6 cameras. We used only videos captured with a frontal camera in indoor environment. The facial captures in videos of each individual are randomly divided to 5-folds such that 2 folds are used for training and 3 folds are used for testing. We carried out 10 rounds of experiments, each one replicated 5 times. In each round, an individual is selected randomly to populate the minority class and 100 individuals are selected to populate the majority class (both training and test sets have skew level of 1:100). Face patterns are obtained in the same way as [19], [23].

Fig. 3 (a) shows that the performance of AdaBoost and RUSBoost improves after using (16). The performance of SMOTEBoost and RB-Boost does not change significantly. Using (17) has no impact on the performance of the Boosting



(a) F_2-measure.



(b) G-mean.

Fig. 3: Average of F_2-measure (a) and G-mean (b) performance of proposed and baseline techniques on videos from FIA dataset over 10 target indivuals, each with 5 replications (test data is sampled to balanced (1:1) and imbalanced (1:100) cases).

ensembles. SMOTEBoost fails when loss factor of [11] is used. For RB-Boost the loss factor of [11] has no noticeable impact on the performance. However, using the loss factor of [11] increase the performance of AdaBoost significantly.

VI. Conclusion

A new loss factor calculation technique is proposed in this paper to improve the performance of Boosting ensembles for classification from imbalanced data. This loss factor is calculated by evaluating classification performance in terms of either F-measure or G-mean, instead of the standard accuracy, which biases the learning algorithm in favor of the majority class. The F-measure effectively represents the impact of imbalance and allows to emphasize the importance of correct classification of the majority class. In contrast, G-mean maximizes the performance of the classifier on both classes, equally. Although the proposed method is not a cost-sensitive approach, expected classification costs could be accounted for by tuning the β parameter of the F-measure. Experimental results on well-known Boosting ensembles show that the proposed loss factors can significantly improve the performance of AdaBoost and RUSBoost, especially when F-measure is used. In addition, they produce ensembles that are more robust to a wide range of skew levels of test data. This is promising for training discriminant classifier systems with unbalanced class distributions when skew level varies over time.

References

[1] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*, Springer, 1995.

[2] C. Seiffert, T.M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSboost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Systems, Man and Cybernetics A*,, 2010.

[3] M. Galar, A. Fernández, E. Barrenechea, and F. Herrera, "Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling," *Patt. Rec.*, 2013.

[4] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Knowledge Discovery in Databases: PKDD*, 2003.

[5] D. Mease, A. Wyner, and A. Buja, "Cost-weighted boosting with jittering and over/under-sampling: Jous-boost," *J. Machine Learning Research*, 2007.

[6] H. Guo and H. L. Viktor, "Learning from imbalanced data sets with boosting and data generation: the databoost-im approach," *ACM SIGKDD Explorations Newsletter*, 2004.

[7] J. F. Díez-Pastor, J. J. Rodríguez, C. García-Osorio, and L. I. Kuncheva, "Random balance: Ensembles of variable priors classifiers for imbalanced data," *Knowledge-Based Systems*, 2015.

[8] Y. Sun, M. S. Kamel, A. K. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Patt. Rec.*, 2007.

[9] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "Adacost: misclassification cost-sensitive boosting," in *ICML*, 1999.

[10] K. M. Ting, "A comparative study of cost-sensitive boosting algorithms," in *ICML*, 2000.

[11] M.-J. Kim, D.-K. Kang, and H. B. Kim, "Geometric mean based boosting algorithm with over-sampling to resolve data imbalance problem for bankruptcy prediction," *Expert Systems with Applications*, 2015.

[12] Y. Freund, R. E. Schapire *et al.*, "Experiments with a new boosting algorithm," in *ICML*, 1996.

[13] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches," *IEEE Trans. Systems, Man, and Cybernetics C,* , 2012.

[14] V. García, R. A. Mollineda, and J. S. Sanchez, "Theoretical analysis of a performance measure for imbalanced data," in *ICPR*, 2010.

[15] J. Hernández-Orallo, P. Flach, and C. Ferri, "A unified view of performance metrics: translating threshold choice into expected classification loss," *J. Machine Learning Res.*, 2012.

[16] P. Flach and M. Kull, "Precision-recall-gain curves: PR analysis done right," in *NIPS*, 2015.

[17] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Inf. Sciences*, 2013.

[18] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Trans. on Intelligent Systems and Technology*, 2011.

[19] C. Pagano, E. Granger, R. Sabourin, G. Marcialis, and F. Roli, "Adaptive ensembles for face recognition in changing video surveillance environments," *Information Sciences*, 2014.

[20] P. V. Radtke, E. Granger, R. Sabourin, and D. O. Gorodnichy, "Skew-sensitive boolean combination for adaptive ensembles–an application to face recognition in video surveillance," *Inf. Fus.*, 2014.

[21] M. De-la Torre, E. Granger, P. V. Radtke, R. Sabourin, and D. O. Gorodnichy, "Partially-supervised learning from facial trajectories for face recognition in video surveillance," *Inf. Fus.*, 2015.

[22] M. De-la Torre, E. Granger, and R. Sabourin, "Adaptive skew-sensitive fusion of ensembles and their application to face re-identification," *Pattern Recognition*, 2015.

[23] R. Soleymani, E. Granger, and G. Fumera, "Classifier ensembles with trajectory under-sampling for face re-identification," in *ICPRAM*, 2016.

[24] R. Goh, L. Liu, X. Liu, and T. Chen, "The CMU face in action (FIA) database," in *An. and Mod. of Faces and Gestures*, 2005.

[25] M. V. Joshi, V. Kumar, and R. C. Agarwal, "Evaluating boosting algorithms to classify rare classes: Comparison and improvements," in *ICDM*, 2001.

[26] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *ICML*, 2006.