Non-Linear Gradient Boosting for Class-Imbalance Learning

Jordan Frery JORDAN.FRERY@UNIV-ST-ETIENNE.FR
Amaury Habrard AMAURY.HABRARD@UNIV-ST-ETIENNE.FR
Marc Sebban MARC.SEBBAN@UNIV-ST-ETIENNE.FR
Univ. Lyon, Univ. St-Etienne F-42000,
UMR CNRS 5516, Laboratoire Hubert-Curien, France

Liyun He-Guelton LIYUN.HE-GUELTON@WORLDLINE.COM
Worldline, 95870 Bezons, France

Editors: Luís Torgo, Stan Matwin, Nathalie Japkowicz, Bartosz Krawczyk, Nuno Moniz, and Paula Branco

Abstract

Gradient boosting relies on linearly combining diverse and weak hypotheses to build a strong classifier. In the class imbalance setting, boosting algorithms often require many hypotheses which tend to be more complex and may increase the risk of overfitting. We propose in this paper to address this issue by adapting the gradient boosting framework to a non-linear setting. In order to learn the idiosyncrasies of the target concept and prevent the algorithm from being biased toward the majority class, we suggest to jointly learn different combinations of the same set of very weak classifiers and expand the expressiveness of the final model by leveraging their non-linear complementarity. We perform an extensive experimental study using decision trees and show that, while requiring much less weak learners with a lower complexity (fewer splits per tree), our model outperforms standard linear gradient boosting.

Keywords: meta-learning, non-linear boosting, class imbalance learning

1. Introduction

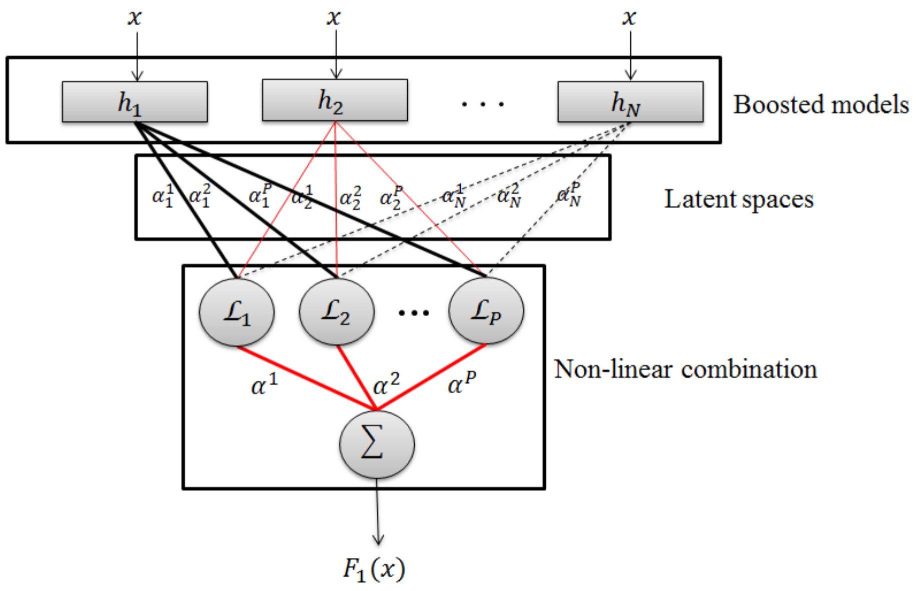
Class imbalance learning has received a lot of interest during the past decade from the machine learning community (He and Garcia, 2008; Chawla et al., 2002). This phenomenon takes its origin from the numerous real-life applications, such as fraud detection, click detection or anomaly detection, where the number of positive examples (i.e data from the class of interest) is much smaller than the *negative* data. As ensemble learning proved to be very effective on real-life datasets such as *bagging* (e.g. random forest (Breiman, 2001)), *stacking* (Wolpert, 1992), *cascade generalization* (Gama and Brazdil, 2000), *boosting* (Freund and Schapire, 1997), etc., it is natural that many contributions around ensemble methods with class imbalance learning were brought to light (Liu et al., 2009; Galar et al., 2012). However, these methods are all based on either cost sensitive learning (Fan et al., 1999) or sampling strategies (Chawla et al., 2003) which often leads to a dramatic effect on the posterior probability due to the change in the training class distribution (Dal Pozzolo et al., 2015). In this paper, we rather work on the original dataset. We focus on gradient boosting which stands out in this class imbalance learning context by combining different classifiers linearly where each of them aims at correcting the error of the previously formed linear combination. As the boosting algorithm adds more classifiers in the ensemble, the minority class receives more attention along the iterations of boosting. Moreover, the popularity of gradient boosting has been increased by recent implementations showing the scalability of the method even with billions of examples (Chen and Guestrin, 2016; Ke et al., 2017).

Despite these advantages, gradient boosting-based methods face a limitation: they usually perform a linear combination of the learned hypotheses which may limit the expressiveness of the final model to reach complex target concepts. More specifically, in class imbalance problems, the class of interest is underrepresented and thus more base learners induced from a more complex family of hypotheses is often required. This may lead to overfitting as well dramatic impacts in terms of time and storage complexity.

In this work, we propose to address these issues by (i) learning different linear projections of the base model outputs and (ii) optimizing a non-linear mapping of these projections. The principle of our method, called NLB for Non-Linear Boosting, is illustrated in Fig. 1. At first glance, it looks similar to boosted neural networks, as done in Han et al. (2016); Opitz et al. (2017), where the embedding layer is learned with boosting in order to infer more diversity. However, our method aims at learning the weak hypotheses iteratively, each new weak learner tries to minimize the error made by the network restricted to the previous hypotheses (see the thickest lines in Fig. 1 that will be used to learn h_2). The other main difference comes from the back-propagation that is performed at each step only on the parameters related to the weak learner subject to an update (see the red lines in Fig. 1). Moreover, inspired from previous research in domain adaptation (Becker et al., 2013) and boosted-multi-task learning (Chapelle et al., 2011), NLB resorts to the same set of boosted weak learners, projects their outputs in different latent spaces and takes advantage of their complementarity to learn non linearly the idiosyncrasies of the underlying concept. It is worth noticing that in standard gradient boosting methods, weak learners are combined linearly and in the presence of highly imbalanced data, many iterations (number of weak learners) are necessary before the boosting method focuses on the minority class examples. With such models, capturing the peculiarities of the class of interest is challenging. Our method allows us to learn more with fewer and weaker models reducing the risk of overfitting and making it suitable to capture patterns from the minority class.

One key point of our method is its ability to generate diversity in the set of combinations and thus benefit from their non-linear complementarity to deal with complex target problems. Along these lines, NLB makes use of very weak base classifiers built over different distributions that tend to emphasize on misclassified instances. Despite its simplicity, NLB benefits from an extensive predictive power.

To the best of our knowledge, only (Garcã-Pedrajas et al., 2007) tackled this topic by proposing a non-linear boosting projection method where, at each iteration of boosting, the authors build a new neural network only with the examples misclassified during the previous round. They finally take the new feature space induced by the hidden layer and feed it as the input space to the next learner. This procedure has the main drawback to train one neural network between each weak classifier requiring a higher time complexity. Moreover, neural networks need extra pre-processing steps to work with special feature types (e.g. categorical features) when our meta-learner rather works in the space of the base model outputs which are continuous and bounded.



x
x
x
h_1
h_2
h_N
Boosted models
α_1^1 α_1^2 α_1^P α_2^1 α_2^2 α_2^P α_N^1 α_N^2 α_N^P
Latent spaces
$\mathcal{L}_1$
$\mathcal{L}_2$
$\mathcal{L}_P$
α^1 α^2 α^P
Σ
Non-linear combination
$F_1(x)$

Figure 1: Graphical representation of our Non-Linear gradient Boosting method: the first top layer corresponds to the learned weak classifiers; the second layer represents different linear combinations of their outputs; the bottom layer proceeds a non-linear transformation of those combinations. The thickest lines show the needed activated path to learn a given classifier (here h_2). The red lines show the update performed only on the parameters concerned by this weak learner. The dashed lines are not taken into account at this iteration.

The rest of this paper is organized as follows: after a presentation of the notations in Section 2, our new non-linear gradient boosting algorithm NLB is presented in Section 3. Section 4 is devoted to a large experimental comparison in the class imbalance setting. We conclude the paper in Section 5.

2. Notations and definitions

In this paper, we consider a binary supervised learning setting with a training set $\mathcal{S} = \{x_i, y_i\}_{i=1}^{M}$ of M examples where $x_i \in \mathcal{X}$ is a feature vector and $y_i \in \{-1, 1\}$ is its label. In this study we aim at learning a function $f : \mathcal{X} \to \mathbb{R}$ that gives a real value to any new $x \in \mathcal{X}$. Let h_t be a *weak learner*. Our final strong model is an ensemble as a set $\{h_t\}_{t=1}^{T}$ of T weak models.

In standard boosting, these models are combined in a linear fashion as follows:

$$F(x) = \sum_{t=1}^{T} \alpha_t h_t(x), \tag{1}$$

where α_t stands for the weight of the weak learner h_t. Note that in our case, and generally in the gradient boosting methods, the weak learners are regressors. In Friedman

(2001) the author studies the case of regression trees as weak learners which later became a standard in gradient boosting.

3. Non-Linear Gradient Boosting algorithm

We now present our Non-Linear gradient Boosting algorithm, named NLB. As shown in Fig. 1, our method maintains P different representations that correspond to different combinations of the T weak learners, projecting their outputs into different latent spaces. Every representation p is updated right after a weak hypothesis is learned. The outputs given by the p representations are then merged together to build a strong classifier $F(x)$. To capture non-linearities during this process, we propose to pass the output of each representation p into a non-linear function $\mathcal{L}_p$. We define the prediction of our model $F(x)$ as follows:

$$F(x) = \sum_{p=1}^{P} \alpha^p \mathcal{L}_p\Big(\sum_{t=1}^{T} \alpha_t^p h_t(x)\Big), \tag{2}$$

where α_t^p are the weights projecting the outputs of the weak learner h_t in the latent space p and α^p the weight of this representation. Eq (2) illustrates clearly the difference with linear boosting formulation of Eq (1). We denote by F_k the classifier restricted to the first k weak learners: $F_k(x) = \sum_{p=1}^{P} \alpha^p \mathcal{L}_p\Big(\sum_{t=1}^{k} \eta \alpha_t^p h_t(x)\Big)$.

Our method aims thus at combining the same set of classifiers into different latent spaces. A key point here relies in making these classifiers diverse while still being relevant in the final decision. To achieve this goal, we update every weak learner h_t to decrease the error of the previous model F_{t-1} such that:

$$h_t = argmin_h \sum_{i=1}^{M} \ell_c\big(\sum_{p=1}^{P} \alpha^p \mathcal{L}_p\big(\sum_{k=1}^{t-1} \alpha_k^p h_k(x_i) + h(x_i)\big), y_i\big), \tag{3}$$

h_t can be rewritten as follows:

$$h_t = argmin_h \sum_{i=1}^{M} \ell_c\big(\sum_{p=1}^{P} \alpha^p \mathcal{L}_p\big(F_{t-1}(x_i) + h(x_i)\big), y_i\big), \tag{4}$$

where $\ell_c(F(x), y)$ is a classification loss. In other words, we look for a learner h_t able to improve the current model F_{t-1}.

In gradient boosting (Friedman, 2001), one way to learn the next weak learner is to approximate the negative gradient (residuals) of F_{t-1} by minimizing the square loss between these residuals and the weak learner predictions. We define r_t^i the residual at iteration t for the example x_i as follows:

$$r_t^i = -\frac{\partial \ell_c(F_{t-1}(x_i), y_i))}{\partial F_{t-1}(x_i)}. \tag{5}$$

In fact, from this functional gradient descent approach, we can define a greedy approximation of Eq (4) by using a regression loss ℓ_r on the residuals computed in Eq (5) with respect to the classification loss ℓ_c:

$$h_t = argmin_h \sum_{i=1}^{M} \ell_r(h(x_i), r_t^i). \tag{6}$$

All the steps of our NLB training process are summarized in Algorithm 1.

In practice, we instantiate our losses with the square loss for the regression and the logistic loss for the classification as follows:

$$\ell_c(f(x_i), y_i) = log(1 + e^{-y_i F_t(x_i)}); \quad \ell_r(f(x_i), r_t^i) = (r_t^i - f(x_i))^2.$$

The choice of the logistic loss is motivated by the need to have bounded gradients in order to avoid their exponential growth with the boosting iterations, which can happen for noisy instances. Then, according to Eq (6), the weak classifiers are updated as follows:

$$h_t = argmin_h \sum_{i=1}^{M} (h(x_i) - r_t^i)^2. \tag{7}$$

The residuals can be obtained thanks to a straight forward closed form:

$$r_t^i = \frac{-y_i}{1 + e^{y_i F_{t-1}(x_i)}}.$$

Finally, we used a *relu* activation function such that $\mathcal{L}(x) = \begin{cases} x & \text{if } x > 0, \\ 0 & \text{otherwise.} \end{cases}$

When a new weak learner h_t is added, we need to find its corresponding weights α_t^p $p \in P$. Adding a new weak learner is changing each representation and thus we also need to update α^p accordingly as follows:

$$\alpha^p = argmin_\alpha \sum_{i=1}^{M} \ell_c(\sum_{p=1}^{P} \alpha \mathcal{L}_p\big(F_{t-1}(x_i) + \alpha_t^p h_t(x_i)\big), y_i) \tag{8}$$

$$\alpha_t^p = argmin_\alpha \sum_{i=1}^{M} \ell_c(\sum_{p=1}^{P} \alpha^p \mathcal{L}_p\big(F_{t-1}(x_i) + \alpha h_t(x_i)\big), y_i) \tag{9}$$

At test time, our model learned using Algorithm 1 predicts as follows:

$$F^*(x) = sign\Big(F(x)\Big) = sign\Big(\sum_{p=1}^{P} \alpha^p \mathcal{L}_p\Big(\sum_{t=1}^{T} \alpha_t^p h_t(x)\Big)\Big).$$

4. Experiments

In this section, we compare our method, NLB, with the linear gradient boosting, GB [1]. We first give a visualization of both algorithms on a didactic dataset. We then take different

1. We used the implementation of the gradient boosting in scikit-learn.

Algorithm 1 Non-linear boosting

```
INPUT: T weak learners, $\{x_i, y_i\}_{i=1}^M$
Initialize $h_0 = 0$
Initialize $\alpha_t^p$ and $\alpha^p$
Predict $F_0(x_i) = h_0 = 0$
for $t = 1$ to $T$ do
    Compute the residuals $r_t^i = \frac{\partial \ell_t(F_{t-1}(x_i), y_i))}{\partial F_{t-1}(x)}$
    $h_t = argmin_h \sum_{i=1}^M (h(x_i) - r_t^i)^2$
    for $p = 1$ to $P$ do
        $\alpha^p = argmin_\alpha \sum_{i=1}^M \ell_c(\sum_{p=1}^P \alpha \mathcal{L}_p\big(F_{t-1}(x_i) + h_t(x_i)\big), y_i)$
        $\alpha_t^p = argmin_\alpha \sum_{i=1}^M \ell_c(\sum_{p=1}^P \alpha^p \mathcal{L}_p\big(F_{t-1}(x_i) + \alpha h_t(x_i)\big), y_i)$
    end for
    Predict $F_t(x_i)$
end for
```

imbalanced datasets from the KEEL repository (Alcalá-Fdez et al., 2011) described in Table 1. Every dataset name presents the new distribution of classes. For example, *abalone-20 vs 8-9-10* is the abalone dataset transformed to the binary format by setting the class 20 against the classes 8,9 and 10. Finally, we show different properties of NLB on a bigger dataset.

4.1. Visualisation with a toy dataset

In this experiment, we propose to evaluate the models with two different metrics. The first one is the F_1 score which is known to be relevant especially in the class imbalance problems where one needs to emphasize on the class of interest (usually, the positive class). We recall the F_β and F_1 scores to be:

$$F_\beta = (1 - \beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall}$$

,

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}$$

, where β represents the weight given to the precision (the higher, the more F_β will emphasis on the precision while lowering the value of β will intensify the importance of the recall). The second evaluation metric is the Average Precision (AP), a well-known measure in the learning to rank community. We explain this choice for two main reasons. 1) It offers a good intuition of the potential of a model regardless of the decision threshold learned. Indeed, in the class imbalance setting, the decision threshold is likely to be suboptimal. 2) It has been shown that AP makes more sense when the classes are highly skewed than other metrics such as the AUC ROC (Davis and Goadrich, 2006; Frery et al., 2017). We recall this measure:

$$AP = \frac{1}{P} \sum_{i=1}^{P} precision(k_i),$$

where P is the number of positive examples and $precision(k_i)$ is the precision when the decision threshold is set at the k_i rank of example x_i.

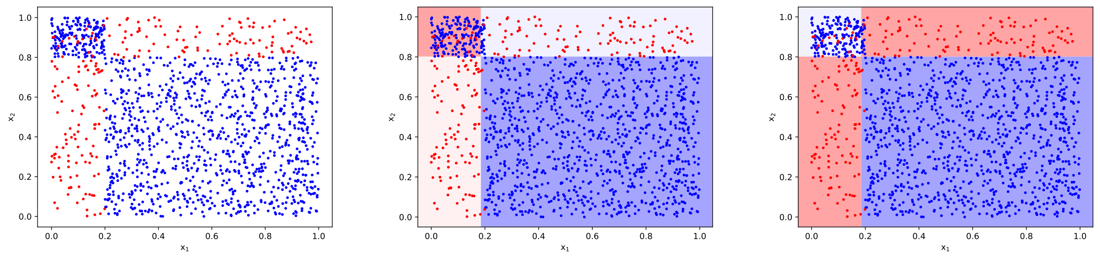


Figure 2: On the left, we present the toy dataset with two classes. The red class is in minority. The figures in the middle and on the right show the probability boundaries of GB and NLB respectively, on the test set. Blue areas show a strong probability for the examples to belong to the blue class while the red areas show a strong probability for the examples to belong to the red class. As the color disappears, the model is uncertain to which class the examples belong.

We first present an experiment on a didactic dataset in Fig 2. We use a binary imbalanced dataset (the red class is in minority) in two dimensions to highlight the main differences while training NLB and GB. The underlying concept is rather easy with a specificity on the top left corner where examples are randomly overlapping with respect to the imbalanced ratio in the dataset $\frac{\#positive}{\#totalexamples} \approx 0.1$. In this experiment, both algorithms are allowed to build two different stumps (trees with only one split) [2] and their probability boundaries (continuous scores) are used instead of the decision boundaries (binary classification) to illustrate internal decision rules on the test set. It is worth noticing that, both algorithms learn the exact same splits. However, their weighting schema is different. In fact, GB (in the middle) and NLB(on the right) build their two hypotheses naturally: first splitting the dataset vertically on x_2. Then splitting horizontally on x_1. For this second split, the only solution using the linear combination of the hypotheses is to assign more weight to the examples on the left to belong to the red class. However, this gives a higher probability for the examples in the upper left area to belong to the red class. NLB, instead, finds a representation of the hypotheses learned such as to give the highest probabilities on parts where the examples are not overlapping. At this stage of learning, GB has an $AP = 0.4476$ while NLB has $AP = 0.9088$. The best F_1 score for both algorithm is $F_1 = 0.7012$ and $F_1 = 0.8874$ for GB and NLB respectively. Also, note that the next learning steps for GB are going to be more specialized on misclassified examples and thus the risk of overfitting will be high. In fact, with decision stumps, GB is not able to reach NLB performances. Finally, we would like to point out that, in this case, the meta-learning part does not increase the complexity of the model. Indeed, the meta-learner does not create new boundaries but rather re-weights the existing areas to improve the performance on the given task and so does not increase the risk of overfitting.

2. It is clear that this toy example could be solved with only one tree more complex than a stump. However, in general, finding the right complexity is not trivial. For example, a tree with depth two would have solved our problem correctly while a tree of depth three would have overfitted.

4.2. Experiments on real datasets

Dataset	#Examples	Imbalance ratio
poker-8 vs 6	1477	0.0115
abalone-20 vs 8-9-10	1916	0.0136
winequality-red-3 vs 5	691	0.0145
winequality-white-3-9 vs 5	1482	0.0169
kr-vs-k-zero vs eight	1460	0.0185
winequality-red-8 vs 6-7	855	0.0211
winequality-white-3 vs 7	900	0.0222
abalone-17 vs 7-8-9-10	2338	0.0248
kr-vs-k-three vs eleven	2935	0.0276
yeast5	1484	0.0296
winequality-white-9 vs 4	168	0.0298
yeast-1-2-8-9 vs 7	947	0.0317
poker-9 vs 7	244	0.0328
car-vgood	1728	0.0376
glass-0-1-6 vs 5	184	0.0489
zoo-3	101	0.0495
abalone9-18	731	0.0575
glass4	214	0.0607
ecoli-0-1-4-6 vs 5	280	0.0714
vowel0	988	0.0911
yeast-0-5-6-7-9 vs 4	528	0.0966
ecoli-0-1 vs 2-3-5	244	0.0984
yeast-0-3-5-9 vs 7-8	506	0.0988
yeast-2 vs 4	514	0.0992

Table 1: Properties of the datasets used in the experiments.

We now present an extensive experimental study of our algorithm compared with GB.

Our experimental protocol generates 30 different 2/3-1/3 splits of the data using stratified sampling to obtain the training and test sets respectively. Note that we use decision trees as our base learners. We tune the models over this 2/3 split using a 3-fold cross-validation. The parameters tuned are the number of weak learners $T \in \{0, 1, ..., 100\}$, the depth of each decision tree and the learning rate. We set a maximum limit of depth 5 such as to have very weak learners.

We report the average of the metrics obtained on each test set in Table 2.

In general, NLB outperforms linear gradient boosting. Interestingly, we can see that the two metrics do not always agree on the best method. In fact, an interpretation of the average precision (which is nothing else than the area under the precision and recall curve) is that a model having a better AP has, on average, a better F_1 score across all possible decision thresholds than a model with a lower AP. Indeed, classification in imbalanced data is very domain specific (i.e. emphasizing more on the recall rather than the precision and vice versa). Instead of computing multiple β for the f_β score, we simply use the AP so as to have a global view on the model performances. The only condition is on the model that must allow probability outputs which is the case for NLB and GB.

Finally, we report in Table 3 the average number of weak learners and the average number of splits built by the trees to which we refer, as the model complexity. We see that, on average, GB builds more complex base learners and needs almost twice as many weak learners as NLB. Also note that the model complexity depends mainly on the hyperparameter of the tree depth and that as the depth increases linearly, the model complexity grows exponentially.

Dataset	NLB(AP)	GB(AP)	NLB(F1)	GB(F1)
poker-8 vs 6	**29.3 ± 19.8**	25.8 ± 31.3	**28.9 ± 24.4**	9.8 ± 19.8
abalone-20 vs 8-9-10	**27.9 ± 11.7**	20.1 ± 18.9	**20.2 ± 15.7**	19.3 ± 20.0
winequality-red-3 vs 5	8.7 ± 6.0	**11.1 ± 12.3**	**7.2 ± 14.0**	2.8 ± 7.9
winequality-white-3-9 vs 5	**23.8 ± 12.6**	14.8 ± 12.9	**25.8 ± 16.9**	14.9 ± 16.3
kr-vs-k-zero vs eight	**99.0 ± 1.5**	95.2 ± 7.0	77.1 ± 7.3	**81.5 ± 16.4**
winequality-red-8 vs 6-7	**13.1 ± 8.1**	6.8 ± 3.9	**12.8 ± 13.2**	4.3 ± 8.4
winequality-white-3 vs 7	**41.5 ± 9.5**	37.7 ± 19.2	**36.2 ± 15.0**	32.7 ± 16.5
abalone-17 vs 7-8-9-10	**28.7 ± 7.9**	21.4 ± 7.5	22.2 ± 10.2	**23.8 ± 7.6**
kr-vs-k-three vs eleven	**99.8 ± 0.6**	96.0 ± 5.1	**96.8 ± 2.4**	96.7 ± 2.8
yeast5	**67.2 ± 8.2**	62.8 ± 16.8	**67.6 ± 4.6**	62.6 ± 13.4
winequality-white-9 vs 4	**41.7 ± 35.4**	30.3 ± 34.6	**22.2 ± 35.1**	5.6 ± 15.7
yeast-1-2-8-9 vs 7	**29.9 ± 12.1**	22.2 ± 13.6	**25.4 ± 14.8**	21.2 ± 16.7
poker-9 vs 7	**35.1 ± 17.1**	25.4 ± 18.7	**24.1 ± 23.0**	15.4 ± 20.2
car-vgood	**99.9 ± 0.2**	97.3 ± 5.0	**96.4 ± 4.2**	83.2 ± 31.7
glass-0-1-6 vs 5	**71.2 ± 28.9**	65.7 ± 32.4	**56.3 ± 34.4**	36.7 ± 35.5
zoo-3	**35.3 ± 29.9**	29.4 ± 21.4	**32.2 ± 30.0**	20.4 ± 29.2
abalone9-18	**40.1 ± 7.4**	30.4 ± 9.9	**37.9 ± 6.4**	30.2 ± 11.4
glass4	**54.4 ± 16.4**	51.2 ± 22.2	46.9 ± 24.8	**54.0 ± 16.1**
ecoli-0-1-4-6 vs 5	69.9 ± 16.0	**74.6 ± 18.4**	68.9 ± 11.1	**69.2 ± 11.8**
vowel0	94.7 ± 5.2	**97.7 ± 2.1**	89.4 ± 5.8	**91.9 ± 4.5**
yeast-0-5-6-7-9 vs 4	46.8 ± 4.4	**55.3 ± 12.7**	40.3 ± 10.8	**52.2 ± 12.3**
ecoli-0-1 vs 2-3-5	**76.5 ± 11.1**	67.7 ± 11.6	**65.9 ± 12.9**	57.0 ± 8.4
yeast-0-3-5-9 vs 7-8	**42.1 ± 8.3**	36.9 ± 11.5	**29.4 ± 6.9**	29.1 ± 11.8
yeast-2 vs 4	**82.7 ± 7.4**	80.7 ± 7.4	**75.2 ± 6.5**	71.0 ± 9.6

Table 2: The Average Precision (AP) and the F1 score (F1) reported for NLB and GB.

Model	Average #Splits	Average #Weak learners
GB	22.13 ± 7.92	67.25 ± 35.55
NLB	5.08 ± 3.83	35.42 ± 39.01

Table 3: Average number of weak learner and number of splits per weak learner for GB and NLB.

While NLB shows a better convergence rate in terms of weak learners, it still needs an extra step to update the meta-learner parameters. However, since we update our parameters sequentially and only once per weak learner and per representation, the overall update time of the meta-learner is not larger than the time to train a basic neural network with one layer and T inputs (the number of weak learners).

4.3. In-depth analysis of NLB

In this section, we present two different qualitative analyses on the latent representations learned by our algorithm. An important point in NLB is to learn **different** representations of the weak models. With this in mind, we provide evidence that NLB can generate some diversity. We also show that these representations contribute in a comparable way to the final decision. Finally, we give some convergence figures comparing NLB to GB. All the following experiments are made on the MNIST dataset in the 1-vs-all conditions.

Our first analysis aims at showing that the learned representations tend to be uncorrelated when using a weak learner. For this purpose, we compute a correlation matrix C between all the representations such that $C_{nm} = \frac{cov_{nm}}{\sqrt{cov_{nn} * cov_{mm}}}$ measures the correlation between the latent representations n and m, cov is the covariance matrix computed with respect to the input weights $\{\alpha_i^m\}_{i=1}^N$ and $\{\alpha_i^n\}_{i=1}^N$ of these representations. We show, in Fig. 3, the matrix, C, for the representations obtained after convergence with the decision stumps as base learners. We can see that most of the representations tend to be uncorrelated or weakly correlated. That said, we still need to assess whether these uncorrelated representations are contributing to the final decision.

We propose to compute, for each representation p, a relative importance coefficient Ω_p by taking the absolute values of the predictions of each representation right before they are merged together with the others to form the final prediction. We average this coefficient over $\{x_i\}_{i=1}^K$ examples taken from a validation set independent from the learning sample as follows:

$$\Omega_p = \frac{1}{K} \sum_{i=1}^{K} |\alpha^p \mathcal{L}_p\big(\sum_{t=1}^{T} \alpha_t^p h_t(x_i)\big)|. \tag{10}$$

We expect for important representations a high Ω_p (*i.e.* having a high impact in the final decision) and a low Ω_p for irrelevant ones (*i.e.* having low impact in the final decision).

We consider the model learned with the decision stumps. We plot the importance coefficient Ω_p (y-axis) against the average correlation of each representation (x-axis) that we define as $\widehat{C}_p = \frac{1}{P}\sum_{i=1}^{P} C_{pi}$. This illustrates the importance of each representation in the final decision with respect to their correlation level.

Fig. 4 gives the plot for the model using the decision stumps as base learners. We see here that all the representations are involved in the final decision and that their relative importance coefficients are rather comparable. In these conditions, NLB has a clear advantage other GB. We give in Fig. 7 the performances F_1 and AP as we add more weak learners. GB not only converges slower toward its final state but it also has an optimal solution which is less efficient than for NLB. With only 15 weak learners, NLB achieves the same results as GB with 100 weak learners.

Now, as stated earlier, GB needs stronger weak learner to achieve better results in some cases and it would be fair to assess NLB in this context. In Fig. 5, we compute the same correlation matrix of the representations as previously, however, the base learner is now set to be a tree of depth 10.

We see that most of the representations are highly correlated. In this case, where $\widehat{C}_p \approx 1$, the final NLB model is comparable to a linear combination that could be built in the standard linear gradient boosting. In addition, in Fig. 6, we can see that many representations are unused in the final model and only representations with few variations were useful with this strong base learner. In general, boosting methods have a hard time to diversify complex models (hence the popular term "weak learner" used in GB) and so it also becomes hard for NLB to build diverse and useful representations of these classifiers.

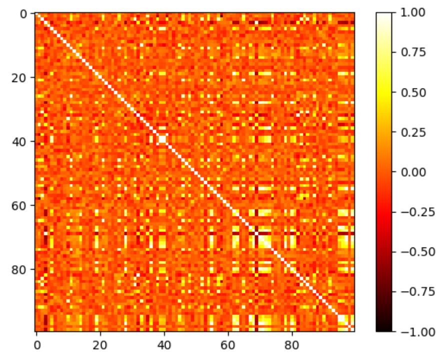


Figure 3: Correlation matrix of the representations with stumps as base learner.

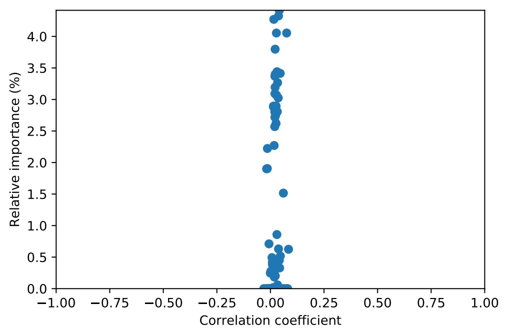


Figure 4: Relative importance of each representation with stumps as base learner.

5. Conclusion

We presented a new non-linear boosting algorithm, NLB. Our method builds different representations of the classifiers during the learning iterations of boosting. We showed several advantages using NLB:
1) A fast convergence rate in terms of weak learners. 2) NLB has a better generalization potential. Since we learn the specificities of the target concept using very weak learners, we reduce the risk of overfitting while increasing the expressiveness of the final model with the non-linear combinations. 3) Suited for class imbalance learning problems. We presented an experimental study that showed a better performance for NLB for the classification task in the imbalanced setting compared to standard linear gradient boosting. Finally, we provided an evidence that the representations learned in NLB bring diverse and relevant knowledge on the given task.

In future works, further analyses of the representations should be conducted in order to improve their collaboration and remove irrelevant ones.

References

Jesús Alcalá-Fdez, Alberto Fernández, Julián Luengo, Joaquín Derrac, Salvador García, Luciano Sánchez, and Francisco Herrera. Keel data-mining software tool: data set repos-

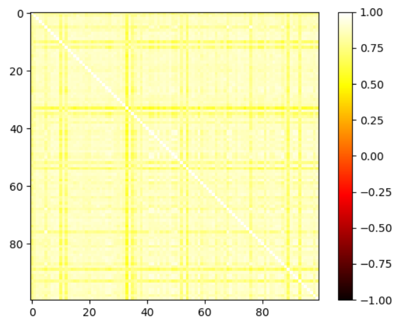


Figure 5: Correlation matrix of the representations with trees of depth 10 as base learner.

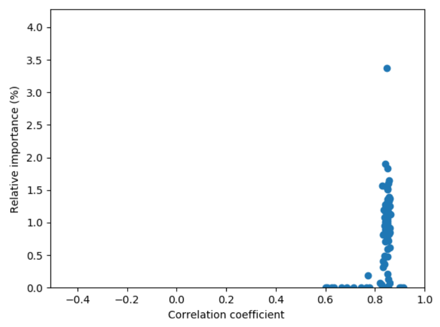
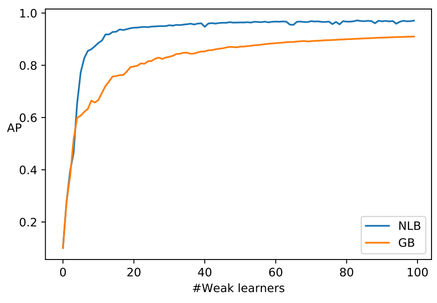


Figure 6: Relative importance of each representation with trees of depth 10 as base learner.



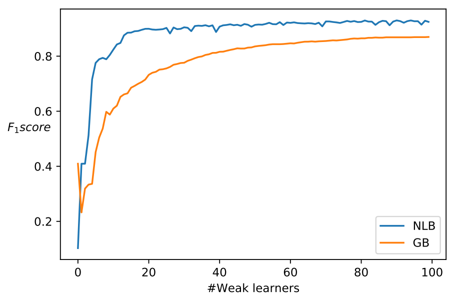


Figure 7: AP(on the left) and F_1score(on the right) for NLB and GB along the iterations of boosting.

itory, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, 17, 2011.

Carlos J Becker, Christos M Christoudias, and Pascal Fua. Non-linear domain adaptation with boosting. In *Advances in Neural Information Processing Systems*, pages 485–493, 2013.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. Boosted multi-task learning. *Machine learning*, 85(1-2): 149–173, 2011.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *JAIR*, pages 321–357, 2002. doi: 10.1613/jair.953. URL http://dx.doi.org/10.1613/jair.953.

Nitesh V. Chawla, Aleksandar Lazarevic, Lawrence O. Hall, and Kevin W. Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *ECML PKDD*, pages 107–119, 2003. doi: 10.1007/978-3-540-39804-2_12. URL http://dx.doi.org/10.1007/978-3-540-39804-2_12.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *SIGKDD*, pages 785–794. ACM, 2016.

Andrea Dal Pozzolo, Olivier Caelen, and Gianluca Bontempi. When is undersampling effective in unbalanced classification tasks? In *ECML PKDD*, pages 200–215, 2015. doi: 10.1007/978-3-319-23528-8_13. URL http://dx.doi.org/10.1007/978-3-319-23528-8_13.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. pages 233–240, 2006.

Wei Fan, Salvatore J. Stolfo, Junxin Zhang, and Philip K. Chan. Adacost: Misclassification cost-sensitive boosting. In *ICML*, pages 97–105, 1999.

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, and Liyun He-Guelton. Efficient top rank optimization with gradient boosting for supervised anomaly detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 20–35. Springer, 2017.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

João Gama and Pavel Brazdil. Cascade generalization. *Machine Learning*, 41(3): 315–343, 2000. doi: 10.1023/A:1007652114878. URL `https://doi.org/10.1023/A:1007652114878`.

Nicolas Garcã-Pedrajas, CÃĐsar Garcã-Osorio, and Colin Fyfe. Nonlinear boosting projections for ensemble construction. *Journal of Machine Learning Research*, 8(Jan):1–33, 2007.

Shizhong Han, Zibo Meng, Ahmed-Shehab Khan, and Yan Tong. Incremental boosting convolutional neural network for facial action unit recognition. In *NIPS*, pages 109–117, 2016.

Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering*, (9):1263–1284, 2008.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, 2017.

Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.

Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Bier-boosting independent embeddings robustly. In *CVPR*, pages 5189–5198, 2017.

David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992. doi: 10.1016/S0893-6080(05)80023-1. URL `https://doi.org/10.1016/S0893-6080(05)80023-1`.