HAL open science

Online Non-Linear Gradient Boosting in Multi-Latent Spaces

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, Liyun He-Guelton

To cite this version:

Jordan Frery, Amaury Habrard, Marc Sebban, Olivier Caelen, Liyun He-Guelton. Online Non-Linear Gradient Boosting in Multi-Latent Spaces. 17th International Symposium on Intelligent Data Analysis (IDA'18), Oct 2018, s-Hertogenbosch, Netherlands. hal-01889994

HAL Id: hal-01889994

https://hal.science/hal-01889994v1

Submitted on 8 Oct 2018

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

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

Online Non-Linear Gradient Boosting in Multi-Latent Spaces

Jordan Frery[1,2], Amaury Habrard[1], Marc Sebban[1], Olivier Caelen[2], and Liyun He-Guelton[2]

[1] Univ. Lyon, Univ. St-Etienne F-42000, UMR CNRS 5516, Laboratoire Hubert-Curien, France
firstname.lastname@univ-st-etienne.fr
[2] Worldline, 95870 Bezons, France
firstname.lastname@worldline.com

Abstract. Gradient Boosting is a popular ensemble method that combines linearly diverse and weak hypotheses to build a strong classifier. In this work, we propose a new Online Non-Linear gradient Boosting (ONLB) algorithm where we suggest to jointly learn different combinations of the same set of weak classifiers in order to learn the idiosyncrasies of the target concept. To expand the expressiveness of the final model, our method leverages the non linear complementarity of these combinations. We perform an experimental study showing that ONLB (i) outperforms most recent online boosting methods in both terms of convergence rate and accuracy and (ii) learns diverse and useful new latent spaces.

1 Introduction

Ensemble learning aims at combining diverse hypotheses to generate a strong classifier and has been shown to be very effective in many real life applications. Several categories of ensemble methods have been proposed in the literature, like *bagging* (e.g. random forest [1]), *stacking* [2], *cascade generalization* [3], *boosting* [4], etc. Those state of the art methods essentially differ by the way they generate diversity and combine the base hypotheses. In this paper, we focus on gradient boosting [5] which - unlike many other machine learning methods - performs an optimization in the *function* space rather than in the *parameter* space. This opens the door to the use of any loss function expanding the spectrum of applications that can be covered by this method. Moreover, the popularity of gradient boosting has been increased by recent implementations showing the scalability of the method even with billions of examples [6,7].

Despite these advantages, real world applications such as fraud detection, click prediction or face recognition are often subject to uninterrupted data flow which is completely ignored in the batch gradient boosting setting. This brings up a major concern: How to train models over always increasing volumes of data that need more memory and more storage? While big data centers can partially solve the problem, training the model from scratch each time new instances arrive remains unrealistic.

To overcome this problem, online boosting has received much attention during the past few years [8,9,10,11,12,13,14]. In these methods, the boosted model is updated after seing each example. While they can process efficiently large amount of data, their practical limitations include: (i) an edge assumption usually made on the asymptotic accuracy (*i.e.* the edge over random guessing) of the weak learners making some approaches hard to tune, (ii) the absence of a weighting scheme of the weak learners that depends on their performance and (iii) for some of them a lack of adaptiveness (despite the fact that it was a strong point of Adaboost [4]).

Moreover, all the previous online methods face another issue: they usually perform a linear combination over a finite number of learned hypotheses which may limit the expressiveness of the final model to reach complex target concepts. While the batch setting would allow us to add step by step new hypotheses and capture the complexity of the underlying problem, an online algorithm keeps the same set of weak learners all along the process. This remark prompted us to investigate the way to develop a **non linear** gradient boosting algorithm with an enhanced expressiveness. To the best of our knowledge, there is only one work specific to non-linear boosting [15] but only usable in a batch setting. This is why the main contribution of this paper takes the form of a new algorithm, called ONLB - for Online Non Linear gradient Boosting. Inspired from previous research in domain adaptation [16], boosted-multi-task learning [17] and boosting in concept drift [18], ONLB 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. ONLB is illustrated in Fig. 1. At first glance, it looks similar to boosted neural networks, as done in [19,20], where the embedding layer is learned with boosting in order to infer more diversity. However, our method aims at learning the weak hypotheses iteratively, the next weak learner trying to minimize the error made by the network restricted to the previous hypotheses (see the solid lines in Fig. 1). 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). Thanks to the non-linear function brought by the last layer to combine the different representation output, ONLB converges much faster than the other state of the art online boosting algorithms.

The paper is organized as follows: Section 2 is devoted to the presentation of the related work. Our new non-linear online gradient boosting algorithm ONLB is presented in Section 3. Section 4 is dedicated to a large experimental comparison with the state of the art methods. We conclude the paper in Section 5.

2 Related work

Online boosting methods have been developed soon after their batch counterpart. The first one introduced in [8] uses a resampling method based on a Poisson distribution and was applied in computer vision by [9] for feature selection. The-

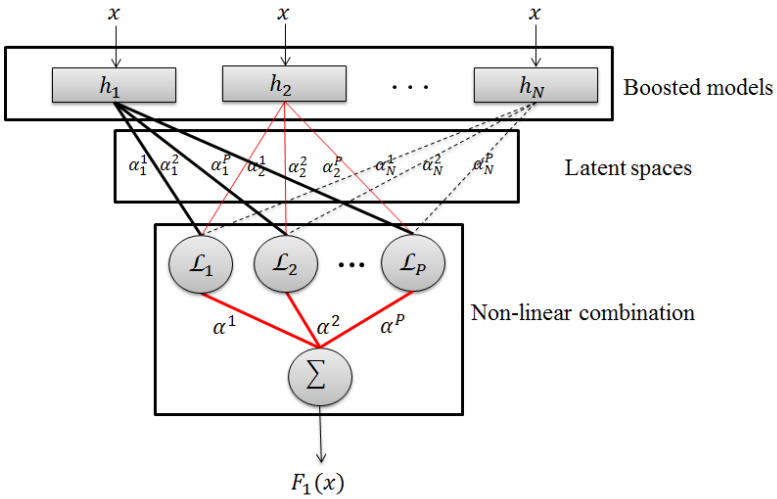

Fig. 1: Graphical representation of our Online Non-Linear gradient Boosting method: the first top layer corresponds to the learned weak classifiers; the second layer represents different linear combinations of their outputs; the bottom layer proceeds a non linear transformation of those combinations. 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.

oretical justifications were developed later in [10] where they notably discuss the number of weak learners needed in an online boosting framework. This is indeed a major concern since having too many of them could lead to predictions dominated by redundant weak learners that perform poorly. On the other hand, too few weak learners could make the boosting process itself irrelevant, as the goal is still to improve upon the performance of a simple base learner. More recently, [11] extends this previous work to propose an optimal version of boosting in terms of the number of weak learners for classification. An adaptation of this framework to multi-class online boosting was proposed in [12]. While these methods come with a solid theory, the assumption usually made on the asymptotic accuracy (*i.e.* the edge over random guessing) of the weak learners leads to two main practical limitations. The first one is the undeniable difficulty to estimate this edge without prior knowledge on the task at hand. The second comes from the fact that the edge of each weak learner might be very different depending on their own performance. And it turns out that the latter is never taken into consideration and might impact the overall performance of boosting.

Online gradient boosting was introduced by [21] allowing one to use more general loss functions but without any theoretical guarantees. Later, [13] and its extension to non smooth losses [14], propose online gradient boosting algorithms

with theoretical justifications. These are the closest approaches to ours but they do not weight the weak learners based on their own performance. Moreover, the linear aspect of these methods limit strongly their expressiveness.

Another series of related works is the use of boosting in neural network methods. Recently, neural networks were used with incremental boosting [19] to train a specific layer. In [20], the authors reused [13] to optimize and increase the diversity of their embedding layer. Our work is related in the sense that we boost a layer to build a new feature space. However, we do not aim at learning a general neural network. This layer is rather used to make connections between our different weak learners. This is why our back-propagation procedure differs by focusing only on the parameters of the weak learner to be optimize at each step.

Apart from online boosting methods, our work is also related to non-linear boosting. However, as far as we know, only [15] tackled this topic by proposing a non-linear boosting projection method where, at each iteration of boosting, they 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 for the next learner. Nonetheless, it is very expensive and unsuitable to online learning.

3 Online Non-Linear gradient Boosting

In this study, we consider a binary supervised online learning setting where at each time step $t = 1, 2, ..., T$ one receives a labeled example $(x_t, y_t) \in \mathcal{X} \times \{-1, 1\}$ where $\mathcal{X}$ is a feature space. In this setting, the learner makes a prediction $f(x_t)$, the true label y_t is then revealed and it suffers a loss $\ell(f(x_t), y_t)$.

Boosting aims at combining different weak hypotheses. In batch gradient boosting, weak learners are learned sequentially while in the online setting, they are not allowed to see all examples at once. Thus, it is not possible to simply add new models iteratively in the combination as in batch boosting. In fact, online boosting maintains a sequence of N weak online learning algorithms $\mathcal{A}_1, ..., \mathcal{A}_N$ such that each weak learner h_i is updated by $\mathcal{A}_i$ in an online fashion. Note that every $\mathcal{A}_i$ considers hypotheses from a given restricted hypothesis class $\mathcal{H}$. The final model corresponds to a weighted linear combination of the N weak learners:

$$F(x) = \sum_{i=1}^{N} \alpha_i h_i(x), \tag{1}$$

where α_i stands for the weight of the weak learner h_i.

We now present our Online Non-Linear gradient Boosting, ONLB. As shown in Fig. 1, our method maintains P different representations that correspond to different combinations of the N learned weak learners, projecting their outputs into different latent spaces. Every representation p is updated right after a weak learner 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:

$$F(x) = \sum_{p=1}^{P} \alpha^p \mathcal{L}_p\Big(\sum_{i=1}^{N} \alpha_i^p h_i(x)\Big), \tag{2}$$

where α_i^p are the weights projecting the outputs of the weak learner h_i 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: $F_k(x) = \sum_{p=1}^{P} \alpha^p \mathcal{L}_p\Big(\sum_{i=1}^{k} \alpha_i^p h_i(x)\Big)$.

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_i to decrease the error of the already learned models in F_{i-1} such that:

$$h_i = argmin_h \sum_{t=1}^{T} \ell_c\big(\sum_{p=1}^{P} \alpha^p \mathcal{L}_p\big(\sum_{k=1}^{i-1} \alpha_k^p h_k(x_t) + h(x_t)\big), y_t\big), \tag{3}$$

where $\ell_c(F(x), y)$ is a classification loss. In other words, we look for a learner h_i that improves over the learned combination, F_{i-1}.

In gradient boosting [5], one way to learn the next weak learner is to approximate the negative gradient (residuals) of F_{i-1} by minimizing the square loss between these residuals and the weak learner predictions. We define r_i^t the residual at iteration i for the example x_t as follows:

$$r_i^t = -\frac{\partial \ell_c(F_{i-1}(x_t), y_t))}{\partial F_{i-1}(x_t)}. \tag{4}$$

In fact, from this functional gradient descent approach, we can define a greedy approximation of Eq (3) by using a regression loss ℓ_r on the residuals computed in Eq (4) with respect to the classification loss ℓ_c:

$$h_i = argmin_h \sum_{t=1}^{T} \ell_r(h(x_t), r_i^t). \tag{5}$$

As stated above, when a weak learner h_i is updated, we need: (i) to update the weights α_t^p associated to this learner in each representation p and (ii) update the representation weights α^p in the final decision as follows:

$$\alpha^p := \alpha_p - \eta \frac{\partial \ell_c(F_i(x_t), y_t)}{\partial \alpha^p}; \quad \alpha_i^p := \alpha_t^p - \eta \frac{\partial \ell_c(F_i(x_t), y_t)}{\partial \alpha_i^p}.$$

All the steps of our ONLB 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:

$$\ell_c(f(x_t), y_t) = log(1 + e^{-y_t F_i(x_t)}); \quad \ell_r(f(x_t), r_i^t) = (r_i^t - f(x_t))^2.$$

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 for example. The square loss is the main loss function for regression tasks and has demonstrated superior computational and theoretical properties for the online setting [22]. Then, according to Eq (5), the weak classifiers are updated as follows:

$$h_i = argmin_h \sum_{t=1}^{T} (h(x_t) - r_i^t)^2. \tag{6}$$

Equ (6) suggests a fairly simple update of each weak learner: each weak online learning algorithm $\mathcal{A}_i$ uses a simple stochastic gradient descent with respect to one example at each step. The residuals can be obtained thanks to a straight forward closed form:

$$r_i^t = \frac{-y_t}{1 + e^{y_t F_{i-1}(x_t)}}.$$

Finally, we used a *relu* activation function such that $\mathcal{L}(x) = \begin{cases} x \text{ if } x > 0, \\ 0 \text{ otherwise.} \end{cases}$ The weights of the latent spaces α_i^p and α^p are now updated as follows:

$$\alpha_i^p := \alpha_i^p + \eta \begin{cases} \frac{y_t \alpha^p h_i(x_t)}{1+e^{y_t F_i(x_t)}} & \text{if } \alpha_i^p h_i(x_t) > 0, \\ 0 & \text{otherwise} \end{cases} ; \quad \alpha^p := \alpha^p + \eta \frac{y_t \mathcal{L}_p\big(\sum_{i=1}^{N} \alpha_i^p h_i(x_t)\big)}{1 + e^{y_t F_i(x_t)}}.$$

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

$$F^*(x) = sign\Big(F(x)\Big) = sign\Big(\sum_{p=1}^{P} \alpha^p \mathcal{L}_p\Big(\sum_{i=1}^{N} \alpha_i^p h_i(x)\Big)\Big).$$

4 Experiments

In this section, we provide an experimental evaluation of our non-linear online boosting method ONLB in terms of both quantitative and qualitative analysis. First, we perform a comparative study with different state-of-the-art online boosting algorithms on public datasets. Second, we present an analysis of the learned representations.

4.1 Classification Results

We use 10 public datasets from the UCI repository by considering binary classification problems (multi-class datasets were converted into binary problems as

Algorithm 1 Online Non-Linear gradient Boosting (ONLB)

INPUT: N online weak learners, a learning rate η and P latent spaces.
Initialize $h_0 = 0$
for $t = 1$ to T **do**
  Receive example x_t
  Predict $F_0(x_t) = h_0 = 0$
  for $i = 1$ to N **do**
    Reveal y_t the label of example x_t
    Compute the residual $r_i^t = \frac{\partial \ell_c(F_{i-1}(x_t), y_t))}{\partial F_{i-1}(x)}$
    Predict $h_i(x_t)$
    $\mathcal{A}_i$ suffers loss $\ell_r(r_i^t, h_i(x_t))$ and updates the hypothesis h_i
    for $p = 1$ to P **do**
      $\alpha^p := \alpha^p - \eta \frac{\partial \ell_c(F_i(x_t), y_t)}{\partial \alpha^p}$; $\alpha_i^p := \alpha_t^p - \eta \frac{\partial \ell_c(F_i(x_t), y_t)}{\partial \alpha_i^p}$
    end for
  end for
end for

	#examples	positives ratio	#features
Covtype	$581,012$	51.2%	54
Poker	$1,025,010$	49.88%	10
MNIST	$70,000$	49%	718
Abalone	$4,177$	49%	8
Pima	767	34.9%	8
Adult	$42,842$	23.9%	14
HIV	$6,590$	13.3%	8
w8a	64000	3%	300
Shuttle	$58,000$	21.4%	9
wine	$6,497$	20.64%	12

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

indicated in parenthesis): Poker (0 vs [1,9]), MNIST ([0,4] vs [5,9]), Wine ([3,6] vs [7,9]), Abalone ([0,9] vs [10,29]), Covtype (2 vs all), Shuttle (1 vs all), Pima, Adult, HIV, w8a. A summary of these datasets is presented in Table 1.

Our experimental setup is defined as follows. For every dataset, we apply a 3-fold cross validation. For tuning the hyper-parameters, we perform in each fold a progressive validation [23] on the training set as proposed in [11]: This validation process uses every new example to evaluate the model and then use it for training. Note that we simulate the online learning setting by giving the examples according to a random order to the algorithm. We train different models in parallel with respect to their hyper-parameter values (i.e. the number of weak learners N, the learning rate η and γ the weak learner edge) and we select the one achieving the lowest progressive validation error. The selected model is then evaluated on the test set.

We compare our method to different online boosting algorithms from current state-of-the-art: the four algorithms online.BBM, Adaboost.OL, Adaboost.OL.W, OGB from [11,13] and streamBoost from [14][3]. For all the algorithms, we choose as a relatively weak classifier a neural network with one hidden layer and two units that we update in an online learning fashion using stochastic gradient descent. We report the classification error obtained for each algorithm in Table 2.

ONLB achieves competitive results with the state of the art online boosting methods and even outperforms them on most datasets. In some cases, such as for MNIST or Poker, we clearly see that, while using much more weak learners (see Fig. 2), the other methods were not able to capture the target concept as much as ONLB did. Note that, a mandatory condition in our experiments was $T > 1$ such that the boosting takes part in the learning process but in some cases, the online boosting algorithms were not able to do better than the baseline on the test set. For example, on the Adult database, only ONLB and OGB achieved an average error lower than the base learner.

In Table 3, we present the average number of weak learners chosen with respect to the progressive validation process for each model. While being an online linear boosting algorithm, online.BBM achieves its performances with a significantly smaller number of weak learners compared to the other linear boosting methods. As mentioned in [11], this algorithm is optimal in the sense that no online linear boosting algorithm can achieve the same error rate with fewer weak learners or examples asymptotically. That being said, ONLB algorithm achieves, on average, better performances with more than twice less weak learners than online.BBM.

Finally, in Fig. 2, we plot the convergence curves with respect to the increasing number of examples used for two datasets: MNIST and Abalone. For all algorithms, each curve corresponds to the evolution of the error rate according to the progressive validation error measured during training. We observe that ONLB still achieves the best convergence rate for both datasets. A similar behavior has been observed for the other datasets and exhibits the nice fast convergence property of our algorithm which needs less weak learners to converge to its optimum.

4.2 Analysis of the learned multi-latent representations

In this section, we present two different qualitative analyses on the latent representations learned by our algorithm. First, we show that given a sufficiently large number of weak base learners, the representations obtained tend to be rather uncorrelated. This provides an evidence that ONLB can generate some diversity. Then, we show that these representations contribute in a comparable way to the final decision. For our study, we use the following setup. We consider a model with 100 representations (*i.e.* $P = 100$). We use two base learners: a relatively

[3] We used the implementations available in Vowpal Wabbit and re-implemented the streamBoost and OGB algorithms.

Table 2: Error rate reported for different online boosting algorithms.

Dataset	Base learner	ONLB	online.BBM	Adaboost.OL	Adaboost.OL.W	OGB	streamBoost
Covtype	0.2401	**0.2057**	0.2242	0.2273	0.2313	0.2264	0.2128
Poker	0.4182	**0.0497**	0.2375	0.1234	0.0953	0.3880	0.2668
MNIST	0.1105	**0.0561**	0.1029	0.1557	0.0830	0.1139	0.0655
Abalone	0.2673	0.2523	0.2831	**0.2487**	0.2531	0.2669	0.2720
Pima	0.2992	**0.2795**	0.2913	0.2952	0.2835	0.2874	0.2953
Adult	0.1523	**0.1465**	0.1530	0.1530	0.1526	0.1476	0.1586
HIV	0.1986	0.1393	**0.1273**	0.1360	0.1291	0.1540	0.1526
Shuttle	0.0211	**0.0024**	0.0173	0.0061	0.0058	0.0133	0.0050
w8a	0.0189	0.0148	0.0158	**0.0146**	0.0167	0.0178	0.0155
wine	0.1979	**0.1687**	0.1921	0.1931	0.1931	0.1743	0.1833

Table 3: Average number of weak learners (N) selected by progressive validation.

Dataset	ONLB	online.BBM	Adaboost.OL	Adaboost.OL.W	OGB	streamBoost
Covtype	6	60	79	59	282	63
Poker	52	222	348	311	320	285
MNIST	14	66	147	207	431	131
Abalone	5	6	12	3	166	8
Pima	65	64	109	141	437	174
Adult	13	6	18	17	161	119
HIV	6	6	94	188	32	16
Shuttle	30	43	243	108	121	159
w8a	4	7	54	42	132	40
wine	5	8	112	91	97	118
Average	20	49	121	116	218	111

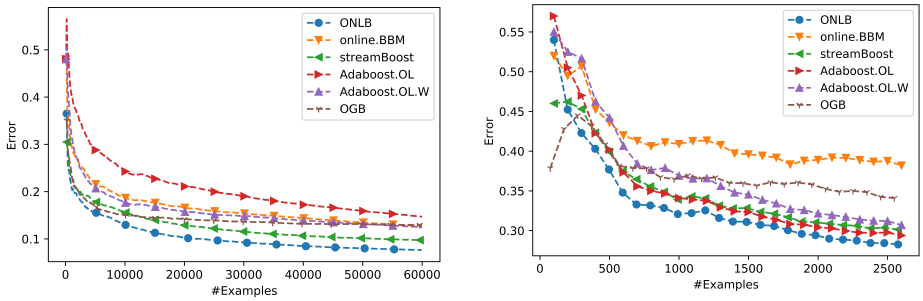

Fig. 2: Progressive validation error with respect to the learning examples for MNIST on the left and Abalone on the right.

weak neural network with one hidden layer composed of 2 units (2-NN) and a stronger learner consisting of a neural network with 500 units in its unique hidden layer (500-NN). All representation weights are initialized following a uniform distribution such that the different representations are highly uncorrelated. We consider one training file of a fold of the MNIST dataset used above for learning.

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 C matrix for the latent space representations obtained after convergence with the 2-NN base learners. We can see that most of the representations tend to be uncorrelated or weakly correlated. In contrast, Fig. 4 presents the C matrix using the 500-NN base learners. We see here that most of the representations are highly correlated. This expriment shows that by using sufficiently weak base learners, we are able to learn diverse and uncorrelated representations.

In our second analysis, we want to confirm that the uncorrelated latent representations are informative enough to contribute in a comparable way to the final strong model. We propose to compute, for each representation p, a relative importance coefficient Ω_p by taking the absolute values of the predictions of p right before they are merged together with the other representation outputs to form the final prediction. We average this coefficient over $\{x_t\}_{t=1}^K$ examples taken from a validation set independent from the learning sample as follows:

$$\Omega_p = \frac{1}{K} \sum_{t=1}^{K} |\alpha^p \mathcal{L}_p\big(\sum_{i=1}^{N} \alpha_i^p h_i(x_t)\big)|. \tag{7}$$

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 then the models learned with the 2-NN and 500-NN base learners as previously. For each model, 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. 5 gives the plot for the model using the 2-NN base learners. We see here that all the representations are involved in the final decision and that their relative importance coefficients are rather comparable. This is in opposition to the plot of Fig. 6 that provides the results for the model using the 500-NN base learners. First, we see that many representations are not used in the final decision and these correspond to the ones that are uncorrelated. In fact, representations involved in the final decision are the ones that are all highly correlated with an average correlation coefficient around 0.75. Clearly, since these representations have a high correlation level, actually only one representation is really useful at the end. But note that this representation can in fact be learned by a standard linear gradient boosting.

From this experiment, we see that complex models are hard to diversify in online boosting. Moreover, tuning their hyperparameters is harder making the probability of overfitting higher and they require a significant larger amount of training time which makes such complex models useless for online boosting.

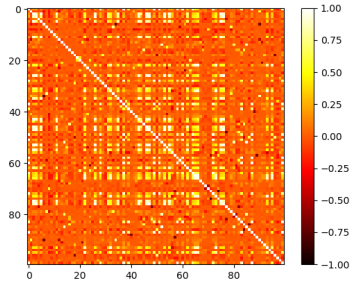

Fig. 3: Correlation matrix of the representations with 2-NN learners.

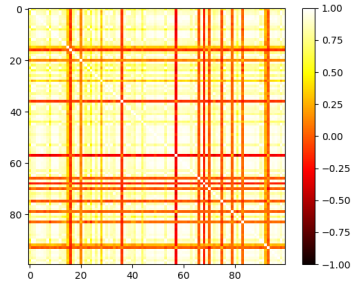

Fig. 4: Correlation matrix of the representations with the 500-NN learners.

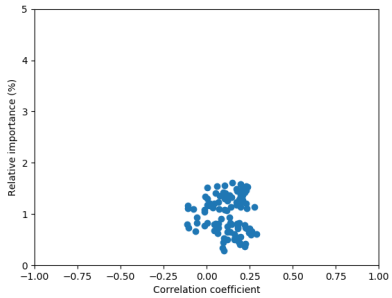

Fig. 5: Importance of each latent representation with the 2-NN learners.

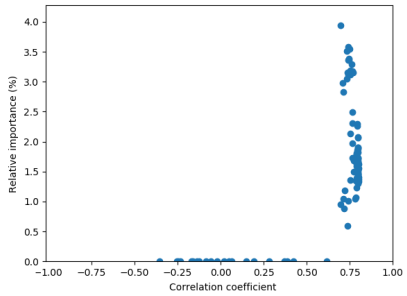

Fig. 6: Importance of each latent representation with the 500-NN learners.

5 Conclusion

In this paper, we presented a new Online Non-Linear Boosting algorithm. In this method, we combine different representations of the same set of weak classifiers to produce a non-linearly boosted model in order to learn the idiosyncrasies of the target concept. Our experimental results showed a general improvement over current state of the art online boosting methods. Additionally, the non-linear architecture of the model allows the method to use less weak learners and to obtain faster convergence in terms of examples. Our approach has also the interesting property to produce efficiently diverse latent spaces contributing actively to the model predictions. This property makes our model adaptive by giving more importance to the best current representations.

Perspectives of this work include adapting our method to the multi-class setting, to study the impact of delayed feedback (i.e. labels arriving only after some time delay) and to investigate possible adaptations for transfer learning and continuous learning in the online setting.

References

1. Breiman, L.: Random forests. Machine learning **45**(1) (2001) 5–32
2. Wolpert, D.H.: Stacked generalization. Neural Networks **5**(2) (1992) 241–259
3. Gama, J., Brazdil, P.: Cascade generalization. Machine Learning **41**(3) (2000) 315–343
4. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences **55**(1) (1997) 119–139
5. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics (2001) 1189–1232
6. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: SIGKDD, ACM (2016) 785–794
7. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.Y.: Lightgbm: A highly efficient gradient boosting decision tree. In: NIPS. (2017)
8. Oza, N.C.: Online bagging and boosting. In: Systems, man and cybernetics, 2005 IEEE international conference on. Volume 3., IEEE (2005) 2340–2345
9. Grabner, H., Bischof, H.: On-line boosting and vision. In: CVPR. Volume 1., IEEE (2006) 260–267
10. Chen, S.T., Lin, H.T., Lu, C.J.: An online boosting algorithm with theoretical justifications. In: ICML. (2012)
11. Beygelzimer, A., Kale, S., Luo, H.: Optimal and adaptive algorithms for online boosting. In: ICML. (2015)
12. Jung, Y.H., Goetz, J., Tewari, A.: Online multiclass boosting. In: Advances in Neural Information Processing Systems. (2017) 920–929
13. Beygelzimer, A., Hazan, E., Kale, S., Luo, H.: Online gradient boosting. In: Advances in Neural Information Processing Systems. (2015) 2458–2466
14. Hu, H., Sun, W., Venkatraman, A., Hebert, M., Bagnell, J.A.: Gradient boosting on stochastic data streams. In: AISTATS. (2017) 595–603
15. García-Pedrajas, N., García-Osorio, C., Fyfe, C.: Nonlinear boosting projections for ensemble construction. Journal of Machine Learning Research **8**(Jan) (2007) 1–33
16. Becker, C.J., Christoudias, C.M., Fua, P.: Non-linear domain adaptation with boosting. In: Advances in Neural Information Processing Systems. (2013) 485–493
17. Chapelle, O., Shivaswamy, P., Vadrevu, S., Weinberger, K., Zhang, Y., Tseng, B.: Boosted multi-task learning. Machine learning **85**(1-2) (2011) 149–173
18. Scholz, M., Klinkenberg, R.: Boosting classifiers for drifting concepts. Intelligent Data Analysis **11**(1) (2007) 3–28
19. Han, S., Meng, Z., Khan, A.S., Tong, Y.: Incremental boosting convolutional neural network for facial action unit recognition. In: NIPS. (2016) 109–117
20. Opitz, M., Waltner, G., Possegger, H., Bischof, H.: Bier-boosting independent embeddings robustly. In: CVPR. (2017) 5189–5198
21. Leistner, C., Saffari, A., Roth, P., Bischof, H.: On robustness of on-line boosting - a competitive study. In: 3rd ICCV Workshop on On-line Computer Vision. (2009)
22. Gao, W., Jin, R., Zhu, S., Zhou, Z.H.: One-pass auc optimization. In: International Conference on Machine Learning. (2013) 906–914
23. Blum, A., Kalai, A., Langford, J.: Beating the hold-out: Bounds for k-fold and progressive cross-validation. In: COLT, ACM (1999) 203–208