HAL
open science

Efficient top rank optimization with gradient boosting for supervised anomaly detection

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. Efficient top rank optimization with gradient boosting for supervised anomaly detection. ECML PKDD 2017 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, Sep 2017, Skopje, Macedonia. hal-01611346v1

HAL Id: hal-01611346

https://hal.science/hal-01611346v1

Submitted on 5 Oct 2017 (v1), last revised 17 Oct 2017 (v2)

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.

Efficient top rank optimization with gradient boosting for supervised anomaly detection

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. In this paper we address the anomaly detection problem in a supervised setting where positive examples might be very sparse. We tackle this task with a learning to rank strategy by optimizing a differentiable smoothed surrogate of the so-called *Average Precision* (AP). Despite its non-convexity, we show how to use it efficiently in a stochastic gradient boosting framework. We show that using *AP* is much better to optimize the top rank alerts than the state of the art measures. We demonstrate on anomaly detection tasks that the interest of our method is even reinforced in highly unbalanced scenarios.

1 Introduction

Anomaly detection in DNA sequences, credit card transactions or cyber security are some illustrations of supervised learning settings where data is often highly unbalanced (i.e. a few anomalies versus a huge amount of genuine/normal data). A naive approach to tackle this binary problem would consist in applying a standard classification, such as SVM, boosting or logistic regression, using classic margin-based surrogate loss functions, like the hinge loss, the exponential loss or the logistic loss. However, since abnormal instances are often very sparse in the feature space using such algorithms cannot be directly appropriate [1]. There exist several methods to get rid of the issues due to unbalanced datasets. The most famous are sampling-based strategies, either by undersampling or oversampling the data [2,3]. The former aims at removing instance from the majority class while the latter creates synthetic data from the minority class. Other hybrid methods such as SMOTEBoost [4], RUSBoost [5] and Adacost [6] combine a learning algorithm with a sampling or cost sensitive method. However, it turns out that these approaches have been shown to be hard to use when facing highly unbalanced situations [7] leading to either insufficient generated diversity (by oversampling) or too drastic reduction of the dataset size (by undersampling). In addition, sampling methods induce a bias in the posterior probabilities [8,9].

On the other hand, it is worth noticing that a peculiarity of the use cases mentioned above is the need to resort to a (often limited) number of human

experts to assess the potential anomalies found by the learned model. Actually, our contribution stands in a context where the number of false positives (FP) may be significantly larger than the false negative (FN) due to the high class imbalance and where the impact of FP is very penalizing. For example, in fraud detection for credit card transactions, it is out of the question to automatically block a credit card without the expert approval (which may risk the confidence of customers having their credit card falsely blocked). In this context, the goal of the automatic system is more to give the shortest list of alerts preventing the expert from going through thousands of transactions. In other words, one aims at maximizing the number of true positives in the top rank alerts (i.e. the so-called *precision*) rather than discriminating between abnormal and normal cases.

This is the reason why we tackle in this paper the supervised anomaly detection task with a *learning to rank* approach. This strategy has gained a lot of interest in the information retrieval community [10]. Given a query, the goal is to give the most relevant links to the user in a small set of top ranked items. It turns out that apart the notion of query, the anomaly detection task can relate to this setting aiming at finding the anomalies with the highest precision without giving too many genuine examples to the experts.

In such settings, different machine learning algorithms have been efficiently used such as SVMs (e.g. SVM-Rank [11], SVM-AP [12]) or ensemble methods (e.g. random forest [13], boosting [14]). It turns out that gradient boosting has shown to be a powerful method on real life datasets to address learning to rank problems [15]. Its popularity comes from two main features: (i) it performs the optimization in function space [16] (rather than in parameter space) which makes the use of custom loss functions much easier; (ii) boosting focuses step by step on difficult examples that gives a nice strategy to deal with unbalanced datasets by strengthening the impact of the positive class. In order to be efficient in learning to rank problems, gradient boosting needs to be fed with a loss function leading to a good precision in the top ranked examples.

In the literature, many approaches resort to pairwise loss functions [17,18,19], typically checking that every positive example is ranked before any negative instance. Note that all those methods implicitly optimize the area under the ROC curve. Therefore they aim at minimizing the number of incorrectly ranked pairs but do not directly optimize the precision of top ranked items as shown in [20]. To overcome this issue, recent works in learning to rank suggested to optimize other criteria like the Average Precision (AP) or the Normalized Discounted Cumulative Gain (NDCG) such as in Adarank [21], LambdaMART [22] or LambdaRank [23]. It has been shown that both AP and $NDCG$ are much more suited for enhancing ranking methods. However, due to the non convexity and non differentiability of those criteria, the previous methods rather work on standard surrogate convex objective functions (such as the pairwise cross-entropy or the exponential loss) and take into account the AP and $NDCG$ in the form of weighting coefficients only. In other words, the gradients are not computed as derivatives of AP and $NDCG$. Therefore, used in this way, these

criteria only tend to guide the optimization process in the right direction. We claim here that there is room for doing much better and directly considering the analytical expressions of those criteria in a gradient boosting method.

In this paper, our contribution is three-fold: (i) focusing on AP, we show how to optimize a loss function based on a surrogate of this criterion; (ii) unlike the state of the art learning to rank methods requiring a quadratic complexity to minimize the ranking measures, we show that AP can be handled linearly in gradient boosting without penalizing the quality of the solution; (iii) compared to the state of the art, we show that our method allows us to highly improve the quality of the top-ranked items. We even show that this advantage is much larger when the imbalance of the datasets is very important. This is a particularly interesting feature when addressing anomaly detection problems where the positive examples are very sparse.

The rest of this paper is organized as follows : In Section 2 we first introduce our notations, then describe our performance measures and present an approximation to AP. We then describe our method in a boosting framework and define a more suitable smoothed AP as the loss function in Section 3. We demonstrate the effectiveness of our work in the experiments section where we compare several state of the art machine learning models in Section 4.

2 Evaluation criteria and related work

We consider a binary supervised learning setting with a training set $\mathcal{S} = \{z_i = (x_i, y_i)\}_{i=1}^M$ composed of M labeled data, where $x_i \in \mathcal{X}$ is a feature vector and $y_i \in \{-1, 1\}$ is the label. In unbalanced scenarios, $y = 1$ often describes the minority (positive) class while $y = -1$ represents the majority (negative) class. Let P (resp. N) be the number of positive (resp. negative) examples such that $P + N = M$. We also define $\mathcal{S}^+ = \{z_i^+ = (x_i^+, y_i^+) | y_i = +1\}_{i=1}^P$ and $\mathcal{S}^- = \{z_i^- = (x_i^-, y_i^-) | y_i = -1\}_{i=1}^N$ where $\mathcal{S}^+ \cup \mathcal{S}^- = \mathcal{S}$. We assume that the training data $z_i = (x_i, y_i)$ is independently and identically distributed according to an unknown joint distribution $\mathcal{D}_\mathcal{Z}$ over $\mathcal{Z} = \mathcal{X} \times \{-1, 1\}$.

In this work, we aim at learning from S a function (or hypothesis) $f : \mathcal{X} \to \mathbb{R}$ that gives a real value to any new $x \in \mathcal{X}$. Assessing the quality of f requires the use of an evaluation criterion. It is worth noticing that most of the criteria are based on the true positive TP, true negative TN, false positive FP and false negative FN quantities. For example, the accuracy is defined as $\frac{TP+TN}{M}$. It is known that optimizing the accuracy is NP-hard due to the non convexity and non differentiability of TP and TN. Therefore, classification algorithms resort to surrogates like the hinge loss, the logistic loss or the exponential loss which are convex functions used in Support Vector Machines, logistic regression and boosting, respectively. However, when S is highly unbalanced, optimizing such losses may lead to a classifier which always predict the negative class. A solution to overcome this issue may consist in addressing unbalanced scenario from a *learning to rank* point of view. Rather that discriminating examples belonging

to the positive and negative classes, we rather aim at ranking the data with a maximal number of TP in the top ranked examples.

In this context, two measures are well used in the literature: the *pairwise* $AUCROC$ measure and the *listwise* average precision AP. From a statistical point of view, the $AUCROC$ represents the probability that a classifier ranks a randomly drawn positive instance higher than a randomly chosen negative one. The expression of this measure is equivalent to the Wilcoxon-Mann-Whitney statistic [24]:

$$AUCROC = \frac{1}{PN} \sum_{i=1}^{P} \sum_{j=1}^{N} \mathrm{I}_{0.5}(f(x_i^+) - f(x_j^-)), \tag{1}$$

where $\mathrm{I}_{0.5}$, is a special indicator function that yields 1 if $f(x_i^+) - f(x_j^-) > 0$, 0.5 if $f(x_i^+) - f(x_j^-) = 0$ and 0 otherwise. In the following we will use the classic indicator function $\mathrm{I}(*)$ that yields 1 if $*$ is true, 0 otherwise.

$1-AUCROC$ has been exploited in Rankboost algorithm [17] as an objective function where the authors use the exponential as a surrogate to the indicator function. Let $\ell_{roc}(z_i, f) = \frac{1}{N} \sum_{j=1}^{N} e^{(f(z_j^-) - f(z_i^+))}$ be the loss suffered by f at z_i. We get the following upper bound on $1 - AUCROC$:

$$1 - AUCROC \leq \frac{1}{P} \sum_{i=1}^{P} \frac{1}{N} \sum_{j=1}^{N} e^{(f(z_j^-) - f(z_i^+))} = \frac{1}{P} \sum_{i=1}^{P} \ell_{roc}(z_i, f) = \mathbb{E}_{z_i \in S^+} \ell_{roc}(z_i, f) \tag{2}$$

We can notice that this objective is a pairwise function inducing an algorithmic complexity $\mathcal{O}(PN)$. Moreover, as illustrated later in this section and shown in [20], ℓ_{roc} is not well suited to maximize the precision in the top ranked items.

A better strategy consists in using an alternative criterion based on the average precision AP, defined as follows:

$$AP = \frac{1}{P} \sum_{i=1}^{P} p(k_i), \tag{3}$$

where $p(k_i)$ is the precision with respect to the rank k_i of the ith positive example. Since the rank depends on the outputs of the model f, we get:

$$p(k_i) = \frac{1}{k_i} \sum_{j=1}^{P} \mathrm{I}(f(x_i^+) \leq f(x_j^+)) \tag{4}$$

with

$$k_i = \sum_{j=1}^{M} \mathrm{I}(f(x_i^+) \leq f(x_j)). \tag{5}$$

Plugging Eq.(4) and Eq.(5) in Eq.(3) we get:

Fig. 1: Two rankings (with two positives and eight negatives examples) ordered from the highest score (at the top) to the lowest. On the left, we get $AUCROC = 0.63$ and $AP = 0.33$. On the right, $AUCROC = 0.56$ and $AP = 0.38$. Therefore, the two criteria disagree on the best ranked list.

$$AP = \frac{1}{P}\sum_{i=1}^{P}\frac{1}{\sum_{j=1}^{M}\mathrm{I}(f(x_i^+) \leq f(x_j))}\sum_{j=1}^{M}\mathrm{I}(y_j = 1)\mathrm{I}(f(x_i^+) \leq f(x_j^+)). \tag{6}$$

AP has been used in recent papers to enhance learning to rank algorithms.

In [20,23], the authors introduce a new objective function, called LambdaRank, which can be used with different criteria, including AP. This function depends on the criterion of interest without requiring to compute the derivatives of that measure. This specificity allows them to bypass the issues due to the non differentiability of the criterion. The objective function takes the following form:

$$\frac{1}{N}\sum_{i=1}^{P}\ell_{\lambda Rank}(z_i^+, f) \tag{7}$$

with $\ell_{\lambda Rank}(z_i^+, f) = \frac{1}{N}\sum_{j=1}^{N}\log(1+e^{-(f(x_i^+)-f(x_j^-))})|AP_{ij}|$ the loss suffered by f at z_i. Here, $|AP_{ij}|$ is the absolute difference in AP when one swaps, in the ranking, example x_i with x_j. LambdaMART [22] made use of LambdaRank in a gradient boosting method and got good results as reported in [15]. However, it is worth noticing that in this algorithm, the analytical expression of AP as defined in Eq.(6) is not involved in the calculation of the gradient. $|AP_{ij}|$ can be viewed as a weighting coefficient which hopefully tends to guide the optimization process towards a good solution. One objective of this paper is to directly use AP in the algorithm and therefore to use the same criterion at both training and test time.

Let us before focus on the effect of $AUCROC$ and AP in terms of quality of top ranked items. Figure 1 compares these criteria in two different situations according to the location of two positive (in dark color) and eight negative (in light color) examples that are ordered according to their predicted scores (highest

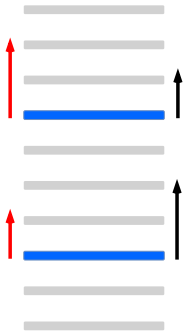

Fig. 2: Comparison of the emphasis given by AP (arrows on the left) and the emphasis given $AUCROC$ (arrows on the right) [20]. One can compare this emphasis to the intensity of gradient w.r.t the examples if AP and $AUCROC$ were continuous functions.

score at the top). The key point of this figure is to show that $AUCROC$ and AP disagree on which list is the best. $AUCROC$ prefers the list on the left because the positive examples are rather well ranked even though the first three items are negative. Therefore, we can note that this criterion is very relevant if we are interested in classifying examples into two classes, for example, the classifier being based on a threshold (likely after the fifth rank, here) splitting the items into two parts. AP is in favor of the list on the right because it prefers to champion the top list accepting to pay the prize to miss some positives. This criterion is thus very relevant to deal with anomaly and fraud detection where the goal is to provide the shortest list of alerts (here, typically the first two items) with the largest precision.

Figure 2 (inspired from [20]) illustrates graphically how the emphasis is done while computing gradients from pairwise loss function such as $AUCROC$ (black arrows on the right) or a listwise loss function such as AP (red arrows on the left) respectively. We can notice that a learning algorithm optimizing the $AUCROC$ would champion first the worst positive to get a good classifier (w.r.t. an appropriate threshold) while the AP would promote first the best positive to get a good top rank.

The previous analysis shows the advantage of optimizing AP in a learning to rank algorithm. This is the objective of the next section where we introduce a differentiable expression of AP in a gradient boosting algorithm.

3 Stochastic gradient boosting with AP

In this section, we present the stochastic gradient boosting framework as introduced in [25]. Then we instantiate the loss function in two different ways: first, we introduce a differentiable version of AP using the sigmoid function. Then, in order to reduce the algorithmic complexity, we suggest to use a rough approx-

imation based on the exponential function. We show that this second strategy allows us not only to drastically reduce the complexity but also, to get similar or even better results than the sigmoid-based loss. We give some explanations about this behavior at the end of the section.

3.1 Stochastic gradient boosting

Like other boosting methods, gradient boosting is based on a sequential and adaptive learning over weak learners that are linearly combined. However, instead of setting a weight for every example, gradient boosting builds each new weak learner on the residuals of the previous linear combination. We can see gradient boosting as gradient descent in functional space. The linear combination at step t is defined as follows:

$$f_t(x) = f_{t-1}(x) + \gamma_t h_t(x),$$

with $h_t \in \mathcal{H}$ an hypothesis belonging to a class of models $\mathcal{H}$ (typically, regression trees) and γ_t the weight underlying the performance of h_t in the linear combination. Residuals are defined by the negative gradients of the loss function computed w.r.t. the previous linear combination of weak learners:

$$g_t(x) = -\Big[\frac{\partial \ell(z_i, f_{t-1}(x_i))}{\partial f_{t-1}(x_i)}\Big], i = 1 \dots M.$$

As in standard boosting, hard examples get more importance along the iterations of gradient boosting. Note that a mini-batch strategy is usually used to speed-up the procedure by randomly selecting a proportion $\lambda \in [0, 1]$ of examples at each iteration. Additionally, this stochastic approach allows us to avoid falling in a local optima. A generic version of the stochastic gradient boosting is presented in Algorithm 1.

Algorithm 1 Stochastic gradient boosting

INPUT: a training set $S = \{z_i = (x_i, y_i)\}_{i=1}^{M}$, a parameter $\lambda \in [0, 1]$, a weak learner
Require: Initialize $f_0 = argmin_h \ell(y, h)$
for $t = 1$ to T **do**
Select randomly from S a set $S' = \{x_i, y_i\}_{i=1}^{\lambda M}$

$$g_t(x) = -\Big[\frac{\partial \ell(z, f_{t-1}(x))}{\partial f_{t-1}(x)}\Big], \forall z = (x, y) \in S' \tag{8}$$

Fit a weak classifier (e.g. a regression tree) $h_t(x)$ to predict the targets $g_t(x)$
Find $\gamma_t = argmin_\gamma \ell(z, f_{t-1}(x) + \gamma h_t(x))$
Update $f_t(x)$ such that $f_t(x) = f_{t-1}(x) + \gamma_t h_t(x)$
end for

The key step of this algorithm takes place in Eq. (8). It requires the definition of a differentiable loss function with its associated gradients. Unlike the state of

the art ranking methods which make use of gradient boosting, we aim at directly optimizing in the loss function ℓ a surrogate of AP.

3.2 Sigmoid-based surrogate of AP

To define a loss function ℓ based on AP, we need to transform the non differentiable Eq.(6) into an expression for which one will be able to compute gradients on AP. Therefore, we need to get rid of the indicator function. A standard way consists in replacing $\mathrm{I}(f(x_i) \leq f(x_j))$ by the sigmoid function :

$$\mathrm{I}(f(x_i) \leq f(x_j)) \approx \frac{1}{1+e^{-\alpha(f(x_j)-f(x_i))}} = \sigma(f(x_j) - f(x_i))$$

with α a smoothing parameter. As α grows the approximation gets closer to the true AP. Considering that $\sum_{j=1}^{M} \mathrm{I}(y_j = 1) = P$, we get the following differentiable surrogate of AP:

$$\begin{aligned}
\hat{AP}_{sig} &= \frac{1}{P}\sum_{i=1}^{P} \frac{1}{\sum_{j=1}^{M} \frac{1}{1+e^{-\alpha(f(x_j)-f(x_i^+))}}} \sum_{j=1}^{P} \frac{1}{1+e^{-\alpha(f(x_j^+)-f(x_i^+))}} \\
&= \frac{1}{P}\sum_{i=1}^{P} \frac{\sum_{j=1}^{P} \sigma(f(x_j^+) - f(x_i^+))}{\sum_{h=1}^{M} \sigma(f(x_h) - f(x_i^+))} \\
&= \frac{1}{P}\sum_{i=1}^{P} \hat{p}(k_i) \approx \frac{1}{P}\sum_{i=1}^{P} p(k_i). \qquad (9)
\end{aligned}$$

From $\hat{AP}_{sig}$, we get the following objective function:

$$1 - \hat{AP}_{sig} = \mathbb{E}_{z_i \in S^+} \ell_{ap}^{sig}(z_i, f),$$

where $\ell_{ap}^{sig}(z_i, f) = 1 - \hat{p}(k_i)$ is the loss suffered by f in terms of precision at z_i (let us remind that k_i is the rank (predicted by f) of the ith positive example z_i). In fact, we can simply rewrite our objective function as:

$$1 - \hat{AP}_{sig} = \frac{1}{P}\sum_{i=1}^{P} \frac{\sum_{j=1}^{N} \sigma(f(x_j^-) - f(x_i^+))}{\sum_{h=1}^{M} \sigma(f(x_h) - f(x_i^+))}$$

For the sake of simplicity, let us use the following notations:
$\sigma(f(x_j) - f(x_i)) = \sigma_{ji}$ and we have

$$\frac{\partial \sigma_{ji}}{\partial f_t(x_j)} = -\sigma_{ji}(1-\sigma_{ji}) = -\sigma'_{ji},$$

$$\frac{\partial \sigma_{ji}}{\partial f_t(x_i)} = \sigma_{ji}(1-\sigma_{ji}) = \sigma'_{ji}.$$

The gradient w.r.t $f_t(x_p^+)$ or $f_t(x_p^-)$, for positive and negative examples respectively, are given by:

$$\begin{aligned}\frac{\partial(1-\hat{AP}_{sig})}{\partial f_t(x_p^+)} =& \frac{\partial(1-\hat{AP}_{sig})}{\partial \sigma_{jp}}\frac{\partial \sigma_{jp}}{\partial f_t(x_p^+)} + \frac{\partial(1-\hat{AP}_{sig})}{\partial \sigma_{pi}}\frac{\partial \sigma_{pi}}{\partial f_t(x_p^+)} \\ =& \sum_{j=1}^{P} \frac{(\sigma'_{jp}\sum_{h=1}^{M}\sigma_{hp} - \sigma_{jp}\sum_{h=1}^{N}\sigma'_{hp})}{(\sum_{h=1}^{N}\sigma_{hp})^2} \\ &+ \sum_{i=1}^{P} \frac{(\sigma'_{pi}\sum_{h=1}^{N}\sigma_{hi} - \sigma_{jp}\sigma'_{pi})}{(\sum_{h=1}^{N}\sigma_{hi})^2},\end{aligned} \tag{10}$$

$$\begin{aligned}\frac{\partial(1-\hat{AP}_{sig})}{\partial f_t(x_p^-)} =& \frac{\partial(1-\hat{AP}_{sig})}{\partial \sigma_{pi}}\frac{\partial \sigma_{pi}}{\partial f_t(x_p^-)} \\ =& \sum_{i=1}^{P}\sum_{j=1}^{P} \frac{-\sigma_{ji}\sigma'_{pi}}{(\sum_{h=1}^{N}\sigma_{hi})^2},\end{aligned}$$

As, $\frac{\partial(1-\hat{AP}_{sig})}{\partial \sigma_{jp}}\frac{\partial \sigma_{jp}}{\partial f_t(x_p^-)} = 0$, since the example x_p from the previous formulation will always be positive in $1-\hat{AP}$.

In the following, we call SGBAP$_{sig}$, Stochastic Gradient Boosting algorithm using this sigmoid-based approximation $1-\hat{AP}_{sig}$.

3.3 Exponential-based surrogate of AP

It is worth noticing that by approximating the indicator function by the sigmoid function, the computation of the gradients as stated above is performed in quadratic time. This can be a too strong algorithmic constraint to deal with real world applications like fraud detection in credit card transactions (see the experimental section where the dataset contains 2,000,000 transactions). To overcome this issue, we suggest here to resort to a less costly surrogate of AP using the exponential function as an approximation of the indicator function.

$$\mathrm{I}(f(x_i) \leq f(x_j)) \approx e^{(f(x_j)-f(x_i))}.$$

As already done in Rankboost [17], we can show that the use of this exponential function allows us to reduce the time complexity for binary datasets to $\mathcal{O}(P+N)$.

Using the new approximation, AP takes the following form:

$$\hat{AP}_{exp} = \frac{1}{P}\sum_{i=1}^{P}\frac{\sum_{j=1}^{P} e^{f(x_j^+)}e^{-f(x_i^+)}}{\sum_{h=1}^{M} e^{f(x_h)}e^{-f(x_i^+)}} = \frac{1}{P}\sum_{i=1}^{P}\frac{e^{-f(x_i^+)}\sum_{j=1}^{P} e^{f(x_j^+)}}{e^{-f(x_i^+)}\sum_{h=1}^{M} e^{f(x_h)}}$$
$$= \frac{\sum_{j=1}^{P} e^{f(x_j^+)}}{\sum_{h=1}^{M} e^{f(x_h)}}$$

As for the sigmoid approximation, we rather use $1 - \hat{AP}_{exp}$ to minimize it.

$$1 - \hat{AP}_{exp} = \frac{\sum_{h=1}^{M} e^{f(x_h)} - \sum_{j=1}^{P} e^{f(x_j^+)}}{\sum_{h=1}^{N} e^{f(x_h)}} = \frac{\sum_{n=1}^{N} e^{f(x_n^-)}}{\sum_{h=1}^{M} e^{f(x_h)}} \tag{11}$$

Finally, finding the gradients of this new objective function is straightforward.

$$\begin{aligned}\frac{\partial 1 - \hat{AP}_{exp}}{\partial f(x_p^+)} &= \frac{-e^{f(x_p^+)}\sum_{n=1}^{N} e^{f(x_n^-)}}{(\sum_{h=1}^{M} e^{f(x_h)})^2} \\ \frac{\partial 1 - \hat{AP}_{exp}}{\partial f(x_p^-)} &= \frac{e^{f(x_p^-)}\sum_{i=1}^{M} e^{f(x_h)} - e^{f(x_p^-)}\sum_{n=1}^{N} e^{f(x_n^-)}}{(\sum_{h=1}^{M} e^{f(x_h)})^2}\end{aligned} \tag{12}$$

In the following, we call our method SGBAP, the stochastic gradient boosting based on our approximation $1 - \hat{AP}_{exp}$.

Note that, in Eq. 12, one can see an adverse effect brought by the exponential approximation of the indicator function. Indeed, if an $f(x_i)$ is first in the ranking, the gradient of x_i, $g(x_i)$, should decrease as there is no other position in which it will improve the overall AP. However, in our approximation, when $f(x_i)$ is significantly high, the gradient for this example will be the highest. Assume $\forall j \in \mathcal{S} \setminus x_i, f(x_i) >> f(x_j)$, we have $g(x_i) \approx 1$ and $g(x_k) \approx 0 \quad \forall k \in \mathcal{S}^- \setminus x_i$. In fact, this effect is limited with stochastic gradient boosting. Indeed, since $g(x_i)$ is not computed during all the iteration thanks to the random mini-batches, the gradient is then automatically regularized. However, running the gradient boosting algorithm instead of the stochastic version would raise the previous effect. The same holds for any basic gradient descent based algorithm.

3.4 Comparison between the approximations of AP

In this section, we compare experimentally the approximations used in this paper - $\hat{AP}_{exp}$ and $\hat{AP}_{sig}$ - with a simple one-dimensional sample described in Table 1. For this experiment, we use a simple linear model $f(x) = \theta_0 + \theta_1 x$. The toy dataset has been made such that the model has three ranking choices: (i) rank the examples in descending order from $x = +7$ to $x = -6$ (when $\theta_1 > 0$), (ii) rank the examples in descending order from $x = -6$ to $x = +7$ ($\theta_1 < 0$) or (iii) give the same rank to every example ($\theta_1 = 0$). We give the AP and $AUCROC$

Table 1: Toy dataset constituted of 14 examples on the real line with their associated labels. x correspond to the feature value and y the class.

x	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
y	-1	-1	-1	$+1$	$+1$	-1	-1	-1	-1	-1	-1	-1	$+1$	-1

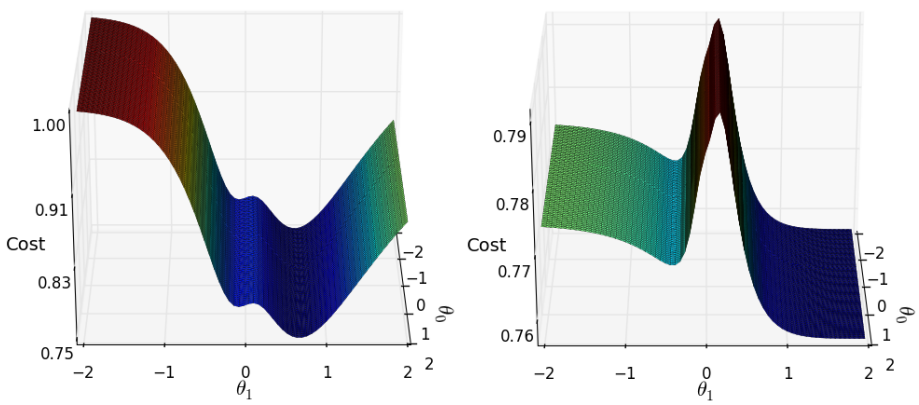

Fig. 3: $1 - \hat{AP}_{exp}$ (on the left) and $1 - \hat{AP}_{sig}$ (on the right) costs in function of the two model parameters θ_0 and θ_1.(better with color)

measures in each case : $AP = 0.29$, $AUCROC = 0.52$ when $\theta_1 < 0$, $AP = 0.33$, $AUCROC = 0.49$ when $\theta_1 > 0$ and $AP = 0.22$, $AUCROC = 0.5$ when $\theta_1 = 0$.

Figure 3, shows that the two objective functions considered are obviously not convex. However, they both find their minimum in $\theta_1 > 0$ which yields the best AP. In comparison, we show in the supplementary material a pairwise based and an accuracy based objective function that find their minimum in $\theta_1 < 0$.

Note that, on Figure 3, $1 - \hat{AP}_{exp}$ has another advantage than the time complexity over the sigmoid approximation. Indeed, for negative examples with high scores (e.g. when $\theta_1 > 1$), $1 - \hat{AP}_{sig}$ tends to have vanishing gradients while, for $1 - \hat{AP}_{exp}$, they tend to increase exponentially. Indeed, on Figure 3, the cost increases for the exponential approximation while it decreases for the sigmoid approximation.

4 Experiments

In this section, we present an experimental evaluation of our approach in two parts. In a first setup, we provide a comparative study with different state-of-the-art methods and various evaluation measures on 5 unbalanced public datasets coming either from the UCI Irvine Machine Learning repository or the LIBSVM datasets [3] and on a real dataset of credit card transactions provided by

[3] http://archive.ics.uci.edu/ml/ and https:/www.csie.ntu.edu.tw/∼cjlin/libsvmtools/

the private company ATOS/Worldline[4]. In a second experiment, we study the robustness of our method to undersampling of positives instances.

4.1 Top-rank quality over unbalanced datasets

In this experiment, we use the public datasets Pima, Breast cancer, HIV, Heart cleveland, w8a and the real fraud detection dataset over credit card transactions provided by ATOS/Worldline. This dataset contains 2 millions transactions labelled as 1 fraudulent or -1 genuine where 0.2% are fraudulent. It is constituted of 2 subsets of transactions of 3 consecutive days each. The first one is fixed as the training set and the second as the test test. Each subset being separated by one week in order to have the same week days (e.g. Tuesday to Thursday) in train and test. This setting models a realistic scenario where the feedback for every transactions is obtained only few days after the transaction was performed. The properties of the different datasets are summarized in Table 2.

Table 2: Properties of the 6 datasets used in the experiments.

	#examples	Positives ratio	#Features
Pima	767	34%	8
Breast cancer	286	30%	9
HIV	$3,272$	13.3%	8
Heart cleveland (4 vs all)	303	4.3%	13
w8a	64000	3%	300
Fraud	$2,000,000$	0.2%	40

We now describe our experimental setup. For the public datasets where the training/testing sets are not available directly, we randomly generate 2/3-1/3 splits of the data to obtain training and test sets respectively. Hyperparameters are tuned thanks to a 5-fold cross-validation over the training set, keeping the values offering the best AP. We repeat the process over 30 runs and average the results.

We compare our method, named SGBAP, to 4 other baselines[5]: SGBAP$_{sig}$ as defined previously, GB-Logistic which is the basic gradient boosting with a negative binomial log-likelihood loss function [16] (pointwise and accuracy based for binary datasets), LambdaMART-AP [22] a version of gradient boosting that optimizes the average precision and RankBoost [17], a pairwise version of AdaBoost for ranking. For each method, we fix a time limit to $86,000$sec.

We evaluate the previous methods according to 4 criteria measured on the test sets. First, we use the classic average precision (AP) and $AUCROC$. Additionally, we also consider 2 measures to best assess the quality of the approaches

[4] ATOS/Wolrdline is leader in e-transaction payments http://worldline.com/

[5] Note that we did not use Adarank in our evaluation because the weights updates rely on a notion of query that is not adapted to our framework.

for top-rank precision. For this purpose, we use the performance $Pos@Top$, defined in [26], that gives the percentage of positive example retrieved before a negative appears in the ranking. In other words, it corresponds to the recall before the precision drops under 100%. We also evaluate the $P@k$ from Eq. 4. In our setup, we set k to be the number of positive examples, which makes sense in our context of highly unbalanced data when the objective is to provide a short list of alerts to an expert and where the number of positive examples is much smaller than the negative examples. In fact, the latter measure is both precision and recall at rank k. This measure is also equivalent to the F_1 score since the latter is an harmonic mean between precision and recall. Note that we show experimentally in the supplementary material that AP is actually highly correlated to the F_1 score.

The results obtained are reported on Table 3. First, we can remark, that except for the Pima dataset that has the highest positive ratio, our approach is always better in terms of AP. SGBAP is also better than other baselines in terms of $Pos@top$ which is the hardest measure for evaluating the top-rank performance. Additionally, we see that for all datasets with a significantly low positive ratio (less than 15%), our approach is always better according to the P@k measure. Overall, we can remark that when the imbalance is high, our approach is always significantly better than other baselines according to 3 criteria: AP, $Pos@top$ and $P@k$ which clearly confirms that our method performs better for optimizing top-rank results. Note that, for the dataset HIV, SGBAP_{sig} performed quite poorly. We believe that this is because of the early vanishing gradient due to the imbalance in the dataset. This effect does not appear in heart cleveland dataset most likely because of the small dataset size.

4.2 Top rank capability for a decreasing positive ratio

In this section, we present an experiment showing the robustness of our approach when the ratio of positives decreases. We consider the Pima dataset because it has the highest ratio of positive instances and because our approach did not perform the best for all criteria. We aim at under-sampling the positive class (I.e. to decrease the positive ratio $\frac{P}{M}$). We start from the original positive ratio (34%) and go down to 3% by steps of ~ 0.05. For every new dataset, we follow the same experimental setup as described previously. At the end of the 30 runs for a given positive ratio dataset, we compute the average rank obtained by the examples in the test set and remove the top k positive instances such that $\frac{P-k}{M}$ is equal to the next positive ratio to evaluate. We repeat the previous set up until we reach 3% of positive examples in the dataset. We repeat this process independently for each method. The objective is to remove from the current dataset the easiest positive examples for each approach to evaluate its capability to move at the top new positive examples. Note that this makes harder the problem of ranking correctly in the top positive instances. Thus, the top rank performance measures should globally decrease.

Table 3: Results obtained for the different evaluation criteria used in the paper. We indicate in bold font the best method with respect to each dataset and each evaluation measure. A $-$ indicates that the method did not finish before the time limit.

Dataset	Algorithm	$AUCROC$	AP	$Pos@Top$	$P@k$
Pima	GB-Logistic	0.8279 ± 0.0352	0.7125 ± 0.0267	0.0388 ± 0.0379	$\mathbf{0.6608 \pm 0.0296}$
	RankBoost	$\mathbf{0.8352 \pm 0.0359}$	0.7281 ± 0.0621	$\mathbf{0.0620 \pm 0.0546}$	0.6586 ± 0.0298
	LambdaMART-AP	0.8177 ± 0.0304	$\mathbf{0.7338 \pm 0.0528}$	0.0407 ± 0.0443	0.6559 ± 0.0257
	SGBAP	0.8276 ± 0.0418	0.7119 ± 0.0486	0.0579 ± 0.0577	0.6455 ± 0.0356
	SGBAP_{sig}	0.8215 ± 0.0215	$0.7091 \pm, 0.0328$	0.0388 ± 0.0346	0.6514 ± 0.0325
Breast cancer	GB-Logistic	0.6821 ± 0.0756	0.5089 ± 0.0562	0.0931 ± 0.0561	0.4457 ± 0.0739
	RankBoost	0.6492 ± 0.0562	0.4838 ± 0.0632	0.0461 ± 0.0513	0.4626 ± 0.0629
	LambdaMART-AP	0.6733 ± 0.0419	0.5280 ± 0.0680	0.0859 ± 0.0828	$\mathbf{0.5196 \pm 0.0624}$
	SGBAP	0.7124 ± 0.0596	$\mathbf{0.5602 \pm 0.0830}$	$\mathbf{0.1019 \pm 0.1018}$	0.4980 ± 0.0612
	SGBAP_{sig}	$\mathbf{0.7131 \pm 0.0521}$	0.5503 ± 0.0443	0.0729 ± 0.0693	0.5061 ± 0.0574
HIV	GB-Logistic	0.8598 ± 0.0155	0.5557 ± 0.0376	0.0303 ± 0.0284	0.5391 ± 0.0364
	RankBoost	0.8599 ± 0.0127	0.5464 ± 0.0276	0.0401 ± 0.0363	0.5309 ± 0.0254
	LambdaMART-AP	0.8222 ± 0.0466	0.4286 ± 0.0887	0.0075 ± 0.0176	0.4874 ± 0.0814
	SGBAP	$\mathbf{0.8661 \pm 0.0150}$	$\mathbf{0.5737 \pm 0.0347}$	$\mathbf{0.0536 \pm 0.0410}$	$\mathbf{0.5445 \pm 0.0351}$
	SGBAP_{sig}	0.7578 ± 0.0231	0.3928 ± 0.0434	0.041 ± 0.0250	0.3902 ± 0.0439
Heart cleveland	GB-Logistic	0.7544 ± 0.1020	0.1638 ± 0.0931	0.0133 ± 0.0498	0.1 ± 0.1420
	Rankboost	$\mathbf{0.8109 \pm 0.0515}$	0.1739 ± 0.0638	0.0150 ± 0.0565	0.0967 ± 0.1335
	LambdaMART-AP	0.7277 ± 0.1225	0.1809 ± 0.1011	0.0383 ± 0.0863	0.1333 ± 0.1287
	SGBAP	0.7789 ± 0.1178	$\mathbf{0.2188 \pm 0.1103}$	$\mathbf{0.0483 \pm 0.0970}$	$\mathbf{0.2017 \pm 0.1044}$
	SGBAP_{sig}	0.7983 ± 0.0638	0.2136 ± 0.0964	0.045 ± 0.0906	0.1566 ± 0.1295
w8a	GB-Logistic	0.9544 ± 0.0039	0.7385 ± 0.0154	0.0534 ± 0.0529	0.7091 ± 0.0152
	RankBoost	$\mathbf{0.9712 \pm 0.0028}$	0.7649 ± 0.0135	0.0392 ± 0.0451	0.7277 ± 0.008
	LambdaMART-AP	$-$	$-$	$-$	$-$
	SGBAP	0.9701 ± 0.0029	$\mathbf{0.8351 \pm 0.0100}$	$\mathbf{0.1779 \pm 0.0978}$	$\mathbf{0.7972 \pm 0.0132}$
	SGBAP_{sig}	$-$	$-$	$-$	$-$
Fraud	GB-Logistic	0.8808	0.1477	0.0009	0.2411
	RankBoost	**0.8829**	0.1560	0.0005	0.2449
	LambdaMART-AP	$-$	$-$	$-$	$-$
	SGBAP	0.6878	**0.1747**	**0.0059**	**0.3203**
	SGBAP_{sig}	$-$	$-$	$-$	$-$

The results with respect to the AP criterion and $P@k$ are presented on Figure 4. From this experiment, we see that SGBAP outperforms the other models as the imbalance ratio increases and notably when the ratio of positives becomes smaller than 15% which confirms that our approach behaves clearly the best when the level of imbalance is high in comparison to other state of the art approaches.

5 Conclusion and perspectives

In this paper, we presented SGBAP, a novel Stochastic Gradient Boosting based approach for optimizing directly a surrogate of the average precision measure. Our approximation is based on an exponential surrogate allowing us to compute our criterion in linear time which is crucial for dealing with large scale datasets such as for fraud detection tasks. We claim that this approach is well adapted for

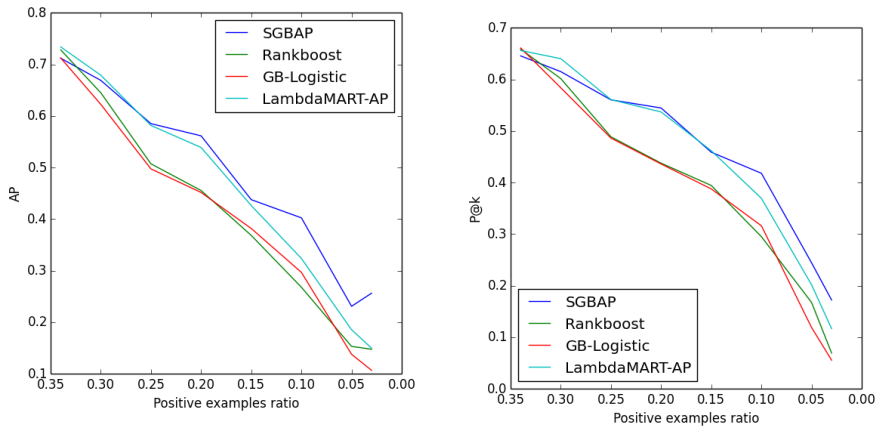

Fig. 4: The average precision and P@k at different positive example ratio for pima dataset.

supervised anomaly detection in the context of highly unbalanced settings. Indeed, our criterion focuses specifically on the top-rank yielding a better *precision* in the top k positions.

A perspective of this work would be to optimize other interesting measures for learning to rank such as NDCG by means of a stochastic gradient descent approach. Another direction, would be to adapt the optimization of the surrogate of average precision to other learning models such as neural networks where we could take benefit from recent results in non-convex optimization.

References

1. Cortes, C., Mohri, M.: AUC optimization vs. error rate minimization. In: NIPS. (2003) 313–320
2. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: synthetic minority over-sampling technique. JAIR (2002) 321–357
3. Ramentol, E., Caballero, Y., Bello, R., Herrera, F.: SMOTE-RSB *: a hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. Knowl. Inf. Syst. (2012) 245–265
4. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: Smoteboost: Improving prediction of the minority class in boosting. In: ECML PKDD. (2003) 107–119
5. Seiffert, C., Khoshgoftaar, T.M., Hulse, J.V., Napolitano, A.: Rusboost: A hybrid approach to alleviating class imbalance. IEEE Trans. Systems, Man, and Cybernetics (1) (2010) 185–197
6. Fan, W., Stolfo, S.J., Zhang, J., Chan, P.K.: Adacost: Misclassification cost-sensitive boosting. In: ICML. (1999) 97–105

7. Pozzolo, A.D., Caelen, O., Bontempi, G.: When is undersampling effective in unbalanced classification tasks? In: ECML PKDD. (2015) 200–215
8. Niculescu-Mizil, A., Caruana, R.: Predicting good probabilities with supervised learning. In: ICML. (2005) 625–632
9. Pozzolo, A.D., Caelen, O., Johnson, R.A., Bontempi, G.: Calibrating probability with undersampling for unbalanced classification. In: SSCI. (2015) 159–166
10. Liu, T.Y.: Learning to Rank for Information Retrieval. Springer (2011)
11. Joachims, T.: Optimizing search engines using clickthrough data. In: SIGKDD. (2002) 133–142
12. Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A support vector method for optimizing average precision. In: SIGIR. (2007) 271–278
13. Breiman, L.: Random forests. Machine learning **45**(1) (2001) 5–32
14. Freund, Y., Schapire, R., Abe, N.: A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence **14**(771-780) (1999) 1612
15. Chapelle, O., Chang, Y.: Yahoo! learning to rank challenge overview. (2011) 1–24
16. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics (2001) 1189–1232
17. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. The Journal of machine learning research **4** (2003) 933–969
18. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: ICML. (2005) 89–96
19. Herschtal, A., Raskutti, B.: Optimising area under the roc curve using gradient descent. In: Proceedings of the twenty-first international conference on Machine learning, ACM (2004) 49
20. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Learning **11** (2010) 23–581
21. Xu, J., Li, H.: Adarank: a boosting algorithm for information retrieval. In: SIGIR, ACM (2007) 391–398
22. Wu, Q., Burges, C.J., Svore, K.M., Gao, J.: Adapting boosting for information retrieval measures. Information Retrieval **13**(3) (2010) 254–270
23. Burges, C.J., Ragno, R., Le, Q.V.: Learning to rank with nonsmooth cost functions. In Schölkopf, P.B., Platt, J.C., Hoffman, T., eds.: NIPS. (2007) 193–200
24. Hanley, J.A., McNeil, B.J.: The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology **143**(1) (1982) 29–36
25. Friedman, J.H.: Stochastic gradient boosting. Computational Statistics & Data Analysis **38**(4) (2002) 367–378
26. Li, N., Jin, R., Zhou, Z.H.: Top rank optimization in linear time. In: NIPS. (2014) 1502–1510