A review on ranking problems in statistical learning

Tino Werner*

September 9, 2019

Ranking problems define a widely spread class of statistical learning problems with many applications, including fraud detection, document ranking or medicine. In this article, we systematically describe different types of ranking problems and investigate existing empirical risk minimization techniques to solve such ranking problems. Furthermore, we discuss whether a Boosting-type algorithm for continuous ranking problems is achievable by using surrogate loss functions.

Keywords— Ranking problems; Gradient Boosting

1 Introduction

Search-engines like Google provide a list of web-sites that are suitable for the user's query in the sense that the first web-sites that are displayed are expected to be the most relevant ones. Mathematically spoken, the search-engine has to solve a ranking problem which is done by the PageRank algorithm (Page et al. [1999]) for Google.

In their seminal paper (Clémençon et al. [2008]), Clémençon and co-authors proposed a statistical framework for ranking problems and proved that the common approach of empirical risk minimization is indeed suitable for ranking problems. Although there already existed ranking techniques, most of them indeed follow the ERM principle and can directly be embedded into the framework of Clémençon et al. [2008].

In general, the responses in data sets corresponding to those problems are binary, therefore a natural criterion for such binary ranking problems is the probability that an instance belongs to the class of interest. While ranking can be generally seen in between classification and regression, those binary ranking problems are very closely related to binary classification tasks (see also Balcan et al. [2008]). For binary ranking problems, there exists vast literature, including theoretical work as well as learning algorithms that use SVMs (Brefeld and Scheffer [2005], Herbrich et al. [1999], Joachims [2002]), Boosting (Freund et al. [2003], Rudin [2009]), neural networks (Burges et al. [2005]) or trees (Clémençon and Vayatis [2008], Clémençon and Vayatis [2010]).

As for the document ranking, the labels may also be discrete, but with $d > 2$ classes, for example in the OHSUMED data set (Hersh et al. [1994]). For such general $d-$partite ranking problems, there also has been developed theoretical work (Clémençon et al. [2013c]) as well as tree-based learning algorithms (Clémençon and Robbiano [2015a], Clémençon and Robbiano [2015b], see also Robbiano [2013]).

Recently, Clémençon investigated a new branch of ranking problems, namely the continuous ranking problems where the name already indicates that the response variable is continuous, with potential applications in natural sciences or quantitative finance (cf. Clémençon and Achab [2017]). This continuous ranking problem can be located on the other flank of the spectrum of ranking problems that is closest to regression.

The continuous ranking problem is especially interesting when trying to rank instances whose response is difficult to quantify. A common technique is to introduce latent variables which are used for example to measure or quantify intelligence (Borsboom et al. [2003]), personality (Anand et al. [2011]) or the

*Institute for Mathematics, Carl von Ossietzky University Oldenburg, P/O Box 2503, 26111 Oldenburg (Oldb), Germany

familiar background (Dickerson and Popli [2016]). While in these cases, the latent variables are treated as features, a continuous ranking problem would arise once a response variable which is hard to measure is implicitly fitted by replacing it with some latent score which is much more general than ranking binary responses by means of their probability of belonging to class 1. An example is given in Lan et al. [2012] where images have to be ranked according to their compatibility to a given query.

The major motivation for this review comes from the risk-based auditing context to detect tax evasion, using the restricted personal resources of tax offices as reasonable as possible. Risk-based auditing can be seen as a general strategy for internal auditing, fraud detection and resource allocation that incorporates different types of risks to be more tailored to the real-world situation, see Pickett [2006] for a broad overview, Moraru and Dumitru [2011] for a short survey of different risks in auditing and Khanna [2008] and Bowlin [2011] for a study on bank-internal risk-based auditing resp. for a study on risk-based auditing for resource planning.

The risk-based auditing context is indeed a very striking example where even the type of the suitable ranking problem is not determined in advance. One can formulate the problem as a binary ranking problem where the response variable is either tax compliance or a wrong report of the tax liabilities. However, as classification is not as informative as ranking since the classes do not have to be ordered while ranking also incorporates an ordering, ranking in turn is less informative than regression since regression tries to predict the actual response values themselves where ranking just tries to find the right ordering. An analogous argument is true for binary ranking problems and continuous ranking problems. If we state a binary ranking problem, we would just get information which taxpayer is most likely to misreport his or her income without providing any information on its amount. On the other hand, if we set up a continuous ranking problem where the amount of damage is the variable of interest, we can directly get information about the compliance of the taxpayer by looking at the sign of the response value. In particular, if information on the compliance is available, then one can assume that the information on the amount of additional payment or back-payment has also been collected, so imposing a binary ranking problem would lead to a large loss of information.

The promising risk-based auditing has first been studied in Alm et al. [1993]. In Gupta and Nagadevara [2007], the goal was to achieve a high strike rate, i.e., the relative part of the fraudulent income tax statements in the whole set of audited instances should be high, in contrast to Hsu et al. [2015], who validated their models (ranging from Naïve Bayes over SVM and Boosting to neural networks) by a profitability criterion, using the revenues from subsequent payments of taxes divided by the costs of the auditing process. Transferring it to the empirical risk minimization setting, the question how to measure the quality of a model outcome directly corresponds to the choice of the loss function for the ranking problem.

Furthermore, a good learning algorithm does not only require to have good predictive performance but also provide interpretability of the resulting model and computational feasibility as well as theoretical statistical foundations. The interpretability is closely connected to the number of predictors that enter the model, but a sparse model is not necessarily well-interpretable, for example RandomForests are hard to interpret. Sparse variable selection is especially indispensable in real-world applications when working with high-dimensional data which are a frequently faced issue particularly when having data coming from medicine or genomics (see e.g. Wang et al. [2011], Hofner et al. [2015]).

This paper is organized as follows. Starting in section 2 with the definition of several different ranking problems that are distinguished by the goal of the analyst, it becomes evident that suitable loss functions have at least a pair-wise structure in this case. We show how these loss functions look like and provide standardized versions of them for better interpretability. We further provide a simple recommendation how to evaluate the so-called hard ranking loss function with $\mathcal{O}(n \ln(n))$ computations. In section 3, we provide an overview of different machine learning algorithms that have already been translated to ranking problems with discrete response. We also list current approaches for high-dimensional data resulting in ranking algorithms with model selection. At the end of the section, we introduce the continuous ranking problem which is exactly corresponds to the problem of ranking different

tax payers based on the real-valued amount of damage. This kind of problem has recently been mathematically formulated, so there is not much related work yet. Section 4 is devoted to a detailed discussion of the applicability of Boosting-type algorithms to the hard continuous ranking problem. We provide a simple class of Boosting algorithms which are based on suitable surrogate losses with a pair-wise structure and provide arguments why this strategy cannot be meaningful, therefore we do not perform a whole simulation study.

2 Ranking problems

In this work, we always have data $\mathcal{D} = (X, Y) \in \mathbb{R}^{n\times(p+1)}$ where $Y_i \in \mathcal{Y}$ and $X_i \in \mathcal{X}$ where X_i denotes the $i-$th row of the regressor matrix X.

Solutions to ranking problems do not necessarily need to recover the responses Y_i based on the observations X_i. In fact, the goal is in general to predict the right ordering of the responses albeit there exist some relaxations of this (hard) ranking problem, e.g., only the top K instances have to be ranked exactly while the predicted ranking of the other instances is not a quantity of interest. Clémençon et al. [2005] and Clémençon et al. [2008] provided the theoretical statistical framework for empirical risk minimization in the ranking setting.

2.1 Different types of ranking problems

We try to rank the instances X_i by comparing their predicted response, i.e., X_i will be ranked higher than X_j if $\hat{Y}_i > \hat{Y}_j$. Then, Clémençon et al. [2008] provide the following definitions.

Definition 2.1. *With the convention above,*
a) *a* ***ranking rule*** *is a mapping* $r : \mathcal{X} \times \mathcal{X} \to \{-1, 1\}$ *where* $r(x, x') = 1$ *indicates that* x *is ranked higher than* x' *and vice versa.*
b) *a* ***ranking rule induced by a scoring rule*** s *is given by*

$$r(x, x', s) = 2I(s(x) \geq s(x')) - 1$$

with a scoring function $s : \mathcal{X} \to \mathbb{R}$ *where* $r(x, x') = 1$ *precisely if* $s(x) \geq s(x')$.

In this work, we will refer to the problem to correctly rank all instances as the **hard ranking problem** which is a global problem. A weaker problem is the **localized ranking problem** that intends to find the correct ordering of the best K instances, so misrankings at the bottom of the list are not taken into account. However, misclassifications have to be additionally penalized in this setting. It is obvious that these two problems are stronger problems than classification problems.

In contrast, sometimes it suffices to tackle the **weak ranking problem** where we only require to reliably detect the best K instances but where their pair-wise ordering is not a quantity of interest. For example, in the tax fraud detection context, we try to find the K most suspicious tax payers whose income tax statements need to be rigorously verified. If one knew that one exactly will review K instances (which is not a realistic assumption), it would not be necessary to try to predict which of them is the most suspicious one. This problem has been identified in Clémençon and Vayatis [2007] as a **classification problem with a mass constraint**, since we require to get exactly K class 1 objects if class 1 is defined as the "interesting" class.

We will always denote the index set of the true best $K \leq n$ instances by $Best_K$ and its empirical counterpart, i.e., the indices of the instances that have been predicted to be the best K ones, by $\widehat{Best_K}$.

Worked out theory for the weak and localized ranking problem is given in Clémençon and Vayatis [2007].

On the other hand, one distinguishes between three other types of ranking problems in dependence of the set $\mathcal{Y}$. If Y is binary-valued, w.l.o.g. $\mathcal{Y} = \{-1, 1\}$, then a ranking problem that intends to retrieve the correct ordering of the probabilities of the instances to belong to class 1 is called a **bipartite ranking problem (binary ranking problem)**. If Y can take d different values, a corresponding ranking problem is referred to as a $d-$**partite ranking problem** and for continuously-valued responses, one faces a **continuous ranking problem**.

Further discussions on possible combinations of these types of ranking problems and their relation to classification and regression follow in section 3.4.

2.2 Ranking by empirical risk minimization

Empirical risk minimization needs the definition of a suitable risk function. The hard ranking risk, i.e., the risk function of the hard ranking problem, introduced in Clémençon et al. [2005] is given by

$$R^{hard}(r) := \mathbb{E}[I((Y - Y')r(X, X') < 0)], \qquad (2.1)$$

so in fact, this is nothing but the probability of a misranking of X and X'. Thus, empirical risk minimization intends to find an optimal ranking rule by solving the optimization problem

$$\min_{r \in \mathcal{R}}(L_n^{hard}(r))$$

where

$$L_n^{hard}(r) = \frac{1}{n(n-1)} \sum\sum_{i \neq j} I((Y_i - Y_j)r(X_i, X_j) < 0) \qquad (2.2)$$

where $\mathcal{R}$ is some class of ranking rules $r : \mathcal{X} \times \mathcal{X} \to \{-1, 1\}$. For the sake of notation, the additional arguments in the loss functions are suppressed. Note that L_n^{hard}, i.e., the hard empirical risk, is also the hard ranking loss function which reflects the global nature of hard ranking problems.

In this work, we restrict ourselves to ranking rules that are induced by scoring rules. Considering some parameter space $\Theta \subset \mathbb{R}^p$, it suffices to empirically find the best scoring function (and with it, the empirically optimal induced ranking rule) by solving the parametric optimization problem

$$\min_{r \in \mathcal{R}}(L_n^{hard}(r))$$

with

$$L_n^{hard}(\theta) = \frac{1}{n(n-1)} \sum\sum_{i \neq j} I((Y_i - Y_j)(s_\theta(X_i) - s_\theta(X_j)) < 0). \qquad (2.3)$$

For the weak ranking problem, Clémençon and Vayatis [2007] introduce the upper $(1-u)-$quantile $Q(s, 1-u)$ for the random variable $s(X)$. To emphasize that this corresponds to a classification problem, we introduce the transformed responses

$$\tilde{Y}_i^{(K)} := 2I(rk(Y_i) \leq K) - 1$$

where the ranks come from a descending ordering. Then the misclassification risk corresponding to the weak ranking problem is given by

$$R^{weak,u}(s) := P(\tilde{Y}(s(X) - Q(s, 1-u)) < 0)$$

with the empirical counterpart

$$L_n^{weak,K}(s) = \frac{1}{n} \sum_{i=1}^{n} I(\tilde{Y}_i^{(K)}(s(X_i) - \hat{Q}(s, 1-u^{(K)})) < 0)$$

for the empirical quantile $\hat{Q}(s, 1-u^{(K)})$. To approximate the $(1-u)-$quantile, we need to set $u^{(K)} = K/n$, i.e., for a given quantile $(1-u)$, we look at the top K instances that represent this upper quantile.

Remark 2.1. *Due to the mass constraint, each false positive generates exactly one false negative, so the loss can be equivalently written as*

$$L_n^{weak,K}(s) = \frac{2}{n} \sum_{i \in Best_K} I(\tilde{Y}_i(s(X_i) - \hat{Q}(s, 1-u^{(K)})) < 0).$$

Note that the weak ranking loss is not standardized, i.e., it does not necessarily take values in the whole interval $[0, 1]$. More precisely, its maximal value is always $\frac{2K}{n}$, so we can only hit the value one if $K = \frac{n}{2}$ for even n and if all instances that belong to the "upper half" and predicted to be in the "lower half" and vice versa. For better comparison of the losses, we propose the **standardized weak ranking loss**

$$L_n^{weak,K,norm}(s) = \frac{1}{K} \sum_{i \in Best_K} I(\tilde{Y}_i(s(X_i) - \hat{Q}(s, 1-u^{(K)})) < 0). \qquad (2.4)$$

Remark 2.2. *Having get rid of the ratio K/n, the standardized weak ranking loss function has a very intuitive interpretation. For a fixed K, a standardized weak ranking loss of c/K means that c of the instances of $Best_K$ did not have been recovered by the model.*

A suitable loss function for the localized ranking problem was proposed in Clémençon and Vayatis [2007], too. In our notation, it is given by

$$L_n^{loc,K}(s) := \frac{n_-}{n} L_n^{weak,K}(s) + \frac{1}{n(n-1)} \sum\sum_{i \neq j} I(\{(s(X_i) - s(X_j))(Y_i - Y_j) < 0\} \cap \{\min(s(X_i), s(X_j)) \geq \hat{Q}(s, 1-u^{(K)})\}) \qquad (2.5)$$

In the second summand, n_- indicates the number of negatives, so the quotient is just an estimation for $P(Y = -1)$. Note that Clémençon and Vayatis [2007] introduced this loss for binary-valued responses. We propose to set $n_- := (n - K)$ for continuously-valued responses since localizing artificially labels the top K instances as class 1 objects, hence we get $(n - K)$ negatives. Again, the second summand may be rewritten as

$$\frac{2}{n(n-1)} \sum\sum_{i<j, i,j \in \widehat{Best}_K} I((s(X_i) - s(X_j))(Y_i - Y_j) < 0).$$

As the weak ranking loss, this loss is not $[0, 1]-$standardized. Taking a closer look on it, the maximal achievable loss given a fixed K is

$$\max(L_n^{loc,K}(s)) = \frac{K(K-1)}{n(n-1)} + \frac{n-K}{n} \cdot \frac{2K}{n} =: m_K,$$

so a standardized version is simply

$$L_n^{loc,K,norm}(s) := \frac{1}{m_K} L_n^{loc,K}(s).$$

Remark 2.3. *Note that even in the case $K = \frac{n}{2}$ for even n, the localized ranking loss cannot take the value one. This is true since*

$$L_n^{loc,n/2}(s) \leq \frac{\frac{n}{2}\left(\frac{n}{2} - 1\right)}{n(n-1)} + \frac{\frac{n}{2}}{n} \cdot 1 < \frac{\frac{1}{2}n(n-1)}{n(n-1)} + \frac{1}{2} = 1.$$

A simple example for clarification is given below in example 2.1. We insist to once more take a look on the U-statistics that arise for the hard and the localized ranking problem. Clémençon et al. [2008] already mentioned that these pair-wise loss functions can be generalized to loss functions with m input arguments. This leads to U-statistics of order m. But if the whole permutations that represent the ordering of the response values should be compared at once (i.e., $m = n$), then this again boils down to a U-statistic of order 2. Let

$$\text{Perm}(1:n) := \{\pi \mid \pi \text{ is a permutation of } \{1, ..., n\}\}$$

and let π, $\hat{\pi} \in \text{Perm}(1:n)$ be the true resp. the estimated permutation, then the empirical hard ranking loss can be equivalently written as

$$L_n^{hard}(\pi, \hat{\pi}) = \frac{2}{n(n-1)} \sum\sum_{i<j} I((\pi_i - \pi_j)(\hat{\pi}_i - \hat{\pi}_j) < 0). \tag{2.6}$$

Example 2.1. *Assume that we have a data set with the true response values*

$$Y := (-3, 10.3, -8, 12, 14, -0.5, 29, -1.1, -5.7, 119)$$

and the fitted values

$$\hat{Y} := (0.02, 0.6, 0.1, 0.47, 0.82, 0.04, 0.77, 0.09, 0.01, 0.79).$$

Then we order the vectors according to Y, so that $Y_1 \geq Y_2 \geq ...$ and get the permutations

$$\pi = (1, 2, ..., 10), \quad \hat{\pi} = (2, 3, 1, 5, 4, 8, 7, 9, 10, 6).$$

For example, $Y_{10} = 119$ is the largest value of Y, having rank 1. So we reorder $\hat{Y}$ such that $\hat{Y}_{10} = 0.79$ is the first entry. But since this is only the second-largest entry of $\hat{Y}$, we have a rank of 2, leading to the first component $\hat{\pi}_1 = 2$ and so forth.

Setting $K = 4$, we obviously get

$$L_n^{weak,4}(\pi, \hat{\pi}) = \frac{2}{10} = 0.2.$$

The standardized weak ranking loss is then

$$L_n^{weak,4,norm}(\pi, \hat{\pi}) = \frac{10}{8} \cdot \frac{2}{10} = 0.25$$

which is most intuitive since one of the indices of the four true best instances is not contained in the predicted set $\widehat{Best}_4$. The second part of the localized loss is then

$$\frac{2}{90}[0 + 1 + 0 + 1 + 0 + 0] = \frac{2}{45}.$$

This makes it obviously why the misclassification loss has to be included since this loss would be same if the instances of rank 4 and 5 were not switched. The complete localized ranking loss is

$$L_n^{loc,4}(\pi, \hat{\pi}) = \frac{2}{45} + \frac{6}{10} \cdot 0.2 = \frac{37}{225}.$$

The standardized localized ranking loss is then

$$L_n^{loc,4,norm}(\pi, \hat{\pi}) = \frac{75}{46} \cdot \frac{37}{225} \approx 0.268.$$

Finally, the hard ranking loss is

$$L_n^{hard}(\pi, \hat{\pi}) = \frac{2}{90} \cdot 8 = \frac{16}{90}.$$

Setting $K = 5$, the weak ranking loss is zero and the localized ranking loss is

$$L_n^{loc,5}(\pi, \hat{\pi}) = \frac{2}{90}[0 + 1 + 0 + 0 + 1 + 0 + 0 + 0 + 0 + 1] + \frac{5}{10} \cdot 0 = \frac{1}{15}.$$

The standardized localized ranking loss is

$$L_n^{loc,5,norm}(\pi, \hat{\pi}) = \frac{18}{13} \cdot \frac{1}{15} \approx 0.092.$$

The hard ranking loss is a global loss and does not change when changing K.

This nice and simple example has shown how important the selection of K can be.

Remark 2.4 (**Optimal ranking rules**). *As pointed out in Clémençon et al. [2008] and Clémençon and Achab [2017], if there is some optimal scoring rule for the hard continuous ranking problem, clearly any strictly increasing transformation of the scoring rule is also optimal.*

2.3 Fast computation of the hard ranking loss

Note that a naïve evaluation of the hard ranking loss requires $\mathcal{O}(n^2)$ comparisons. This will surely become infeasible for data sets with many observations, instead we provide a solution which comes up with $\mathcal{O}(n \ln(n))$ evaluations.

A key question concerning the implementation arises when the hard ranking loss of 2.3, equivalently formulated for predictions $\hat{Y}_i$ instead of a ranking rule r by

$$L_n^{hard}(Y, \hat{Y}) = \frac{1}{n(n-1)} \sum_{i \neq j} \sum I((Y_i - Y_j)(\hat{Y}_i - \hat{Y}_j) < 0),$$

needs to be evaluated several times.

Unfortunately, the naïve computation by comparing all pairs is very time-consuming for high numbers n of observations, making this approach non-competitive.

We take a look at the concordance measure

$$\tau(Y, \hat{Y}) := \frac{1}{n(n-1)} \sum_{i \neq j} \sum \text{sign}((Y_i - Y_j)(\hat{Y}_i - \hat{Y}_j))$$

called **Kendall's Tau**. Unlike the ranking loss which is high if there are many misrankings and which is $[0, 1]-$valued, the Kendall's Tau is high if many pairs are concordant, i.e., if the pair-wise ranking is correct in most cases and takes values in $[-1, 1]$.

This leads to a bijection between these two quantities if we do not face ties.

Lemma 2.1 (**Hard ranking loss and Kendall's Tau**). *Assume the vectors x and y have the same length n and do not contain ties. Then it holds that*

$$L_n^{hard}(x, y) = \frac{1 - \tau(x, y)}{2}.$$

Proof. *In the case of a perfect concordance, the ranking loss function is zero whereas Kendall's τ takes the value 1. If we produce one misranking, w.l.o.g. by swapping the largest and the second largest entry x_{j_1} resp. x_{j_2} of x, the indicator function in the ranking loss jumps from zero to 1 for $(i, j) \in \{(j_1, j_2), (j_2, j_1)\}$, increasing the total ranking loss by $\frac{2}{n(n-1)}$. The same manipulation results in the summands for the same indices in the Kendall's τ changing from 1 to -1, decreasing it by $\frac{4}{n(n-1)}$.*

By induction, the claimed formula is valid.

□

This indeed turns out to be useful in practice since there exists an R−command that provides fast computation of Kendall's τ, namely the command `cor.fk` from the package `pcaPP` (Filzmoser et al. [2018]). The algorithm essentially goes back to Knight (Knight [1966]) and relies on the idea of fast ordering algorithms. So in fact, we first compute Kendall's Tau using `cor.fk` and then, we use the bijection to compute the hard ranking loss which results in the number of calculations necessary for the computation of the hard ranking loss decreasing from $\mathcal{O}(n^2)$ in the naïve implementation to $\mathcal{O}(n \ln(n))$.

For the localized ranking loss 2.5, which includes the computation of the hard ranking loss restricted on the estimated set of best instances $\widehat{Best_K}$ (though differently standardized), we can use this function again. However, since K is usually small in comparison with n, the benefit of the accelerated computation may be smaller.

See figure 2.3 for an example where we compare the computation time of the hard and localized ranking losses using `cor.fk` with the naïve evaluation of the hard ranking loss. One may think of a data matrix with p regressor columns and one response column, each with n components, where we summed up the cost to quantify the dissimilarity in terms of ranking by computing the hard resp. some localized ranking losses of the response column w.r.t. each

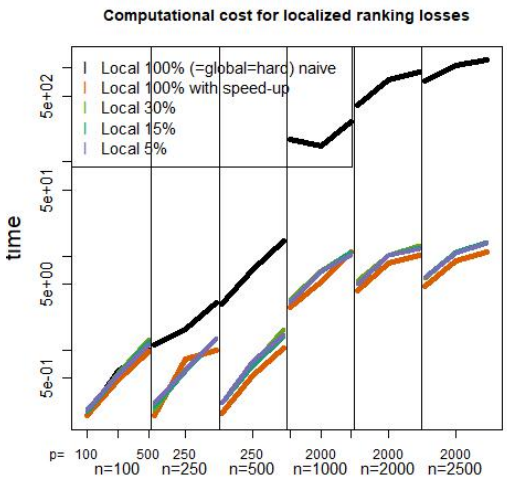

Figure 1: Time consumption: Naïve computation of the hard ranking loss and our computations of the hard and some localized ranking losses, exemplary on one data set for each allocation

single regressor column. It is expected that the cost grows linearly with p which results in the concave-shaped parts as we see in the figure. Furthermore, for different n we expect that the cost satisfies the complexity $\mathcal{O}(n^2)$ for the naïve computation (black) and $\mathcal{O}(n\ln(n))$ otherwise which for the latter does not exactly in the figure due to the very low times for these computations.

3 Current techniques to solve ranking problems

This section is divided into four parts. First, we review ranking algorithms that are based on machine learning algorithms like trees, Boosting or Support Vector Machines. Although some of these algorithms inherit the feature selection properties of the techniques they are based on, we provide an extra part devoted to ranking algorithms that are explicitely tailored to model selection. The third part introduces the continuous ranking problem. In the last part of this section, we discuss the differences of ranking and ordinal regression as well as the nine combined ranking problems where we start with the types of combined ranking problems that are primarily related to the recapitulated algorithms.

3.1 Existing algorithms for ranking problems

The existing methods to solve ranking problems make use of standard machine learning algorithms like Support Vector Machines (SVM), Boosting, trees or linear regression. For a useful overview and empirical comparison of some of these ranking algorithms, we refer to Clémençon et al. [2013b].

First, we note that there also exist Bayes-type algorithms where a distribution on the set of possible permutations is computed. Two prominent models are the Mallows model and the Plackett-Luce model. The Mallows model (Mallows [1957]) is based on distances between different permutations, in general based on Kendall's Tau, which leads to a maximum likelihood approach. The Plackett-Luce model (Luce [1959], Plackett [1975]) performs a Bayes estimation. However, we do not go into detail since the types of algorithms that motivated this work are different.

In the case of **bipartite ranking**, the sometimes called "plug-in approach" that estimates the conditional probability $P(Y = 1|X = x)$ can be realized for example by **LogitBoost**, i.e., minimizing the loss

$$\frac{1}{n}\sum_i \log_2(1+\exp(-2Y_i s(X_i))).$$

The resulting function s is then used as a ($[0,1]-$valued) scoring function for the ranking. However, the plug-in approach has disadvantages when facing high-dimensional data and when trying to establish an approximation of the true ROC curve in certain norms as pointed out in Clémençon and Vayatis [2008], Clémençon and Vayatis [2010].

Remark 3.1. *Taking a closer look on this loss function, it is indeed a convex surrogate of the misclassification loss. Concerning informativity, one just applies an algorithm that solves a classification problem which is less informative than a ranking problem which is another aspect why this approach may not be optimal.*

One approach to solve the bipartite ranking problem is to empirically maximize the AUC. This has been done in Rakotomamonjy [2004] and Ataman and Street [2005] by defining a formula of Mann-Whitney-type to calculate the empirical AUC resulting in an SVM-type ranking algorithm. A similar idea was presented in

Brefeld and Scheffer [2005], providing the SVM-type algorithm **AUC-SVM**.

A Boosting-type algorithm that essentially minimizes the empirical pair-wise exponential loss

$$\frac{1}{n(n-1)} \sum \sum_{i<j} \exp(-(Y_i - Y_j)(s(X_i) - s(X_j)))$$

for some scoring rule s is the **RankBoost** algorithm developed in Freund et al. [2003]. It is shown in Rudin and Schapire [2009] that in the case of binary outcome variables, RankBoost and the well-known classifier AdaBoost (Freund and Schapire [1997]) are equivalent under very weak assumptions. Therefore, RankBoost can also be seen as an AUC maximizer in the bipartite ranking problem. Another algorithm that includes gradients is the **RankNet** of Burges et al. [2005] that applies neural networks.

Note that there is a small mistake in Section 3.2.1 of Clémençon et al. [2013b] since the minus sign in the exponential function is missing. But if $Y_i > Y_j$ and $s(X_i) > s(X_j)$, the sign of the product is positive which would imply a high loss due to a positive exponent without the minus sign.

Clémençon and coauthors provided two tree-type algorithms, **TreeRank** and **RankOver** (Clémençon and Vayatis [2008], Clémençon and Vayatis [2010]). The idea behind the Tree-Rank algorithm is to divide the feature space $\mathcal{X}$ into disjoint parts C_j and to construct a piece-wise constant scoring function

$$s_N(x) = \sum_{j=1}^{N} a_j I(x \in C_j)$$

for $a_1 > ... > a_N$. This results in a ROC curve that is piece-wise linear with $(N-1)$ nodes (not counting $(0,0),(1,1)$) as shown in [Clémençon and Vayatis, 2008, Prop. 13]. The TreeRank algorithm then adaptively adds nodes between all existing nodes such that the ROC curve approximates the optimal ROC curve by splitting each region C_j in two parts. Extensions by combining the TreeRank algorithm with bagging in a RandomForest-like sense are given in Clémençon et al. [2009], Clémençon et al. [2013a]. The question how to prune a ranking tree was tackled in Clémençon et al. [2011].

Similarly, the RankOver algorithm constructs a piece-wise linear approximaton of the optimal ROC curve by computing a piece-wise constant scoring function, too, but instead of partitioning the feature space, it generates a partition of the ROC space.

Theoretically, these tree-type algorithms provide an advantage over the algorithms that optimize the AUC since they approximate the optimal ROC curve in an $L_\infty-$sense while the competitors just optimize the ROC in an L_1-sense (see [Clémençon and Vayatis, 2010, Sec. 2.2]). On the other hand, they suffer from strong assumptions since it is required that the optimal ROC curve is known. Additionally, this optimal ROC curve has to fulfill some regularity conditions which is differentiability and concavity for the TreeRank algorithm and twice differentiability with bounded second derivatives for the RankOver algorithm.

It is pointed out in Clémençon et al. [2013b] that ranking algorithms that intend to maximize the global AUC like RankBoost are not expected to be good candidates for the localized ranking problem whereas TreeRank is also constructed to solve that problem.

There are some other ranking algorithms that directly follow the ERM principle. As an extension of RankBoost, Rudin [2009] modified the exponential loss of RankBoost to the power loss

$$\frac{1}{n(n-1)} \sum \sum_{i<j} (\exp((Y_i - Y_j)(s(X_i) - s(X_j))))^p$$

for some $p \geq 1$. This algorithm is called the **p-Norm-Push**. The case $p = \infty$ has been studied in Rakotomamonjy [2012].

The **RankingSVM** algorithm (see Herbrich et al. [1999], Joachims [2002]) minimizes the empirical Hinge-type loss

$$\frac{2}{n(n-1)} \sum \sum_{i<j} [1 - (Y_i - Y_j)(f(X_i) - f(X_j))]_+ + \lambda ||f||^2_{\mathcal{H}_K}.$$

where $\mathcal{H}_K$ is some Reproducing Kernel Hilbert Space (RKHS) defined by a kernel K (see Schölkopf et al. [2001]).

Note that RankBoost, p-Norm-Push and RankingSVM essentially use surrogates for the hard ranking loss. For theoretical work on surrogate losses for the bipartite ranking problem, see Agarwal [2014].

Another approach is to try to predict the differences of pair-wise responses by the differences of the corresponding features. This can be achieved by minimizing the least-squares-type criterion

$$\frac{1}{n(n-1)} \sum_{i<j} \sum ((f(X_i) - f(X_j)) - (Y_i - Y_j))^2 + \lambda ||f||^2_{\mathcal{H}_k}.$$

for some function $f : \mathcal{X} \to \mathbb{R}$ and some kernel K with corresponding RKHS $\mathcal{H}_K$ (Pahikkala et al. [2007]). Using the representer theorem (see e.g. Schölkopf et al. [2001]), the solution has the form

$$f(X) = \sum_{i=1}^{N} a_i K(X, X_i)$$

for some $a_i \in \mathbb{R}$. The algorithm is called **RankRLS** ("regularized least squares").

The $d-$**partite ranking problem**, i.e., where the responses can take d different values, has been theoretically studied in Clémençon et al. [2013c] and Clémençon and Robbiano [2014].

First, an extension of the AUC in the multipartite case was adapted to the concrete setting in Clémençon et al. [2013c]. The AUC is generalized to the **volume under the ROC surface (VUS)** where the ROC surface is the "continuous extension" of discontinuity points based on hyperplane parts (note that the ROC surface originally was introduced in Yang and Carlin [2000]). Clémençon and Robbiano [2015b] provide the algorithm **TreeRank Tournament** which is based on RankTree and locally optimizes the ROC surface. A bagged and randomized version of TreeRank Tournament has been studied in Clémençon and Robbiano [2015a].

Remark 3.2. *Note that approaches that require the computation of the ROC curve and therefore of the AUC get infeasible in high dimensions due to the curse of dimensionality affecting numerical quadrature.*

3.2 Other ranking approaches with model selection

Clearly, the LogitBoost approach and the tree-type approaches Clémençon and co-authors provided already perform model selection. Pan et al. [2009] mention that it also has been tried to combine the RankBoost algorithm with stumps, but that this strategy has weaknesses in their search engine context. The first work on solving a ranking problem while explicitly concerning on sparse model selection was done by Geng et. al. (Geng et al. [2007]). They propose to assign an importance score to every single feature by computing a ranking based on each of them and computing a measure like the MAP (mean average precision) or NDCG (normalized discounted cumulative gain). Furthermore, a similarity measure between the features based on Kendall's τ is computed. Geng et al. [2007] solve an optimization problem to maximize the importance while minimizing the similarity to get a reduced feature set and eventually perform RankingSVM or RankNet on it. A similar strategy is spelled out in Pan et al. [2009] where stumps are used as baselearners and where the importance measure of [Friedman, 2001, Sec. 8] is used to find the most relevant features.

A robust and sparse ranking algorithm has been established in Sun et al. [2009], called **RSRank** for binary responses. They make use of a result in Balcan et al. [2008] that allows to reduce the binary ranking problem to weighted pair-wise classification. The goal in Sun et al. [2009] is to directly optimize the ranking loss measured by NDCG by minimizing the loss function

$$\sum_{i<j} \sum w(Y_i, Y_j, c) I(Y_i > Y_j) c(X_i, X_j)$$

where c is a pair-wise classifier and w a suitably chosen importance weight. For optimization, the indicator function is replaced by the Huber function and an l_1-penalty term is added. The minimization of the objective function is done by a truncated Gradient Boosting algorithm.

A pointed out in Tsivtsivadze and Heskes [2013], RankingSVM can also provide sparse solutions but they suffer from the lack of interpretability inherited from support vector machines. The objective function of RankingSVM is essentially l_2-constrained. Therefore, Lai et al. [2013a] replace this one with an l_1-regularization term and solve the problem by invoking Fenchel duality. Their algorithm is hence called **FenchelRank** which has been empirically shown to be even superior to RSRank in terms of precision. An iterative gradient procedure for this problem has been developed in Lai et al. [2013b] and shows comparable performance. As an extension of FenchelRank, Laporte et al. [2014] tackle the same

problem with nonconvex regularization to get even sparser models. They solve the problem with a so-called majorization minimization method where the nonconvex regularization term is represented by the difference of two convex functions. In addition, for convex regularization, they present an approach that relies on differentiability and Lipschitz continuity of the penalty term so that the ISTA-algorithm can be applied.

3.3 Ranking with continuous outcomes

All the presented existing algorithms are tailored to ranking problems with discrete labels since most of them are motivated by information retrieval or document ranking. To the best of our knowledge, the only approach explicitly designed for ranking real-valued responses was recently proposed by Clémençon (Clémençon and Achab [2017]).

Let w.l.o.g. $Y \in [0,1]$. Then each **subproblem**

$$\max_s (P(s(X) > t|Y > y) - P(s(X) > t|Y < y))$$

for $y \in [0,1]$, i.e., $s(X)$ given $Y > y$ should be **stochastically larger than** $s(X)$ given $Y < y$, is a bipartite ranking problem, so the continuous ranking problem can be regarded as a so-called **"continuum" of bipartite ranking problems** (Clémençon and Achab [2017]).

As a suitable performance measure, they provide the area under the integrated ROC curve

$$\text{IAUC}(s) := \int_0^1 \underset{s}{\text{IROC}}(\alpha) d\alpha := \int_0^1 \int \underset{s,y}{\text{ROC}}(\alpha) dF_y(y) d\alpha$$

where $\text{ROC}_{s,y}$ indicates the ROC curve of scoring function s for the bipartite ranking problem corresponding to $y \in]0,1[$ and where F_y is the marginal distribution of Y. Alternatively, they make use of Kendall's τ as a performance measure for continuous ranking.

The approach presented in Clémençon and Achab [2017] manifests itself in the tree-type **CRank** algorithm that divides the input space and therefore the training data into disjoint regions. In each step/node, the binary classification problem corresponding to the median of the current part of the training data is formulated and solved. Then, all instances whose predicted label was positive are delegated to the left children node, the others to the right children node. Stopping when a predefined depth of the tree is reached, the instance of the leftmost leaf is ranked highest and so far, so the rightmost leaf indicates the bottom instance.

Remark 3.3 (**Interpretability**). *Note that single trees in general lack the properties of stability and robustness (see for example Friedman et al. [2001]). On the other hand, when aggregating trees like it has been done for trees that solve the bipartite ranking problem (Clémençon et al. [2009], Clémençon et al. [2013a]), one essentially faces partial rankings predicted by each single tree. To aggregate them, they compute a so-called consensus ranking from these partial rankings and a median scoring rule. As for RandomForests, it would be very hard to interpret this combined model in the sense of quantifying the impact of a single predictor on the consensus ranking.*

3.4 Ranking vs. ordinal regression

Ordinal regression problems are indeed very closely related to ranking problems. As already pointed out in Robbiano [2013], especially multipartite ranking problems (Clémençon et al. [2013c]) share the main ingredient, i.e., the computation of a scoring function that should provide pseudo-responses with a suitable ordering. However, the main difference is that the multipartite ranking problem is already solved once the ordering of the pseudo-responses is correct while the ordinal regression problem still needs thresholds such that a discretization of the pseudo-responses into the d classes of the original responses is correct.

Note that due to the discretization, ordinal regression problems can also be perfectly solved even if the rankings provided by the scoring function are not perfect. For example, consider observations with indices $i_1, ..., i_{m_k}$ that belong to class k. If for a scoring rule s we had the predicted ordering $s(X_{i_1}) < s(X_{i_2}) < ... < s(X_{i_{m_k}})$ but the true ordering is different, then we can still choose thresholds such that all m_k instances that belong to class k (and no other instance) are classified into this class, provided that $s(X_i) \notin [s(X_{i_1}), s(X_{i_{m_k}})] \ \forall i \notin \{i_1, ..., i_{m_k}\}$.

Though, as Robbiano [2013] already pointed out, the ordinal regression is based on another loss function.

However, the setting of ordinal regression is of course not obsolete since despite even if such a $d-$partite ranking problem can be solved, the model provides a predicted ordering of the test instances, more precisely, the ordering of the pseudo-responses is sophisticatedly predicted, but there are still no true responses. Therefore, in all applications where predicted labels for the test instances are needed, ordinal regression in the sense that suitable cutoffs have to be defined is indispensable.

Concerning informativity, one can state that multipartite ranking problems are more informative than ordinal regression problems due to the chunking that is done in the latter ones. But in fact, in an intermediary step, i.e., when having computed the scoring function, the ordinal regression problem is as informative as multipartite ranking problems. This is also true for standard logit or probit models (the two classes generally are not ordered, but when artificially replacing the true labels by -1 and $+1$ where the particular assignment does not affect the quality of the models, they can indeed be treated as ordinal regression models) where the real-valued pseudo-responses computed by the scoring function are discretized at the end to have again two classes.

The continuous ranking problem can be treated as a special case where no pseudo-responses are needed since the original responses are already real-valued, but again, instead of optimizing some regression loss function, the goal is actually to optimize a ranking loss function.

For further discussions on the relation of ranking and ordinal regression (also called "ordinal classification" and "ordinal ranking" in the reference), see Lin [2008].

From this point of view, the three combined problems for the continuous case, i.e., weak, hard and localized continuous ranking problems, are easy to distinguish and are all meaningful. Hard bipartite and hard $d-$partite ranking problems are essentially optimized by the corresponding algorithms that we listed earlier in section 3.1 and localized bipartite ranking problems can be solved using the tree-type algorithms of Clémençon as pointed out for instance in Clémençon et al. [2013b].

Clearly, these localized bipartite problems directly reflect the motivation from risk-based auditing or document retrieval. It has been mentioned in Clémençon and Robbiano [2015b] that their tree-type algorithm is not able to optimize the VUS locally. To the best of our knowledge, this has not been achieved until now. But indeed, localized $d-$partite ranking problems can also be interesting in document retrieval settings where the classes represent different degrees of relevance. Then it would be interesting for example to just recover the correct ranking of the relevant instances, i.e., the ones from the "best" $(d-1)$ classes.

As mentioned earlier, weak ranking problems can be identified with binary classification with a mass constraint. In the case of weak bipartite ranking problems, it may sounds strange to essentially mix up two classification paradigms, but one can think of performing binary classification by computing a scoring function and by predicting each instance as element of class 1 whose score exceeds some threshold, as it is done for example in logit or probit models. One can think of choosing the threshold such that there are exactly K instances classified into class 1 instead of optimizing the AUC or some misclassification rate.

The only combination that does not seem to be meaningful at all would be weak $d-$partite ranking problems. By its inherent nature, a weak ranking problem imposes are binarity which cannot be reasonably given for the $d-$partite case. Even in the document retrieval setting, a weak $d-$partite ranking problem may be thought of trying to find the K most important documents which implied that the information that is already given by the d classes would be boiled down to essentially two classes, so this combination is not reasonable.

4 Gradient Boosting for ranking problems?

Gradient Boosting (see e.g. Bühlmann and Hothorn [2007]) methods form a very flexible class of algorithms that efficiently provide sparse models and which are excellently implemented in the R−package `mboost` (Hothorn et al. [2017]).

While for binary ranking problems there are some adaptations like RankBoost or the p-Norm-Push, for continuous ranking problems so far there is no corresponding Boosting procedure. In order to make Boosting accessible for the hard continuous ranking problem, we try to embed this problem into the Gradient Boosting framework of Bühlmann.

We will see that the hard ranking loss function itself does not satisfy the required regularity properties to make Gradient Boosting applicable. Motivated by standard surrogate losses from classification, we try to define some kind of pair-wise surrogate losses for the hard ranking loss and to construct three Gradient Boosting algorithms, each for another surrogate loss.

At the end, we provide heuristic arguments why Gradient Boosting algorithms of this type always suffer from the pair-wise nature of the loss functions, resulting in a very poor speed, aside from the empirically revealed fact that their ranking performance on test data is rather poor due to other issues that arise from the choice of the surrogates.

4.1 Gradient Boosting

The following generic functional Gradient Boosting algorithm goes back to Friedman (Friedman [2001]). We use the notation of Bühlmann and Hothorn [2007].

The function L is concerned to be a loss function with two arguments. One is the response, the other one is the predicted response, understood in a functional way as the model that is used for prediction. The loss function has to be differentiable and convex in the second argument. The main idea behind this algorithm is to iteratively proceed along the steepest gradient.

As pointed out in Bühlmann and Hothorn [2007], the models $\hat{g}^{(k)}$ can be regarded as an approximation of the current negative gradient vector.

A very popular algorithm is L_2-Boosting (Bühlmann and Yu [2003], Bühlmann [2006]) which corresponds to the squared loss.

Initialization: Data (X, Y), step size $\kappa \in]0, 1]$, number m_{iter} of iterations and

$$\hat{f}^{(0)}(\cdot) \equiv \underset{c}{\operatorname{argmin}} \left(\frac{1}{n} \sum_i L(Y_i, c) \right)$$

as offset value;
for $k = 1, ..., m_{iter}$ **do**
Compute the negative gradients and evaluate them at the current model:

$$U_i = -\partial_f L(Y_i, f)|_{f=\hat{f}^{(k-1)}(X_i)}$$

for all $i = 1, ..., n$;
Treat the vector $U = (U_i)_i$ as response and fit a model

$$(X_i, U_i)_i \overset{\text{base procedure}}{\longrightarrow} \hat{g}^{(k)}(\cdot)$$

with a preselected real-valued base procedure;
Update the current model via

$$\hat{f}^{(k)}(\cdot) = \hat{f}^{(k-1)}(\cdot) + \kappa \hat{g}^{(k)}(\cdot)$$

end
Algorithm 1: Generic functional Gradient Boosting

4.2 Arising problems

As introduced in equation (2.3), the loss function for the hard ranking problem may be rewritten as

$$\frac{1}{n(n-1)} \sum_{i \neq j} \sum L((Y_i, Y_j), f),$$
$$L((Y_i, Y_j), f) := I((Y_i - Y_j)(f(X_i) - f(X_j)) < 0) \tag{4.1}$$

to mimic the notation of Bühlmann and Hothorn [2007]. Usually, one would compute the gradient by differentiating the loss function L with respect to f. However, as pointed out in [Bühlmann and Hothorn, 2007, Ch. 2], L has to be convex and differentiable in f which is obviously not the case. So, the hard ranking loss function is not suitable for Gradient Boosting.

In such cases, e.g. when facing the $0/1-$loss in a classification setting, one commonly defines some convex surrogate loss function, for example by the exponential loss or the Hinge loss. See figure **??** for (convex and non-convex) surrogate losses. Invoking the exponential surrogate for the indicator function in the ranking setting has already been done in Freund et al. [2003], Rudin and Schapire

[2009] which led to the RankBoost algorithm and its modification, the p-Norm-Push, which we already mentioned before. However, their algorithms (see the two former references or the survey Clémençon et al. [2013b]) are tailored to binary classification problems.

4.3 An exponential surrogate

The question that arises from the nature of the hard ranking loss function as double sum of indicator functions is if the approximation of the indicator function by suitable surrogates can lead to a Gradient Boosting procedure for the hard ranking problem. In this section, we replace the indicator functions by exponential functions, exactly as it has been done for the algorithm RankBoost. To write down the empirical risk in the form of Bühlmann and Hothorn [2007], we define

$$L_i^{exp}(Y,f) := \frac{1}{n-1}\sum_{j\neq i}\exp(-(Y_i-Y_j)(f(X_i)-f(X_j))) \tag{4.2}$$

so that the empirical loss is the empirical mean of all L_i over i.

The most natural way to define the required gradients (see section 4.1) is to set

$$U_i^{exp} := -\partial_f L_i^{exp}(Y,f)|_{f=\hat{f}^{(m)}}$$
$$= \frac{1}{n-1}\sum_{j\neq i}(Y_i-Y_j)\exp(-(Y_i-Y_j)(\hat{f}^{(m)}(X_i-X_j))) \tag{4.3}$$

provided that f is a linear function, so that $f(X_i-X_j)$ equals $f(X_i)-f(X_j)$. This resembles the usual Gradient Boosting, the only difference is that all components of Y and $f(X)$ have to enter the gradient in our case.

Due to the sum structure of the losses, even each of the L_i inherits the combinatorial nature of the ranking problem, therefore we cannot determine an exact minimizer, i.e., a suitable baselearner which is tailored to the exponential surrogate. Algorithmically, we proceed as in the component-wise L_2-Boosting (see Bühlmann [2006]) where a regression w.r.t. every single predictor, i.e., a simple regression, is performed and the predictor that decreased the RSS most is taken. The difference in our case is that we evaluate the exponential loss instead of the RSS and that we have to evaluate the gradient (4.3) instead of the least squares residual.

Additionally, we are not able do define an offset as in other Boosting methods. The offset is an empirical population minimizer and usually the mean or the median of the responses in the case of L_2- or L_1-Boosting which does clearly not make any sense in the ranking case since a constant always leads to a perfect zero ranking loss. Therefore, we decided to start which the original values of Y in the first step. Just for internal reference, we call the proposed algorithm **ExpBoost**.

Initialization: Data (X,Y), step size $\kappa \in]0,1]$, number m_{iter} of iterations and parameter vector $\hat{\beta}^{(0)} = 0_{p+1}$;
Set $r^{(0)} := Y$;
for $k = 1, ..., m_{iter}$ **do**
  for $j = 1, ..., p$ **do**
    Fit the current gradients $r^{(k-1)}$ by a simple least squares regression model using the predictor variable j;
    Compute the exponential loss;
  end
  Take the variable $\hat{j}_k$ whose simple model $\hat{\beta}_{\hat{j}_k} \in \mathbb{R}^{p+1}$ provides the smallest exponential loss;
  Update the model via $\hat{\beta}^{(k)} = \hat{\beta}^{(k-1)} + \kappa\hat{\beta}_{\hat{j}_k}$;
  Compute the current gradients $r^{(k)} = (-\partial_f L_i^{exp}(Y,f)|_{f=\hat{\beta}^{(k)}})_{i=1}^n$ as in (4.3)
end

Algorithm 2: ExpBoost with component-wise linear baselearners

Experiments show that, as expected, it performs rather poor. The reason is the highly non-robust exponential loss function which is a really bad approximant of the indicator function for small values since in contrast to classification, its argument takes values in $\mathbb{R}$ instead of $\{\pm 1\}$. Furthermore, the function does not take a minimum in $\mathbb{R}$.

Surprisingly, the algorithm can even break down after one single iteration on a moderate data set! We also tried an *expit* transformation of the response values before applying the algorithm or a further scaling of the gradients by n^{-1}, but after some iterations, the gradients also diverged. Evidently, the exponential loss is definitely not appropriate for the continuous ranking

problem.

Remark 4.1. *This is no contradiction to the RankBoost algorithm since RankBoost proceeds in the same manner as AdaBoost, so its iterations compute weights assigned to the observations (more precisely, a distribution on the pairs of the observations is updated). The gradients are not evaluated explicitly.*

4.4 A Hinge surrogate

Next, we try to use the Hinge loss that at most grows linearly for decreasing values. In the same manner as before, we propose

$$L_i^{hinge}(Y,f) := \frac{1}{n-1}\sum_{j\neq i}\max(0, 1-(Y_i-Y_j)(f(X_i)-f(X_j))). \tag{4.4}$$

The gradient is given by

$$U_i^{hinge} := -\partial_f L_i^{hinge}(Y,f)|_{f=\hat{f}^{(m)}} = \frac{1}{n-1}\sum_{j\neq i}\begin{cases} Y_i - Y_j, & (Y_i-Y_j)(\hat{f}^{(m)}(X_i-X_j)) < 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.5}$$

We rely on the argument of [Bühlmann and Hothorn, 2007, Ch. 3.2] for the L_1-loss that the region of non-differentiability of the loss is just a single point that has zero probability which indeed holds for the Hinge loss.

However, the running time of this algorithm that we refer to as **HingeBoost** is still very high due to the pair-wise structure of the single gradients and losses. Compared to the L_2-Boosting with component-wise linear baselearners, it is absurdly slow.

4.5 A piece-wise linear surrogate

We propose the much tighter surrogate

$$L_i^c(Y,f) := \frac{1}{n-1}\sum_{j\neq i}\begin{cases} 1, & (Y_i-Y_j)f(X_i-X_j) < 0 \\ 0, & (Y_i-Y_j)f(X_i-X_j) > c \\ 1-\frac{(Y_i-Y_j)f(X_i-X_j)}{c}, & \text{otherwise} \end{cases},$$

so we approximate the indicator function in a piece-wise linear manner, for $c \in]0,1]$. The probability of the product of the differences being located at 0 or c is zero, so the differentiability is still valid almost-everywhere. Although we violate the convexity assumption, this loss function is still piece-wise convex.

The gradient is

$$U_i^c = -\partial_f L_i^c(Y,f)|_{f=\hat{f}^{(m)}} = \frac{1}{n-1}\sum_{j\neq i}\frac{Y_i-Y_j}{c} I((Y_i-Y_j)\hat{f}^{(m)}(X_i-X_j) \in]0,c[).$$

Besides its very slow running time, this algorithm can be completely worthless due to an astounding issue. In each iteration, the algorithm selects the variable which is exactly the one whose column ranking of the data matrix already is at most comparable with either the response or the negative response, i.e., it selects predictor variable j_0 with

$$j_0 = \underset{j}{\operatorname{argmin}}(\min(L_n^{hard}(Y, X_{\cdot,j}), L_n^{hard}(-Y, X_{\cdot,j}))).$$

The coefficient is positive or negative, depending on the fact if the column ranking resembles the ranking of the response or of the negative response most, respectively (which is clearly inherited from the fact that the simple regression slope estimator is a standardized covariance between the response and the respective column in the regressor matrix). It does not necessarily happen, but the issue sometimes occurs that this variable is again chosen in the subsequent iterations, and in this case, the whole Boosting model just contains one variable (and the intercept), so the resulting model is meaningless. Taking a closer look on it, we clearly see that the gradients that are computed are likely to be zero. Only in the small region $]0,c[$, we get a non-zero gradient. This is the reason why the algorithm sometimes ends up in just having selected one single variable.

Remark 4.2 (**Comparison with L_2-Boosting**). *It maybe looks irritating that this algorithm does not perform well since L_2-Boosting selects in each step the variable which is most correlated with the current residual. If there was a column whose ranking was exactly the same as the ranking of the residual column, L_2-Boosting would clearly select this column. But one has to keep in mind that concordance and correlation are only identical in the case of perfect concordance or*

perfect discordance and that L_2-Boosting and our proposed Boosting algorithm proceed along very different gradients.

4.6 Simulation

The abovely mentioned issues can be directly realized in practical applications.

In a simple simulation, we generated the regressor matrix from a multivariate normal distribution of dimension p with mean zero and with uncorrelated columns. The non-zero coefficients stem from a standard normal distribution and the response vector is generated according to the linear model $Y_i = X_i\beta$. We use a signal-to-noise ratio of 2.

We try three different allocations for p and n, namely $(p, n) \in \{(25, 50), (100, 50), (250, 200)\}$ where 10 resp. 20 resp. 10 coefficients are non-zero.

We apply L_2-Boosting, ExpBoost and HingeBoost with 100 iterations and learning rate 0.1 to each data set as well as the Boosting procedure using the piece-wise linear surrogate with $c \in \{0.1, 0.5, 0.9\}$. Furthermore, we apply an *expit* transformation to the response vector to reduce its range to the interval $]0, 1[$ and apply each of these algorithms again. We compute the in-sample hard ranking loss.

ExpBoost never finishes since the gradients diverge, leading to the algorithm break down.

The Boosting based on piece-wise linear surrogates always leads to hard ranking losses which are much worse than the loss for L_2-Boosting. Furthermore, it either selects very few variables or overfits, depending on c. HingeBoost is much worse than L_2-Boosting for the original responses but profits from the *expit* transformation. However, even in this case, the performance is worse than that of L_2-Boosting, not taking the immense computational time into account.

Further details and R−code are available from the author upon request.

5 Conclusion and outlook

We provided a systematic review of different ranking problems, concerning both the type of the response variable and the underlying loss function. We emphasized that there are applications like the risk-based auditing setting where different approaches could be valid, leading to a different amount of information and therefore to different interpretability properties of the models. We discussed the evaluation of the pair-wise hard ranking loss function and provided a strategy that reduces the computational complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \ln(n))$.

As for continuous ranking problems, we discussed that Boosting-type algorithms that rely on surrogates of the hard ranking loss function are not recommended due to the insufficient approximation of the hard ranking loss and due to the pair-wise structure which leads to computational infeasibility for high dimensions.

One may ask if we can accelerate the evaluation of the Hinge loss in the HingeBoost algorithm benefitting from the constant part.

An idea could be to find a partial ordering of the products of the pair-wise differences. If for some product $f(X_i - X_j)(Y_i - Y_j)$ the Hinge loss is zero, then for all other products of pair-wise differences that are larger, the Hinge loss is also zero. There exists work where such partial ordered sets, so-called "posets", are treated, see e.g. Daskalakis et al. [2011] and references therein, but sorting them cannot be achieved with a lower complexity than $\mathcal{O}(n \ln(n))$, to the best of our knowledge.

We can deduct that there will always be a loss in performance w.r.t. the L_2- or L_1-Boosting when having pair-wise loss functions. Since the algorithms are nearly identical, we only need to concern the cost of evaluating the loss function in and evaluating the gradient at the end of each iteration. The cost is of order $\mathcal{O}(pn)$ for Gradient Boosting with standard losses in each Boosting iteration. In the way we implemented ExpBoost and HingeBoost, it is $\mathcal{O}(pn^2)$. Even if one could invoke fast sorting algorithms for our case, the cost would be a least of order $\mathcal{O}(pn \ln(n))$. That may do not look that bad, but this would be an absolutely idealistic situation.

Assume that we have a poset P of the products of pair-wise differences as above, in an ascending order. In fact, the poset contains all quadruples (X_i, X_j, Y_i, Y_j). To benefit from the idea, we would need to determine the element P_{i_0} of the poset which leads to the smallest product of pair-wise differences that exceeds one, so we would need to find this element in a set of cardinality $\binom{n}{2}$. Furthermore, the usage of a partial order is only meaningful in the area in which the loss function is constant. In the case of the Hinge loss, we still have to evaluate the loss for any element P_k, $k < i_0$, of the poset.

Therefore, we expect that no Gradient Boosting algorithm with a pair-wise loss function could be competitive with Gradient Boosting with standard losses in terms of computational efficiency.

5.1 Outlook

We tried to make Gradient Boosting accessible for the hard continuous ranking problem by transferring the concept of surrogate losses for classification to the ranking setting. In constrast to binary ranking problems where the structure of the responses allows for Gradient Boosting, continuous responses are still problematic.

This can be thought of a starting point to develop a new strategy for model selection and coefficient estimation in the case of irregular or complicated loss functions of which the hard ranking loss is an exemplar. Our proposal is a so-called "gradient-free Gradient Boosting" technique which can cope with irregular loss functions by performing so-called "singular iterations" w.r.t. the complicated loss function. This can be combined with a special Stability Selection which is tailored to such problems. The ideas behind this class of algorithms are described in detail in the author's PhD thesis (Werner [2019]) and will be presented in future papers.

Another issue is a fair and objective, procedure-independent criterion at which to compare solutions obtained through different approaches. This can be solved through the elicitability property of the corresponding statistical functional (see e.g. Gneiting [2011], Fissler et al. [2015], Ziegel [2016]). We will provide results on elicitability from Werner [2019] in another forthcoming paper.

Acknowledgements

The results presented in this paper are part of the author's PhD thesis (Werner [2019]) supervised by P. Ruckdeschel at Carl von Ossietzky University Oldenburg.

References

S. Agarwal. Surrogate regret bounds for bipartite ranking via strongly proper losses. *The Journal of Machine Learning Research*, 15(1):1653–1674, 2014.

J. Alm, M. B. Cronshaw, and M. McKee. Tax compliance with endogenous audit selection rules. *Kyklos*, 46(1):27–45, 1993.

P. Anand, J. Krishnakumar, and N. B. Tran. Measuring welfare: Latent variable models for happiness and capabilities in the presence of unobservable heterogeneity. *Journal of public economics*, 95(3-4): 205–215, 2011.

K. Ataman and W. N. Street. Optimizing area under the ROC curve using ranking SVMs. In *Proceedings of International Conference on Knowledge Discovery in Data Mining*, 2005.

M.-F. Balcan, N. Bansal, A. Beygelzimer, D. Coppersmith, J. Langford, and G. B. Sorkin. Robust reductions from ranking to classification. *Machine learning*, 72(1-2):139–153, 2008.

D. Borsboom, G. J. Mellenbergh, and J. Van Heerden. The theoretical status of latent variables. *Psychological review*, 110(2):203–219, 2003.

K. Bowlin. Risk-based auditing, strategic prompts, and auditor sensitivity to the strategic risk of fraud. *The Accounting Review*, 86(4):1231–1253, 2011.

U. Brefeld and T. Scheffer. AUC maximizing support vector learning. In *Proceedings of the ICML 2005 workshop on ROC Analysis in Machine Learning*, 2005.

P. Bühlmann. Boosting for high-dimensional linear models. *The Annals of Statistics*, pages 559–583, 2006.

P. Bühlmann and T. Hothorn. Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, pages 477–505, 2007.

P. Bühlmann and B. Yu. Boosting with the l_2 loss: Regression and Classification. *Journal of the American Statistical Association*, 98(462):324–339, 2003.

C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

S. Clémençon, G. Lugosi, and N. Vayatis. Ranking and empirical minimization of U-statistics. *The Annals of Statistics*, pages 844–874, 2008.

S. Clémençon and M. Achab. Ranking data with continuous labels through oriented recursive partitions. In *Advances in Neural Information Processing Systems*, pages 4603–4611, 2017.

S. Clémençon and S. Robbiano. Building confidence regions for the ROC surface. *Pattern Recognition Letters*, 46:67–74, 2014.

S. Clémençon and S. Robbiano. An ensemble learning technique for multipartite ranking. In *Proceedings*, pages 397–402. Presses universitaires de Louvain, 2015a.

S. Clémençon and S. Robbiano. The TreeRank Tournament algorithm for multipartite ranking. *Journal of Nonparametric Statistics*, 27(1):107–126, 2015b.

S. Clémençon and N. Vayatis. Ranking the best instances. *Journal of Machine Learning Research*, 8 (Dec):2671–2699, 2007.

S. Clémençon and N. Vayatis. Tree-structured ranking rules and approximation of the optimal ROC curve. In *Proceedings of the 2008 conference on Algorithmic Learning Theory. Lect. Notes Art. Int*, volume 5254, pages 22–37, 2008.

S. Clémençon and N. Vayatis. Overlaying classifiers: a practical approach to optimal scoring. *Constructive Approximation*, 32(3):619–648, 2010.

S. Clémençon, G. Lugosi, N. Vayatis, P. Aurer, and R. Meir. Ranking and scoring using empirical risk minimization. In *Colt*, volume 3559, pages 1–15. Springer, 2005.

S. Clémençon, M. Depecker, and N. Vayatis. Bagging ranking trees. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 658–663. IEEE, 2009.

S. Clémençon, M. Depecker, and N. Vayatis. Adaptive partitioning schemes for bipartite ranking. *Machine Learning*, 83(1):31–69, 2011.

S. Clémençon, M. Depecker, and N. Vayatis. Ranking forests. *Journal of Machine Learning Research*, 14 (Jan):39–73, 2013a.

S. Clémençon, M. Depecker, and N. Vayatis. An empirical comparison of learning algorithms for nonparametric scoring: the TreeRank algorithm and other methods. *Pattern Analysis and Applications*, 16(4): 475–496, 2013b.

S. Clémençon, S. Robbiano, and N. Vayatis. Ranking data with ordinal labels: optimality and pairwise aggregation. *Machine Learning*, 91(1):67–104, 2013c.

C. Daskalakis, R. M. Karp, E. Mossel, S. J. Riesenfeld, and E. Verbin. Sorting and selection in posets. *SIAM Journal on Computing*, 40(3):597–622, 2011.

A. Dickerson and G. K. Popli. Persistent poverty and children's cognitive development: evidence from the UK millennium cohort study. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 179(2):535–558, 2016.

P. Filzmoser, H. Fritz, and K. Kalcher. *pcaPP: Robust PCA by Projection Pursuit*, 2018. URL `https://CRAN.R-project.org/package=pcaPP`. R package version 1.9-73.

T. Fissler, J. F. Ziegel, and T. Gneiting. Expected shortfall is jointly elicitable with value at risk-implications for backtesting. *arXiv preprint arXiv:1507.00244*, 2015.

Y. Freund and R. 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.

Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4 (Nov):933–969, 2003.

J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

X. Geng, T.-Y. Liu, T. Qin, and H. Li. Feature selection for ranking. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 407–414. ACM, 2007.

T. Gneiting. Making and evaluating point forecasts. *Journal of the American Statistical Association*, 106 (494):746–762, 2011.

M. Gupta and V. Nagadevara. Audit selection strategy for improving tax compliance–Application of data mining techniques. In *Foundations of Risk-Based Audits. Proceedings of the eleventh International Conference on e-Governance, Hyderabad, India, December*, pages 28–30, 2007.

R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. 1999.

W. Hersh, C. Buckley, T. Leone, and D. Hickam. OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *SIGIR'94*, pages 192–201. Springer, 1994.

B. Hofner, L. Boccuto, and M. Göker. Controlling false discoveries in high-dimensional situations: Boosting with stability selection. *BMC Bioinformatics*, 16(1): 144, 2015.

T. Hothorn, P. Bühlmann, T. Kneib, M. Schmid, and B. Hofner. *mboost: Model-Based Boosting*, 2017. URL `https://CRAN.R-project.org/package=mboost`. R package version 2.8-1.

K.-W. Hsu, N. Pathak, J. Srivastava, G. Tschida, and E. Bjorklund. Data mining based tax audit selection: a case study of a pilot project at the Minnesota department of revenue. In *Real world data mining applications*, pages 221–245. Springer, 2015.

T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

V. K. Khanna. Risk-based internal audit in Indian banks: A modified and improved approach for conduct of branch audit. *ICFAI Journal of Audit Practice*, 5(4), 2008.

W. R. Knight. A computer method for calculating Kendall's tau with ungrouped data. *Journal of the American Statistical Association*, 61(314):436–439, 1966.

H. Lai, Y. Pan, C. Liu, L. Lin, and J. Wu. Sparse learning-to-rank via an efficient primal-dual algorithm. *IEEE Transactions on Computers*, 62(6): 1221–1233, 2013a.

H. Lai, Y. Pan, Y. Tang, and N. Liu. Efficient gradient descent algorithm for sparse models with application in learning-to-rank. *Knowledge-Based Systems*, 49: 190–198, 2013b.

T. Lan, W. Yang, Y. Wang, and G. Mori. Image retrieval with structured object queries using latent ranking SVM. In *European conference on computer vision*, pages 129–142. Springer, 2012.

L. Laporte, R. Flamary, S. Canu, S. Déjean, and J. Mothe. Nonconvex regularizations for feature selection in ranking with sparse SVM. *IEEE Transactions on Neural Networks and Learning Systems*, 25 (6):1118–1130, 2014.

H.-T. Lin. *From ordinal ranking to binary classification*. PhD thesis, California Institute of Technology, 2008.

R. D. Luce. Individual choice behavior. 1959.

C. L. Mallows. Non-null ranking models. I. *Biometrika*, 44(1/2):114–130, 1957.

M. Moraru and F. Dumitru. The risks in the audit activity. *Annals of the University of Petrosani. Economics*, 11:187–194, 2011.

L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Boberg, and T. Salakoski. Learning to rank with pairwise regularized least-squares. In *SIGIR 2007 workshop on learning to rank for information retrieval*, volume 80, pages 27–33, 2007.

F. Pan, T. Converse, D. Ahn, F. Salvetti, and G. Donato. Feature selection for ranking using boosted trees. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 2025–2028. ACM, 2009.

K. S. Pickett. *Audit planning: a risk-based approach*. John Wiley & Sons, 2006.

R. L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 24(2):193–202, 1975.

A. Rakotomamonjy. Optimizing area under Roc curve with SVMs. In *ROCAI*, pages 71–80, 2004.

A. Rakotomamonjy. Sparse support vector infinite push. *arXiv preprint arXiv:1206.6432*, 2012.

S. Robbiano. *Méthodes d'apprentissage statistique pour le ranking théorie, algorithmes et applications*. PhD thesis, Télécom ParisTech, 2013.

C. Rudin. The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *Journal of Machine Learning Research*, 10(Oct): 2233–2271, 2009.

C. Rudin and R. E. Schapire. Margin-based ranking and an equivalence between AdaBoost and RankBoost. *Journal of Machine Learning Research*, 10 (Oct):2193–2232, 2009.

B. Schölkopf, R. Herbrich, and A. Smola. A generalized representer theorem. In *Computational learning theory*, pages 416–426. Springer, 2001.

Z. Sun, T. Qin, Q. Tao, and J. Wang. Robust sparse rank learning for non-smooth ranking measures. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 259–266. ACM, 2009.

E. Tsivtsivadze and T. Heskes. Semi-supervised ranking pursuit. *arXiv preprint arXiv:1307.0846*, 2013.

S. Wang, B. Nan, S. Rosset, and J. Zhu. Random lasso. *The Annals of Applied Statistics*, 5(1):468, 2011.

T. Werner. *Gradient-Free Gradient Boosting*. PhD thesis, Carl von Ossietzky Universität Oldenburg, 2019.

H. Yang and D. Carlin. ROC surface: a generalization of ROC curve analysis. *Journal of biopharmaceutical statistics*, 10(2):183–196, 2000.

J. F. Ziegel. Coherence and elicitability. *Mathematical Finance*, 26(4):901–918, 2016.