A review on ranking problems in statistical learning

Tino Werner

November 14, 2019

Ranking problems define a widely spread class of statistical learning problems with many applications, including fraud detection, document ranking, medicine, credit risk screening, image ranking or media memorability. In this article, we systematically describe different types of non-probabilistic supervised ranking problems, i.e., ranking problems that require the prediction of an order of the response variables, and the corresponding loss functions resp. goodness criteria. We discuss the difficulties when trying to optimize those criteria. As for a detailed and comprehensive overview of existing machine learning techniques to solve such ranking problems, we group the suitable techniques into SVM-, tree-, Boosting and Neural Network-type approaches and recapitulate the corresponding optimization problems in a unified notation. We also discuss to which of the ranking problems the respective algorithms are tailored and identify their strengths and limitations. Computational aspects and open research problems are also considered.

Keywords— Ranking problems; Supervised learning; Empirical risk minimization; Structural risk minimization; Surrogate losses

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. [2002]), personality (Anand et al. [2011]) or the 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. Another application of continuous ranking problems is given in 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.

This paper is organized as follows. Starting in section 2 with the definition of several different ranking problems that are distinguished by the nature of the response variable and by the goal of the analyst, it becomes evident that suitable loss functions have at least a pair-wise structure in this case. We describe in detail the loss functions corresponding to the different types of ranking problems and related quality criteria which are optimized especially for ranking problems with discrete response variable. We also carefully discuss the combined ranking problems and distinguish between ranking and ordinal regression. In section 3 we provide a systematic overview of different machine learning algorithms by grouping them into SVM-, tree-, Boosting- and Neural Network-type approaches. We review these approaches and discuss their strengths, limitations and computational aspects. We conclude with open research problems for supervised ranking.

2 Supervised ranking problems

In this work, we always have data \( D = (X, Y) \in \mathbb{R}^{n \times (p+1)} \) 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 \( r : \mathcal{X} \times \mathcal{X} \rightarrow \{-1, 1\} \) 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 \( s : \mathcal{X} \rightarrow \mathbb{R} \) 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 \( \text{Best}_K \) and its empirical counterpart, i.e., the indices of the instances that have been predicted to be the best \( K \) ones, by \( \tilde{\text{Best}}_K \).

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

Further discussions on possible combinations of these types of ranking problems and their relation to classification and regression follow in section 2.5.

### 2.2 Loss functions for supervised ranking

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

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

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

\[
\min_{r \in \mathcal{R}} I_n^\text{hard}(r)
\]

where

\[
I_n^\text{hard}(r) = \frac{1}{n(n-1)} \sum_{i \neq j} I((Y_i - Y_j)r(X_i, X_j) < 0) \tag{2.2}
\]

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 \( I_n^\text{hard} \), i.e., the hard empirical risk, is also the hard ranking loss function which reflects the global nature of hard ranking problems.

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

\[
\min_{\theta \in \Theta} I_n^\text{hard}(\theta)
\]

with

\[
I_n^\text{hard}(\theta) = \frac{1}{n(n-1)} \sum_{i \neq j} I((Y_i - Y_j)(s_\theta(X_i) - s_\theta(X_j)) < 0). \tag{2.3}
\]

For the weak ranking problem, Clémençon and Vayatis [2007] introduce the upper \((1 - u)\)-quantile \( Q(s, 1 - u) \) for the random variable \( s(X) \) for binary responses. Since a weak ranking problem can also be formulated for continuous-valued responses, Werner [2019] introduced the transformed responses

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

where the ranks come from a descending ordering. Then the misclassification risk corresponding to the weak ranking problem in the sense of Clémençon and Vayatis [2007] is given by

\[
R^\text{weak,n}(s) := P(\tilde{Y}(s(X) - Q(s, 1 - u)) < 0)
\]
for the empirical quantile $\hat{Q}(s, 1 - u^{(K)})$. To approximate the $(1 - u)$-quantile, one needs to set $u^{(K)} = K/n$, i.e., for a given quantile $(1 - u)$, one looks at the top $K$ instances that represent this upper quantile.

Remark 2.1. Werner [2019] mention that due to the mass constraint, each false positive generates exactly one false negative, so the loss can be equivalently written as

$$L_{n}^{\text{weak},K}(s) = \frac{1}{n} \sum_{i=1}^{n} I(\tilde{Y}_{i}^{(K)}(s(X_{i}) - \hat{Q}(s, 1 - u^{(K)})) < 0)$$

for the empirical quantile $\hat{Q}(s, 1 - u^{(K)})$. Note that even in the case $\text{loc},s \in \{0, 1\}$—standardized, taking a closer look on it, the maximal achievable loss given a fixed $K$ is

$$\max(n_{\text{loc},K}(s)) = \frac{K(K - 1)}{n(n - 1)} + \frac{n - K}{n} \cdot \frac{2K}{n} =: m_{K},$$

so a standardized version, provided in Werner [2019], is simply

$$L_{n}^{\text{loc},K,\text{norm}}(s) := \frac{1}{m_{K}} L_{n}^{\text{loc},K}(s).$$

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 as mentioned in Werner [2019]. This is true since

$$L_{n}^{\text{loc},n/2}(s) \leq \frac{n - 1}{n(n - 1)} + \frac{1}{n} \cdot \frac{n - 1}{n(n - 1)} = \frac{1}{2} = 1.$$

A simple example for clarification is given below in example 2.1, which we borrow from Werner [2019]. We insist to once more take a look on the $U$-statistics that arise for the hard and the localized ranking problem. Cléménon 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

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

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

$$L_{n}^{\text{hard}}(\pi, \hat{\pi}) = \frac{2}{n(n - 1)} \sum_{i<j} I((\pi_{i} - \pi_{j})(\hat{\pi}_{i} - \hat{\pi}_{j}) < 0).$$

Example 2.1. Assume that we have a data set with the true response values
\begin{itemize}
\item $Y := (-3, 10.3, -8, 12, 14, -0.5, 29, -1.1, -5.7, 119)$
\item and the fitted values
\[ \hat{Y} := (0.02, 0.6, 0.1, 0.47, 0.82, 0.04, 0.77, 0.09, 0.01, 0.79). \]
\end{itemize}

Then we order the vectors according to $Y$, so that $Y_1 \geq Y_2 \geq \ldots$ and get the permutations
\[ \pi = (1, 2, \ldots, 10), \quad \hat{\pi} = (2, 3, 1, 5, 4, 8, 7, 9, 10, 6). \]

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

The standardized weak ranking loss is then
\[ L_n^{\text{weak},4,norm}(\pi, \hat{\pi}) = \frac{10}{8} \cdot \frac{2}{10} = 0.25 \]

which is most intuitive since one of the indices of the four true best instances is not contained in the predicted set $\text{Best}_4$. The second part of the localized loss is then
\[ \frac{2}{90} |0 + 1 + 0 + 0 + 0 + 0| = \frac{2}{45}. \]

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

The standardized localized ranking loss is then
\[ L_n^{\text{loc},4,norm}(\pi, \hat{\pi}) = \frac{75}{46} \cdot \frac{37}{225} \approx 0.268. \]

Finally, the hard ranking loss is
\[ L_n^{\text{hard}}(\pi, \hat{\pi}) = \frac{2}{90} \cdot 8 = \frac{16}{90}. \]

Setting $K = 5$, the weak ranking loss is zero and the localized ranking loss is
\[ L_n^{\text{loc},5}(\pi, \hat{\pi}) = \frac{2}{90} |0 + 1 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 1| + \frac{5}{10} \cdot 0 = \frac{1}{15}. \]

The standardized localized ranking loss is
\[ L_n^{\text{loc},5,norm}(\pi, \hat{\pi}) = \frac{18}{13} \cdot \frac{1}{15} \approx 0.092. \]

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.

### 2.3 Fast computation of the hard ranking loss

A naive evaluation of the hard ranking loss requires $\mathcal{O}(n^2)$ comparisons. This will surely become infeasible for data sets with many observations, therefore Werner [2019] provided a solution which comes up with $\mathcal{O}(n \ln(n))$ evaluations.

They take a look at the concordance measure
\[ \tau(Y, \hat{Y}) := \frac{1}{n(n-1)} \sum_{i \neq j} \text{sign}(Y_i - Y_j)(\hat{Y}_i - \hat{Y}_j) \]

called Kendall’s Tau. Unlike the ranking loss which is high if there are many misrankings and which is in $[0, 1]$—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 as given in the following lemma from Werner [2019]. We also recapitulate its proof.

**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
\[ L_n^{\text{hard}}(x, y) = \frac{1 - \tau(x, y)}{2}. \]

**Proof.** In the case of a perfect concordance, the ranking loss function is zero whereas Kendall’s $\tau$ takes the value 1. If we produce one misranking, w.l.o.g. by swapping the largest and the second largest entry $x_i$, resp. $x_j$, 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 $\tau$ changing from 1 to -1, decreasing it by $\frac{4}{n(n-1)}$.

By induction, the claimed formula is valid. \qed
This indeed turns out to be useful in practice since there exists an \( R \)-command that provides fast computation of Kendall’s \( \tau \), namely the command \texttt{cor.fk} from the package \texttt{pcaPP} \cite{Filzmoser2018}. The algorithm essentially goes back to Knight (1966) and relies on the idea of fast ordering algorithms. So in fact, they first compute Kendall’s Tau using \texttt{cor.fk} and then, they 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 \( O(n^2) \) in the na"ive implementation to \( O(n \ln(n)) \).

2.4 Quality criteria for ranking

So far, we presented loss functions for ranking problems that lead to algorithms in the spirit of the ERM resp. SRM paradigm. On the other hand, there also exist quality measures that are popular in classification settings but which already have been transferred to the ranking setting. Before we go into detail, we recapitulate the definition of a common and well-known quality criterion for classification.

Definition 2.2. Let \( Y_1, \ldots, Y_n \) take values in \( \{-1, 1\} \) where the total number of positives is \( n_+ \) and the total number of negatives is \( n_- \). Let \( \hat{Y}_i \in \{-1, 1\}, \ i = 1, \ldots, n \), be predicted values.

\begin{enumerate}[a)]
\item The \textbf{true positive rate (TPR)} and the \textbf{false positive rate (FPR)} are given by
\[
\text{TPR} = \frac{1}{n_+} \sum_{i} I(\hat{Y}_i = 1) I(Y_i = 1),
\]
\[
\text{FPR} = \frac{1}{n_-} \sum_{i} I(\hat{Y}_i = 1) I(Y_i = -1).
\]
\item The \textbf{Receiver Operation Characteristic Curve (ROC Curve)} is the plot of the true positive rate against the false positive rate.
\item The \textbf{AUC} is the abbreviation for the \textit{area under the ROC curve}.
\end{enumerate}

For theoretical aspects of the empirical AUC and its optimization, we refer to Agarwal et al. \cite{Agarwal2005}, Cortes and Mohri \cite{Cortes2004} and Calders and Jaroszewicz \cite{Calders2007}. We continue presenting the reparametrization of the ROC curve as it has been introduced in Clémençon et al. \cite{Clemencon2008} and used in subsequent papers of Clémençon and coauthors.

Definition 2.3. For a scoring function \( s \), the true positive rate and the false positive rate are given by

\[
\text{TPR}(x) = P(s(X) \geq x \mid Y = 1), \quad \text{FPR}(x) = P(s(X) \geq x \mid Y = -1).
\]

Setting

\[
q_{s, \alpha} := \inf\{x \in [0, 1] \mid \text{FPR}(x) \leq \alpha\},
\]

the \textit{ROC curve} is the plot of \( \text{TPR}_s(q_{s, \alpha}) \) against the level \( \alpha \).

The ROC curve is a standard tool to validate binary classification rules. If the classification depends on a threshold, different points of the ROC curve are generated by changing the threshold and calculating the TPR and the FPR. Since the goal is to achieve a TPR as high as possible for the price of a FPR as low as possible, one usually chooses the threshold corresponding to the upper-leftmost point of the empirical ROC curve. A combined quality measure that incorporates all points of the ROC curve is the AUC where a classification rule is better the higher the empirical AUC is. Random guessing clearly has a theoretical AUC of 0.5.

For the bipartite localized ranking problem, Clémençon and Vayatis \cite{Clemencon2007} provide the following localized version of the AUC. It is important to note a strong equivalence between the AUC and the ranking error \( P((Y - Y')(s(X) - s(X')) < 0) \) in the sense that minimizing this error is equivalent to maximizing the AUC corresponding to the scoring function \( s \) (see Clémençon and Vayatis \cite{Clemencon2007}).

Definition 2.4. The \textbf{localized AUC} is defined as

\[
\text{LocAUC}(s, \alpha) := P(s(X) > s(X') \mid s(X) \geq Q(s, 1 - \alpha) \mid Y = 1, Y' = -1).
\]

As for \( d \)-partite ranking problems, i.e., \( Y \) can take \( d \) different values, Clémençon et al. \cite{Clemencon2013a} proposed the \textit{VUS} (volume under the ROC surface) as quality criterion.

Definition 2.5. Let \( w.l.o.g. \ Y \) take values in \( \{1, \ldots, d\} \) and let again \( X \) take values in \( X \subset \mathbb{R}^p \). For a scoring function \( s : X \to \mathbb{R} \), define

\[
F_{s,k}(t) := P(s(X) \leq t \mid Y = k)
\]
for $k = 1, \ldots, d$.
a) The ROC surface is the "continuous extension" (Clémençon et al. [2013c]) of the plot
\[
(t_1, \ldots, t_{d-1}) \mapsto (F_{s,1}(t_1), F_{s,2}(t_2) - F_{s,2}(t_1), \ldots, 1 - F_{s,d}(t_{d-1}))
\]
for $t_1 < t_2 < \ldots < t_{d-1}$.

b) The VUS is the volume under the ROC surface.

In this definition, the term "continuous extension" means to connect the points by hyperplane parts as described in Clémençon et al. [2013c]. The ROC surface can be interpreted as joint plot of the class-wise true positive rates since if the value of the scoring function is between $t_k$ and $t_{k+1}$ (artificially define $t_0 := -\infty$ and $t_d := \infty$), the instance is assigned to class $(k + 1)$.

Other well-known quality criteria for ranking problems are for example the MAP (mean average precision) and the NCDG (normalized discounted cumulative gain), but since we focus on losses in this article, we do not consider algorithms that optimize such quality criteria that are not directly related to a ranking loss like the AUC.

2.5 Discussion of the different ranking problems

In this subsection, we recapitulate a discussion from Werner [2019] of the different types of ranking problems introduced earlier from a qualitative point of view and the differences between ranking and 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 $i_1, \ldots, i_{m_k}$ that belong to class $k$. If for a scoring rule $s$ we had the predicted ordering $s(X_{i_1}) < s(X_{i_2}) < \ldots < s(X_{i_{m_k}})$ 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 $s(X_i) \notin [s(X_{i_1}), s(X_{i_{m_k}})] \forall i \notin \{i_1, \ldots, i_{m_k}\}$.

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 will describe in subsection 3.2 and localized bipartite ranking problems can be solved using the tree-type algorithms of Clémençon as pointed out for instance.
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 (Clémençon and Vayatis [2007]). In the case of weak bipartite ranking problems, it may sound 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.

2.6 Other ranking problems

In this subsection, we want to clarify the scope of this review by briefly referring to examples of probabilistic resp. unsupervised ranking problems.

First, probabilistic approaches predict a probability distribution on the set of permutations of the instances, i.e., the set $\text{Perm}(1:n)$. 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 we encounter in this review are different.

Unsupervised ranking refers to the situation where instances $X_1,...,X_n \in \mathcal{X}$ are given and should be ranked according to their degree of anomaly as described in Clémençon and Robbiano [2014], therefore they refer to it as anomaly ranking. They propose a so-called mass-volume curve as quality criterion for the unsupervised ranking problem and show in Clémençon and Robbiano [2014, Thm. 1] under which conditions the ROC curve is related to the MV curve in terms of a bijection. They extend their TreeRank algorithm for supervised binary ranking problems (see Subsection 3.2) to this case. See also Goix et al. [2015], Clémençon et al. [2015] and Clémençon and Thomas [2017] as well as references therein for further details on anomaly ranking.

The PageRank algorithm (Page et al. [1999]) can also be identified with unsupervised ranking since it does not invoke any response variable but is based on a graphical model including an adjacency matrix that represents the links connecting the different websites.

3 Current techniques to solve ranking problems

This section is divided into four parts. Each subsection is devoted to a particular underlying machine learning algorithm for the discussed ranking approach, i.e., Support Vector Machines (SVM), trees, Boosting and Neural Networks resp. Deep Learning.

3.1 SVM-type approaches

Joachims [2002] provided the RankingSVM algorithm for document retrieval which is essentially based
on a similar approach for ordinal regression introduced in Herbrich et al. [1999]. In their situation, a set $D = (X_1, ..., X_n)$ of documents is given. The goal is that for a given query, a scoring function $s$ has to be computed such that the ordering of the documents according to the scoring function is as concordant as possible with the true ordering according to the relevance of the documents w.r.t. the query.

Their goal is to solve a hard bipartite ranking problem, but they do not formulate the hard ranking loss but the constraint inequalities in the sense that $s(X_i) > s(X_j)$ for $X_i$ being more relevant than $X_j$, given each of the queries. As they argue, trying to find a scoring function such that every inequality is satisfied would be NPhard, so they introduce slack variables and formulate the problem as a standard SVM problem but with all the relaxed inequalities as constraints, so that one gets a standard SVM-type solution $s(x) = \sum_i \alpha_i K(x, X_i)$ for a kernel $K$ as identified in Clémençon et al. [2013b].

Due to the equivalence of SVM problems and structural risk minimization problems with a Hinge loss, Clémençon et al. [2013b] translated the criterion in Joachims [2002] into the regularized pair-wise empirical loss

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

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

An implementation of RankingSVM is given as $SVMlight$ software package [Joachims 1999] in C language[1] as well as an improved implementation in the software package $SVMrank$ relying on the cutting-plane algorithm from Joachims [2006]. As for the computation of the solutions, note that Chapelle and Keerthi [2010] argued that the $SVMlight$ implementation for RankingSVM requires the computation of all pairwise differences $X_i - X_j$ which leads to a complexity of $O(n^2)$. They propose a truncated Newton step which is computed via conjugate gradients in order to remedy this issue and result with the MATLAB implementation $PRSVM$[2] essentially reducing the respective complexity to $O(np)$ for $n > p$.

They propose two methods to get a suitable kernel approximation. The first is a Nyström approximation where $m \ll n$ rows of $X$, say, $\hat{X}_1, ..., \hat{X}_m$, are sampled uniformly, followed by a singular value decomposition of the matrix $(K(\hat{X}_i, \hat{X}_j))_{i,j=1,...,m}$. Truncating the SVD by taking just the first $k$ columns of the orthonormal matrix and the upper left $k \times k$-submatrix of the diagonal matrix, one gets a rank-$k$-approximation, reducing the complexity to $O(npk + k^3)$. Another strategy is to Fourier transform the kernel, i.e.,

$$K(x, x') = \int q(\omega) \exp(i \omega^T (x - x')) d\omega,$$

and to draw $m$ samples according to $q$, providing a kernel approximation using Bochner’s theorem. Despite the approximation error is higher than for the Nyström approximation (for equal $m$), the complexity is just $O(np)$. Chen et al. [2017] provide publicly available MATLAB code[3].

Rakotomamonjy [2004] and Ataman and Street [2005] use the fact that the binary hard ranking problem can be solved by maximizing the AUC of the scoring function. Since the responses are binary-valued, Rakotomamonjy [2004] explicitly distinguishes between positive and negative instances by writing $X^+ - X^-$ for the features. The empirical AUC can be estimated by

$$\hat{AUC} = \frac{1}{n_{+} n_{-}} \sum_{i=1}^{n_{+}} \sum_{j=1}^{n_{-}} I(s(X^+_i) > s(X^-_j)) = \frac{1}{n_{+} n_{-}} \sum_{i=1}^{n_{+}} \sum_{j=1}^{n_{-}} I(\xi_{ij} > 0)$$

for $\xi_{ij} := s(X^+_i) - s(X^-_j)$. Using the definition of $\xi_{ij}$ as equality constraint, Rakotomamonjy [2004] formulate the problem as SVM-type problem by considering linear scoring functions $s(x) := w^T x + b$. They show that the solution essentially has the form

$$s(x) = \sum_{i} \sum_{j} \alpha_{ij} (K(X^+_i, x) - K(X^-_j, x)) + b.
in the general case when using kernels. In Rakotomamonjy [2004], the algorithm is applied to different data sets, including a cancer and a credit data set. They conclude that their algorithm also provides good accuracy performances. Ataman and Street [2005] try a MATLAB and a WEKA implementation, the algorithm from Rakotomamonjy [2004] can be found in a MATLAB toolbox [4 Canu et al. 2005].

Brefeld and Scheffer [2005] provide a very similar approach, but they both provide a so-called “1-Norm” and “2-Norm” problem, namely

\[ \frac{1}{2}||w||^2 + C \sum_i \sum_j \xi_{ij} \]

for the target function of the SVM, where \( r \in \{1, 2\} \), and the corresponding solutions. A recommendation for the choice of \( r \) is however not given. Due to the evaluation of the kernel matrix and the quadratically growing number of constraints, their algorithm is of complexity \( O(n^4) \). They however make some suggestions how to reduce the complexity.

Cao et al. [2006] argue that a major weakness of RankingSVM is that misranking on the top of the list get the same loss as misrankings at the bottom. Therefore, they propose a weighted variant of the Hinge loss in the sense that the weights are higher the higher the importance of the documents and the queries is. They apply their algorithm to the OHSUMED data set.

Jung et al. [2011] provide Ensemble RankingSVM by combining different RankingSVM models.

Tian et al. [2011] consider essentially the same problem as Rakotomamonjy [2004], but with the crucial difference that the target function is

\[ ||w||_p^p + C \sum_i \sum_j \xi_{ij} \]

for \( 0 < p < 1 \), so \( ||w||_2^2 \) has been replaced by a concave loss. They solve the problem with a multi-stage convex relaxation technique. They conclude that by the \( l_p \)-norm, the algorithm indeed performs feature selection which results from the equivalence to write an SVM problem as a regularized problem with the Hinge loss. Since the number of constraints grows quadratically with the number of observations, they propose to cluster the observations first and to just perform the computations on the representatives.

Another approach is given in Lai et al. [2013a] where they replace the quadratic penalty (i.e., \( ||w||_2^2 \) in the equivalent formulation) with an \( l_1 \)-regularization term and use the squared Hinge loss. They solve the problem by invoking Fenchel duality (hence the name FenchelRank) and prove convergence of the solution. After experiments on real data sets for document retrieval, they conclude sparsity of the solutions as well as superiority of FenchelRank to non-sparse algorithms. They implement their method in MATLAB. 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 analogous 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. They provide publicly available MATLAB code.

Another approach that does not provide an SVM-type solution at the first glance is given in Pahikkala et al. [2007]. They intend to predict the differences of the responses by the differences of the scores assigned to the respective features, i.e., to essentially solve

\[ \frac{1}{n(n-1)} \sum_i \sum_j \frac{1}{2} \text{sign}(s(X_i) - s(X_j)) - \text{sign}(Y_i - Y_j) \]

\[ + \lambda ||s||_2^2 \]

for some function \( s : X \rightarrow \mathbb{R} \) and some kernel \( K \) with corresponding RKHS \( \mathcal{H}_K \). Since this problem is clearly not tractable, as Pahikkala et al. [2007] point out, they instead minimize the regularized least-squares-type cri-

\[ \text{http://asi.insa-rouen.fr/enseignants/arakoto/toolbox/} \]

\[ \text{http://remi.flamary.com/soft/soft-ranksvm-nc.html} \]
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 \text{RankRLS} ("regularized least squares"). The complexity of the algorithm is of order \( O(p^3 + np^2) \) resulting from matrix inversion and matrix multiplication. Note that Pahikkala et al. [2010] provided a greedy method to compute the respective inverse by successively selecting up to \( k < p \) features which results in an overall complexity of their greedy \text{RankRLS} algorithm of \( O(knp) \). Pahikkala et al. [2010] provided a link leading to implementations of both \text{RankRLS} and greedy \text{RankRLS}, but it does not seem to be available anymore.

Summarizing, there exist a rich variety of SVM-type ranking algorithms in order to minimize the hard ranking loss, including approaches that provide sparse solutions. The approach of Cao et al. [2006] minimizes a weighted hard ranking loss and can be seen as the closest SVM-type approach for localized ranking problems. However, note that the algorithms are tailored to bipartite ranking problems. Furthermore, SVM solutions are in general hard to interpret. In contrast to the AUC-maximizing approaches, the other algorithms make use of a surrogate loss function for the hard ranking loss which is either a pair-wise Hinge or pair-wise squared loss.

### 3.2 Tree-type approaches

Clémençon and coauthors provide tree-type algorithms which turn out to be an impressively flexible class of ranking algorithms that can be applied to all hard ranking problems as well as to localized binary ranking problems.

As for binary ranking problems, they provided \text{TreeRank} and \text{RankOver} (Clémençon and Vayatis [2008], Clémençon and Vayatis [2010]). The idea behind the \text{TreeRank} 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 \((0, 0)\) and \((1, 1)\)) as shown in Clémençon and Vayatis [2008, Prop. 13]. The \text{TreeRank} algorithm then recursively 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. More precisely, one starts with the region \( C_{0,0} = \mathcal{X} \) and the coefficients \( \alpha_{0,1} = \beta_{0,1} = 1 \). In each stage \( d = 0, ..., D-1 \) of the tree and in every iteration \( k = 0, ..., 2^d - 1 \), one computes the estimates
\[ \hat{\alpha}(C_{d,k}) := \frac{1}{n} \sum_i I(X_i \in C_{d,k}, Y_i = -1) \]
\[ \hat{\beta}(C_{d,k}) := \frac{1}{n} \sum_{i} I(X_i \in C_{d,k}, Y_i = 1) \]

and optimizes the entropy measure

\[ \text{Ent}(C_{d,k}) := (\alpha_{d,k+1} - \alpha_{d,k}) \hat{\beta}(C_{d,k}) - (\beta_{d,k+1} - \beta_{d,k}) \alpha(C_{d,k}) \]

by finding a subset of \( C_{d,k} \) which maximizes this empirical entropy. The coefficients are updated recursively.

Similarly, the \text{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. However, the authors seem to prefer \text{TreeRank} over it since their subsequent algorithms are based on the former, so we do not explain more details of \text{RankOver}.

\cite{cl2006} already mention that \text{TreeRank} may be used as weak ranker for a Boosting-type approach.

Extensions by combining the \text{TreeRank} algorithm in combination with bagging resp. in a \text{RandomForest} and optimizes the entropy measure

\[ \text{Ent}(C_{d,k}) := (\alpha_{d,k+1} - \alpha_{d,k}) \hat{\beta}(C_{d,k}) - (\beta_{d,k+1} - \beta_{d,k}) \alpha(C_{d,k}) \]

As for the pruning of ranking trees, we refer to \cite{cl2011} and \cite{cl2013} who recommend to use the penalized empirical AUC as pruning criterion, i.e., for a tree \( T \), one selects the subtree \( T_{\text{sub}} \) which maximizes

\[ \hat{AUC}_{s_{\text{sub}}} - \lambda |T_{\text{sub}}| \]

where \( s_T \) denotes the scoring function corresponding to tree \( T \).

The \text{TreeRank} algorithm has been available in the \( R \)-package \text{TreeRank}, but it had been removed. Nevertheless, the source code is still available.

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 \cite{cl2010}, 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 \text{TreeRank} algorithm and twice differentiability with bounded second derivatives for the \text{RankOver} algorithm.

These tree-type algorithms are tailored to bipartite ranking problems. However, as pointed out in \cite{cl2013}, they can be used for local AUC optimization (see Def. 2.4), so they are applicable for both hard and localized bipartite ranking problems while the AUC-maximizing competitors show inferior local ranking performance in the simulation studies of \cite{cl2013}.

As for the \( d \)-partite ranking problems, \cite{cl2013} argued that they can be regarded as collection of bipartite ranking problems if one considers approaches like one-versus-all or one-versus-one. In \cite{cl2013}, they apply different algorithms tailored to bipartite ranking problems like \text{TreeRank}, \text{RankBoost} or \text{RankingSVM} and evaluate their performance in the VUS criterion.

However, since the algorithms are not designed for VUS-optimization, \cite{cl2015} modify their \text{TreeRank} algorithm such that the splits a each node are performed first in a one-versus-one sense (but only for adjacent classes) and then the optimal split of them is selected according to the VUS criterion. The resulting \text{TreeRankTournament}
algorithm therefore solved the hard $d$-partite ranking problem. [Clémençon and Rohbian 2015a] provide a bagged and a RandomForest-type version of this algorithm, analogously to the bipartite case.

Recently, [Clémençon and Achab 2017] provided pioneer work for the hard continuous ranking problem which did not have been considered so far. Let w.l.o.g. $Y \in [0,1]$. Then each subproblem

$$\max_y \{ P(s(Y) > t | Y > y) - P(s(Y) > t | Y < y) \}$$

for $y \in [0,1]$, i.e., $s(Y)$ given $Y > y$ should be stochastically larger than $s(Y)$ 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

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

where 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 $\tau$ as a performance measure for continuous ranking.

The approach presented in [Clémençon and Achab 2017] manifests itself in the tree-type $\text{CRank}$ 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.

[Clémençon and Achab 2017] already announced a forthcoming paper where a RandomForest-type approach for $\text{CRank}$ will be presented.

All these tree-type approaches focus on a sophisticated optimization of the AUC or another appropriate criterion. For the price of getting models that are difficult to interpret, these techniques are very flexible and are applicable to the most types of ranking problems.

### 3.3 Boosting-type approaches

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 (see e.g. Bühlmann and Van De Geer 2011), i.e., minimizing the loss

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

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 it furthermore just optimizes the ROC curve in an $L_1$-sense as pointed out in [Clémençon and Vayatis 2008, Clémençon and Vayatis 2010]. Taking a closer look on this loss function, it is indeed a convex surrogate of the misclassification loss and does not respect a pair-wise structure. 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 cannot be optimal. As mentioned in [Clémençon et al. 2013b], a kernel logistic regression may also be thinkable in the same plug-in sense (which has the same weaknesses).

Freund et al. 2003 developed a Boosting-type algorithm (RankBoost) which combines weak rankers in an AdaBoost-style (for the latter, see Freund and Schapire 1997) benefitting from the binarity of the response variable. First, they propose a distribution $D$ on the space $X \times X$ which, for data $D$, is represented as a matrix that essentially contains weights. These weights can be thought of representing the importance to rank the corresponding pair correctly. As for the weak rankers which are nothing but a scoring function $\tilde{s}$, they consider either the identity function or a function that maps the features essentially into the set $\{0,1\}$ according to some threshold. More precisely, the weak ranker is chosen such that the quality measure

$$\sum_{i \neq j \in \{X_i, X_j\}} D(X_i, X_j)(\tilde{s}(X_i) - \tilde{s}(X_j))$$
is maximized where \( r \) again denotes the ranking rule introduced in Definition 2.1 meaning that the sum runs over all pairs \((X_i, X_j)\) where \(X_i\) is ranked higher than \(X_j\). As the AdaBoost algorithm minimizes the exponential surrogate of the 0/1-classification loss, Clémençon et al. [2013b] pointed out that RankBoost minimizes the pair-wise surrogate loss function
\[
\frac{1}{n(n-1)} \sum_{i < j} \exp(-(Y_i - Y_j)(s(X_i) - s(X_j))).
\]
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.

It is shown in Rudin and Schapire [2009] that in the case of binary outcome variables, RankBoost and the classifier AdaBoost are equivalent under very weak assumptions. Therefore, RankBoost can also be seen as an AUC maximizer in the bipartite ranking problem. Freund et al. [2003] apply RankBoost for document retrieval. The algorithm is available at the RankLib library (Dang [2013]).

An extension of RankBoost has been provided in Rudin [2009]. They intend to optimize essentially
\[
\frac{2}{n(n-1)} \sum_i \left( \sum_{j>i} \exp((Y_i - Y_j)(s(X_i) - s(X_j))) \right)^p
\]
for some \( p \geq 1 \) (Rudin [2009] originally distinguish positive and negative instances, but Clémençon et al. [2013b] used the notation as in the display above). The argument behind this power loss given in Rudin [2009] is that the higher \( p \) is chosen, the higher the difference between the loss of misrankings at the top of the list and misrankings at the bottom of the list becomes. The algorithm parallels the RankBoost algorithm in combining weak rankers, but since the weights are not always analytically computable, they may use a linesearch. They call their algorithm \( p \)-Norm-Push. The case \( p = \infty \) has been studied in Rakotomamonjy [2012].

So, while RankBoost is tailored to hard bipartite ranking problems (and may be used for \( d \)-partite ranking problems in the sense of Clémençon et al. [2013b]), the \( p \)-Norm-Push is closest to handle localized bipartite ranking problems. However, the results of the simulation study in Clémençon et al. [2013b] reveal that the localized AUC criterion for the corresponding predictions is not better than for RankBoost. To the best of our knowledge, the \( p \)-Norm-Push has never been applied to \( d \)-partite ranking problems.

Another generalization of RankBoost has been proposed in Zheng et al. [2008], again for hard bipartite ranking problems. They suggest to use a sufficiently regular surrogate of the ranking loss like a squared or a squared Hinge loss and to apply Gradient Boosting (Friedman [2001], Bühlmann and Hothorn [2007]) to this surrogate loss. As weak learner, they consider a so-called “regression weak learner” to fit the gradients in each iteration. They apply their algorithm to document retrieval data.

In contrast to the already reviewed Boosting-type approaches which are designed for bipartite ranking problems, Werner [2019] argue that in the context of risk-based auditing (see e.g. Alm et al. [1993], Gupta and Nagadevara [2007], Hsu et al. [2015]), it is more reasonable to solve a continuous ranking problem. The risk-based auditing context is in fact an 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 one states a binary ranking problem, one 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 one sets up a continuous ranking problem where the amount of damage is the variable of interest, one 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 Boosting-type and most of the SVM-type approaches that we reviewed so far invoke surrogate losses of the hard ranking loss (or even of the 0/1-classification loss). It is discussed in Werner (2019) whether an analogous approach is appropriate for a Gradient Boosting algorithm (see e.g. Bühlmann and Hothorn 2007) for the hard continuous ranking problem. They conclude that due to the support of the response variable which is no longer just \{-1,1\} or some finite set as in the \(d\)--partite ranking problem, exponential or Hinge surrogates would dramatically fail to be meaningful surrogates for the hard ranking loss. Another weakness would be the necessity to evaluate the gradients of the pair-wise loss (which are sums themselves) in each Boosting iteration, making the algorithm computationally expensive.

To handle these issues, Werner and Ruckdeschel 2019 proposed a so-called "gradient-free Gradient Boosting" approach to make Gradient Boosting accessible to non-regular loss functions like the hard ranking loss. Their approach is based on \(L_2\)--Boosting with component-wise linear baselearners (Bühlmann 2006) which minimizes the squared loss by successively selecting the simple linear regression model, i.e., the linear regression model based on one single column, that minimizes the squared loss w.r.t. the resulting combined model most. Werner and Ruckdeschel 2019 propose to alternatingly perform \((M-1)\) of these standard iterations for some \(M > 1\) and one "singular iteration" where the linear baselearner which improves the hard ranking loss of the combined model most is selected.

However, despite they derive estimation and consistency results for their "SingBoost" algorithm (based on similar theorems in Bühlmann 2006 resp. Bühlmann and Van De Geer 2011), they discuss that the resulting Boosting solution suffers from overfitting (as Gradient Boosting solutions without early stopping generally do) and that the predictor set corresponding to the solution is not stable. They argue that a combination with a Stability Selection (Meinshausen and Bühlmann 2010, Hofner et al. 2013) would be necessary which is outlined in Werner 2019. Nevertheless, the approach presented in Werner 2019 is mainly designed for the hard continuous ranking problem.

As for the computational complexity of SingBoost, they show that using the formula from Lemma 2.1 which is applicable since ties occur with probability zero for continuous-valued response variables, the algorithm requires \(O(mn\ln(n)p)\) computations for the number \(m\) of Boosting iterations which is only insignificantly more than the complexity of \(L_2\)--Boosting of \(O(mnp)\). Their algorithm is implemented in R, though not yet publicly available.

3.4 Approaches with neural networks and Deep Learning

Burges et al. 2005 suggest to define a pair-wise variant of the cross-entropy loss as surrogate for the hard ranking loss. More precisely, their pair-wise cross-entropy loss is given by

\[
L_{ij}^{CE}(s) := -p_{ij} \ln(\hat{p}_{ij}) - (1 - p_{ij}) \ln(1 - \hat{p}_{ij})
\]

where

\[
\hat{p}_{ij} := \frac{\exp(\hat{s}(X_i) - \hat{s}(X_j))}{1 + \exp(\hat{s}(X_i) - \hat{s}(X_j))}
\]

and where \(p_{ij}\) is the analog for the theoretical differences. From a probabilistic point of view, the \(p_{ij}\) are interpreted as posterior probabilities that instance \(i\) is ranked higher than instance \(j\). The main contribution of Burges et al. 2005 is to generalize the back-propagation algorithm used when fitting neural networks.

They propose a two-layer neural network and define the following pair-wise linear combination of features:

\[
s(X_i) := h^{(3)} \left( \sum_j w_{ij}^{(32)} h^{(2)} \left( \sum_k w_{ijk}^{(21)} X_k + b_j^{(2)} \right) + b_i^{(3)} \right).
\]

The \(h^{(i)}\) are considered to be activation functions. The back-propagation algorithm then is based on the partial derivatives of \(s\) w.r.t. the weights resp. the offsets.

Again, this RankNet algorithm is tailored to the hard bipartite ranking problem and the experiments in Burges et al. 2005 are based on document retrieval data. It is available at the RankLib library (Dang
In fact, Song et al. [2016] apply their results to bipartite hard ranking problems. Their work is based on Hazan et al. [2010] who proved that

\[ \nabla_\theta \mathbb{E}[L(Y, s_\theta(X))] = \lim_{\epsilon \to 0} \left( \frac{1}{\epsilon} \mathbb{E}[\nabla_\theta F(X, Y_{\text{direct}}, \theta) - \nabla_\theta F(X, Y, \theta)] \right) \]

where

\[ Y_{\text{direct}} = \arg\max_{\tilde{Y}} (F(X, \tilde{Y}, \theta) \pm \epsilon L(Y, \tilde{Y})) \]

and

\[ Y_\theta := \arg\max_{\tilde{Y}} (F(X, \tilde{Y}, \theta)) \]

for some function \( F \) that is linear in \( \theta \). Song et al. [2016] extend this result for non-linear and non-convex functions. In fact, Song et al. [2016] apply their results to bipartite ranking problems by setting

\[ F(X, Y, \theta) := \frac{1}{n-n_+} \sum_i r(X_i, X_j)(\tilde{s}(X_i, \theta) - \tilde{s}(X_j, \theta)) \]

for the ranking rule introduced in Def. 2.1 and by invoking the loss function

\[ L(Y, \tilde{Y}) := 1 - \frac{1}{n_+} \sum_{j:rk(\tilde{Y}) \leq j} \frac{1}{n_+} \sum_i I(rk(\tilde{Y}_i) \leq j)I(Y_i = 1) \]

where \( \tilde{s} \) is the scoring function that is learned by the Deep Neural Network. Song et al. [2016] prove how their theoretical results can be applied to the case with the given functions \( F \) and \( L \) and show that a back-propagation strategy with a suitable Bellman recursion is available.

The approach in Song et al. [2016] is designed for bipartite hard ranking problems. Engilberge et al. [2019] propose to use Deep Learning and essentially combine two Deep Neural Networks. They discuss several smooth surrogate losses, for example, for losses corresponding to Spearman correlation, Mean Average Precision or Recall at \( K \) and argue that since they are all rank-based, i.e., depend on \( \text{rk}(Y) \) and \( \text{rk}(\tilde{Y}) \), it is hard to optimize them due to non-differentiability. Therefore, they propose to invoke a real-valued scoring function such that the fitted scoring function \( \hat{s}_1 \) approximates the true ranking vector \( \text{rk}(Y) \) as best as possible by considering the \( L_1 \)-loss function

\[ \frac{1}{n} ||s_1(X) - \text{rk}(Y)||_1. \]

According to Engilberge et al. [2019, Sec. 3.2], \( \hat{s}_1 \) needs to be trained on synthetic training data, using a sorting Deep Neural Network.

Having real-valued scores, they propose the surrogate loss

\[ \sum_i ||\hat{s}_1(s_2(X_i)) - \text{rk}(Y_i)||_2 \]

for a loss based on Spearman’s correlation and in the case of multilabel responses with classes \( 1, \ldots, d \), they propose the surrogate loss

\[ \sum_{i=1}^d \langle \hat{s}_1(s_2(Y_i)), Y_i \rangle \]

based on the Mean Average Precision, where \( Y_i \) is a binary vector with ones where the respective component of \( Y \) is from class \( l \) and where \( s_2(Y)_l \) is considered to be the score vector for class \( l \). They also propose a surrogate for the Recall at \( K \) criterion. The scoring function \( \hat{s}_2 \) is again computed using a Deep Neural Network. Engilberge et al. [2019] call their approach SoDeep and apply it to media memorability, image classification and cross-modal retrieval tasks, each task corresponding to one of their three surrogate losses. In fact, SoDeep is applicable to hard and localized (the latter with the surrogate for Recall at \( K \)) ranking problems and does not make requirements for \( Y \). On the other hand, the solution, as well as the one from Song et al. [2016], suffers from the common disadvantages of Deep Neural Networks, i.e., they do not perform variable selection and are very difficult to interpret.

4 Conclusion and outlook

We provided a systematic review of different ranking problems, concerning both the type of the response variable and the goal of the analysis. We analyzed and discussed the corresponding loss functions resp. quality criteria and carefully described the nine
combined ranking problems.

Section 3 contains a detailed review of existing learning algorithms for supervised ranking based on the empirical resp. structural risk minimization principle in a unified notation, grouped by the underlying machine learning algorithm.

Note that there exist other kinds of ranking problems and approaches, for example unsupervised ranking or ranking based on the maximization of some quality criterion like NDCG that cannot be directly translated to one of the ranking losses we described. Furthermore, we restricted ourselves to batch learning approaches.

4.1 Open problems

Despite there exist a vast variety of approaches to solve ranking problems, most of the current approaches are either designed for discrete- or for continuous-valued response variables. Additionally, nearly all of the reviewed techniques require an appropriate surrogate loss function for one of the ranking losses which is generally convex and therefore cannot be regarded as robust. Tree-type and Deep Learning approaches usually suffer from the lack of interpretability. Similarly, many of the approaches do not invoke a suitable sparse model selection.

As for future research, a unified approach which does not depend on whether the response variable and which provides a sparse, robust, stable and well-interpretable model would be a desirable goal. Deep Learning has gained a lot of attention during the last decade as is capable to result in excellent predictions, but sparsity and interpretability of the model is still an open-research question.

Acknowledgements

Most of this paper is part of T. Werner’s PhD thesis at Oldenburg University under the supervision of P. Ruckdeschel.

References

S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth. Generalization bounds for the area under the ROC curve. Journal of Machine Learning Research, 6(Apr):393–425, 2005.

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. Cowper-smith, 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 S. Van De Geer. Statistics for high-dimensional data: Methods, theory and applications. Springer Science & Business Media, 2011.

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.

T. Calders and S. Jaroszewicz. Efficient AUC optimization for classification. In PKDD, volume 4702, pages 42–53. Springer, 2007.

S. Canu, Y. Grandvalet, V. Guigue, and A. Rakotomamonjy. Svm and kernel methods matlab toolbox. Perception Systemes et Information, INSA de Rouen, Rouen, France, 2005.

Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking SVM to document retrieval. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 186–193. ACM, 2006.

O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with SVMs. Information Retrieval, 13(3):201–215, 2010.

K. Chen, R. Li, Y. Dou, Z. Liang, and Q. Lv. Ranking support vector machine with kernel approximation. Computational intelligence and neuroscience, 2017, 2017.

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. Anomaly ranking as supervised bipartite ranking. In International Conference on Machine Learning, pages 343–351, 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 A. Thomas. Mass volume curves and anomaly ranking. arXiv preprint arXiv:1705.01305, 2017.

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. On partitioning rules for bipartite ranking. In Artificial Intelligence and Statistics, pages 97–104, 2009.

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.

S. Clémençon, N. Baskiotis, and N. Vayatis. Anomaly ranking in a high dimensional space: The unsupervised treerank algorithm. In Unsupervised Learning Algorithms, pages 33–54. Springer, 2016.

C. Cortes and M. Mohri. AUC optimization vs. error rate minimization. In Advances in neural information processing systems, pages 313–320, 2004.

V. Dang. Ranklib-a library of ranking algorithms, 2013.
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.

M. Engilberge, L. Chevallier, P. Pérez, and M. Cord. Sodeep: a sorting deep net to learn ranking loss surrogates. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10792–10801, 2019.

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.

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

N. Goix, A. Sabourin, and S. Clémençon. On anomaly ranking and excess-mass curves. In *Artificial Intelligence and Statistics*, pages 287–295, 2015.

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.

T. Hazan, J. Keshet, and D. A. McAllester. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems*, pages 1594–1602, 2010.

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.

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. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA, 1999.

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.

T. Joachims. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

C. Jung, L. Jiao, and Y. Shen. Ensemble ranking SVM for learning to rank. In *Machine Learning for Signal Processing (MLSP), 2011 IEEE International Workshop on*, pages 1–6. IEEE, 2011.

J. G. Kemeny. Mathematics without numbers. *Daedalus*, 88(4):577–591, 1959.

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.

A. Korba, S. Clémençon, and E. Sibony. A learning theory of ranking aggregation. In *Artificial Intelligence and Statistics*, pages 1001–1010, 2017.

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.

N. Meinshausen and P. Bühlmann. Stability selection. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 72(4):417–473, 2010.

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. Boerg, 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.

T. Pahikkala, A. Airola, P. Naula, and T. Salakoski. Greedy rankrls: a linear time algorithm for learning sparse ranking models. In Sigir 2010 workshop on feature generation and selection for information retrieval, pages 11–18. ACM, 2010.

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.

Y. Song, A. Schwing, R. Urtasun, et al. Training deep neural networks via direct loss minimization. In International Conference on Machine Learning, pages 2169–2177, 2016.

Y. Tian, Y. Shi, X. Chen, and W. Chen. AUC maximizing support vector machines with feature selection. Procedia Computer Science, 4:1691–1698, 2011.

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

T. Werner and P. Ruckdeschel. The column measure and gradient-free gradient boosting. Submitted. Available on arXiv, arXiv: 1909.10960, 2019.

Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In Advances in neural information processing systems, pages 1697–1704, 2008.