A review on instance ranking problems in statistical learning

Tino Werner

Received: 18 December 2020 / Revised: 25 October 2021 / Accepted: 28 October 2021 / Published online: 18 November 2021
© The Author(s) 2021

Abstract
Ranking problems, also known as preference learning problems, define a widely spread class of statistical learning problems with many applications, including fraud detection, document ranking, medicine, chemistry, credit risk screening, image ranking or media memorability. While there already exist reviews concentrating on specific types of ranking problems like label and object ranking problems, there does not yet seem to exist an overview concentrating on instance ranking problems that both includes developments in distinguishing between different types of instance ranking problems as well as careful discussions about their differences and the applicability of the existing ranking algorithms to them. In instance ranking, one explicitly takes the responses into account with the goal to infer a scoring function which directly maps feature vectors to real-valued ranking scores, in contrast to object ranking problems where the ranks are given as preference information with the goal to learn a permutation. In this article, we systematically review different types of instance ranking problems 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 systematize existing techniques and recapitulate the corresponding optimization problems in a unified notation. We also discuss to which of the instance ranking problems the respective algorithms are tailored and identify their strengths and limitations. Computational aspects and open research problems are also considered.

Keywords Instance 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 has been done by the PageRank algorithm (Page et al. 1999) for Google until 2013 when it became part of the Hummingbird algorithm. However, PageRank is essentially an unsupervised ranking algorithm 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 web-sites.

In this work, we focus on instance ranking problems, which belong to the family of supervised ranking problems. In supervised ranking problems, the training data consist of a single set or multiple sets of instances or instance-query pairs where each instance is composed by a feature vector and a certain type of preference information. One can divide these problems further into label, instance and object ranking problems depending on the preference information (Fürnkranz and Hüllermeier 2011). If the instances can belong to finitely many classes and for each instance the preference information is a ranking of the classes or of the class-specific probabilities, a label ranking problem is given. The goal is to predict the ranking of the class labels for new feature vectors. An object ranking problem is given if the preference information provides a pair-wise ranking, more precisely, if for each pair of feature vectors, the information which of them is ranked higher is provided. This can be interpreted as if the rank of each feature vector is provided as response. The goal is to learn a model that predicts an ordering of a set of new feature vectors. Related to object ranking problems but inherently different are instance ranking problems where the preference information is a discrete- or continuous-valued response assigned to each feature vector. The goal is to learn a scoring function that assigns new feature vectors to a real-valued ranking score. Having learned such a scoring function, new feature vectors receive a ranking score which directly leads to a ranking of them, inherited from the natural ordering on the real line. Note that these ranking scores do not have to match the responses as in regression but that their ordering should be as close as possible to the ordering of the original responses. In this review, we include a few ranking algorithms primarily designed for object ranking problems but which can directly be applied to instance ranking problems. We however exclude all algorithms that operate on ranks (which may be computed in a pre-processing step from actual real-valued responses), which is done in many information retrieval works.

In their seminal paper (Clémençon et al. 2008), Clémençon and co-authors proposed a statistical framework for instance ranking problems, which emerge from ordinal regression (Herbrich et al. 1999a, b) and proved that the common approach of empirical risk minimization (ERM) is indeed suitable for such ranking problems. Although there already existed instance 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 which refers to the situation that the responses either take w.l.o.g. the values 1 or -1 where 1 refers to the class of interest, therefore, a natural criterion for such binary or bipartite instance 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 instance ranking problems are very closely related to binary classification tasks (see also Balcan et al. 2008). For binary instance ranking problems, there exists vast literature,
including theoretical work as well as learning algorithms that use SVMs (Brefeld and Scheffer 2005; Herbrich et al. 1999a; Joachims 2002), Boosting (Freund et al. 2003; Rudin 2009), neural networks (Burges et al. 2005) or trees (Clémençon and Vayatis 2008, 2010).

As for the document ranking, the labels are also discrete, but with \( d > 2 \) classes, for example in the OHSUMED data set (Hersh et al. 1994). For such general \( d \)-partite instance ranking problems, there also has been developed theoretical work (Clémençon et al. 2013c), binary classification approaches (e.g. Fürnkranz et al. 2009) as well as tree-based learning algorithms (Clémençon and Robbiano 2015a, b, see also Robbiano 2013).

Continuous instance ranking problems (e.g. Sculley 2010) invoke continuous response variables, with potential applications in natural sciences or quantitative finance (cf. Clémençon and Achab 2017). This continuous instance ranking problem can be located on the other flank of the spectrum of ranking problems that is closest to regression. The continuous instance 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), the familiar background (Dickerson and Popli 2016), or performances in sports like an ELO-score in chess (Langville and Meyer 2012). A continuous instance 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 reasonably 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.

There already exist surveys for label ranking (Vembu and Gärtner 2010) and object ranking (Kamishima et al. 2010), while a review of object and instance ranking problems has also been given in Liu (2011), who concentrate on distinguishing between the actual learning paradigm in the sense of point-, pair- or list-wise learning, see also (Li 2011a) for a short overview of ranking algorithms and (Li 2011b) for a somewhat extended version, again distinguished by the learning paradigm. Guo et al. (2020) concentrated on ranking models based on neural networks. An overview paper where all the different types of instance ranking problems are discussed, including a concise separation between object and instance ranking and a discussion of the applicability of the existing ranking algorithms, has not been proposed so far. This work should close this gap.

This paper is organized as follows. Starting in Sect. 2 with the definition of several different ranking problems that are distinguished by the shape of the training data, the nature of the response variable and by the goal of the analyst, it becomes evident that suitable loss functions usually 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 a discrete response variable. In Sect. 3, we investigate different paradigms for instance ranking problems like identifying them with a family of binary classification problems and point-, pair- and list-wise loss approaches. We discuss further similarities and differences between label, object and instance ranking and point out that, depending on the application, even the type of
the appropriate ranking problem that has to be solved may be far from trivial to define. In Sect. 4, we provide a systematic overview of different machine learning algorithms by grouping them into SVM-, Boosting-, tree- and Neural Network-type approaches. We also describe the plug-in approach and further approaches that cannot be identified with one of these techniques. We review these approaches and discuss their strengths, limitations and computational aspects. Section 5 is devoted to a careful discussion of the combined ranking problems and a distinction between ranking and ordinal regression. We conclude with open research problems for instance ranking.

2 Instance ranking problems

2.1 Different types of ranking problems

In order to systematically categorize ranking problems, one has to answer three questions in the following order: What kind of data set do we have (feature-response-pairs, feature-permutation-pairs, a query structure, only features)? What type of response variable, if it exists, do we have (categorical, continuous)? What is the goal of the analyst?

At the top level, one can distinguish between label ranking, object ranking and instance ranking problems (Fürnkranz and Hüllermeier 2011). In label ranking problems (see e.g. Har-Peled et al. 2002; Cheng et al. 2012; Fürnkranz et al. 2008; Hüllermeier and Fürnkranz 2010; Lee and Lin 2014), the training data consists of features $X_i \in X$ for some measurable space $X$ and corresponding permutations $\pi(\text{perm}(1 : d))$ where

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

denotes the symmetric group on the label set $\{1, \ldots, d\}$. A permutation $\pi(X_i)$ is interpreted in the sense that $\pi(X_i)$ represents the most preferred class for instance $X_i$. Once a new feature vector $X_{\text{new}}$ appears, a prediction of the ordering of the class relevance is made, either by predicting class probabilities or directly predicting a permutation of $\{1, \ldots, d\}$.

Instance ranking considers data consisting of instances $(X_i, Y_i)$ with $Y_i \in \mathcal{Y}$ for some measurable, ordered space $\mathcal{Y}$ where one is interested in finding an ordering of the $X_i$ according to the natural ordering of the $Y_i$ in the respective response space. These responses are exploited in instance ranking by learning a scoring function $s : \mathcal{X} \rightarrow \mathbb{R}$ so that for a new feature vectors $X_{\text{new}}^i$, one predicts ranking scores $s(X_{\text{new}}^i)$ which induce an ordering of these new instances. Object ranking (see e.g. Cohen et al. 1999; Szörényi et al. 2015) can be regarded as a counterpart of instance ranking where no instance-specific preference information is given but only pair-wise preference information, more precisely, for given features $X_i, X_j$, one has the information that $X_i$ is better than $X_j$, denoted by $X_i \succ X_j$, or vice versa, but there is no individual ranking score for any $X_i$. The pair-wise preferences are often translated into individual ranks, so $X_i$ is associated with its rank in a given set of instances. The goal is to predict a permutation of the set $\{1, \ldots, n_{\text{new}}\}$ once a batch $X_{\text{new}}^1, \ldots, X_{\text{new}}^{n_{\text{new}}}$ of test instances appears but again, in contrast to instance ranking, a prediction for an individual test instance cannot be made.

In this review, we only consider instance ranking problems and we assume having data $D = (X, Y)$ where $Y_i \in \mathcal{Y} \subset \mathbb{R}$ and $X_i \in \mathcal{X}$ where $X_i$ denotes the $i$-th row of the regressor matrix $X$. As for the space $\mathcal{X}$, it is usually a subset of $\mathbb{R}^p$ but can be different for structured data like query-document lists (Cao et al. 2007), texts (Severyn and Moschitti 2015), reactant-condition tuples (Kayala et al. 2011) or images as in the context of image quality.
assessment (Zhai and Min 2020). Usually, the original structure is represented by a feature vector so that one again is in the setting where \( X_i \in \mathbb{R}^p \), but in principle, \( \mathcal{X} \) is not restricted to be a subset of \( \mathbb{R}^p \).

Solutions of instance 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 \( K < n \) instances have to be ranked exactly while the predicted ranking of the other instances is not a quantity of interest. We go into detail in the next subsection.

### 2.2 Different types of instance ranking problems

The goal in this review is to rank the \( X_i \) by comparing their predicted response, i.e., \( X_i \) is ranked higher than \( X_j \) if \( Y_i > Y_j \). We recapitulate the following definitions from (Clémençon et al. 2008).

**Definition 1**

(a) A **ranking rule** is a mapping \( r : \mathcal{X} \times \mathcal{X} \rightarrow \{-1, 1\} \) where \( r(X_i, X_j) = 1 \) indicates that \( X_i \) is ranked higher than \( X_j \) and vice versa.

(b) A **ranking rule induced by a scoring rule** \( s \) is given by \( r(X_i, X_j, s) = 2I(s(X_i) \geq s(X_j)) - 1 \) with a scoring function \( s : \mathcal{X} \rightarrow \mathbb{R} \) where \( r(X_i, X_j) = 1 \) if and only if \( s(X_i) \geq s(X_j) \). \( I \) is the indicator function which takes the value 1 if the logical statement in the brackets is satisfied and 0 otherwise.

Note that scoring rules are also used in label ranking and object ranking. In label ranking, one prefers class \( c_k \) over class \( c_l \) for feature \( X_i \) if \( s_k(X_i) \geq s_l(X_i) \) for scoring functions \( s_k, s_l : \mathcal{X} \rightarrow \mathbb{R} \) which in Hüllermeier et al. (2008) are called utility functions. Cohen et al. (1999) refer to such functions as ordering functions. Furthermore, the ranking rule defined in Clémençon et al. (2008) is a hard ranking rule, making a clear decision whether \( X_i \) has to be preferred over \( X_j \). Cohen et al. (1999) work with the concept of a probabilistic preference function where a value in \([0, 1]\) is assigned to a pair \((X_i, X_j)\) where a value close to 1 indicates that \( X_i \) is preferred over \( X_j \) and vice versa.

In this work, we will refer to the problem to correctly rank all instances as the **hard instance ranking problem**, which is a global problem. A weaker problem is the **localized instance ranking problem** that intends to find the correct ordering of the best \( K < n \) instances, so misrankings at the bottom of the list are not taken into account. However, misclassifications in the sense that instances that belong to the top \( K \) ones are predicted as belonging to the bottom of the list or vice versa 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 instance ranking problem** where one only requires to reliably detect the best \( K \) instances but where their pair-wise ordering is not a quantity of interest. 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 \( K < n \) 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 \( \hat{\text{Best}}_K \). Worked out theory for the weak and localized instance ranking problem is given in Clémençon and Vayatis (2007).
On the other hand, one distinguishes between three other types of instance ranking problems in dependence of the set \( \mathcal{Y} \). If \( \mathcal{Y} \) 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 \textbf{bipartite (binary) instance ranking problem}, see (Fürnkranz and Hüllermeier 2011). We have to be more precise here: The binarity refers to the response set itself, i.e., all training instances have label 1 or label -1 which, for example, in document ranking problems represent the classes “relevant” and “not relevant”. If \( \mathcal{Y} \) can take \( d \) different values, a corresponding ranking problem is referred to as a \textbf{\( d \)-partite instance ranking problem}, also generally called multi-partite ranking problem (cf. Fürnkranz and Hüllermeier 2011). For continuously-valued responses, one faces a \textbf{continuous instance ranking problem}.

So far, we distinguished between different types of ranking problems on different levels. We will discuss in Sect. 5 what combinations define meaningful problems. Evidently, one may also distinguish between hard, weak and localized object and label ranking problems. Such an idea has already been proposed in Fürnkranz et al. (2008) for label ranking where one has to learn both a (hard) ranking but also a binary classification into relevant and non-relevant labels.

### 2.3 Loss functions for supervised ranking

#### 2.3.1 Hard ranking

Empirical risk minimization (ERM) is a standard learning technique where one defines a loss function \( L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R} \) which assigns a real-valued loss to a pair \((Y_i, \hat{Y}_i)\) for the prediction \( \hat{Y}_i \) of \( Y_i \). The risk is the expected loss, so if \( \hat{Y}_i = s(X_i) \), the risk is given by

\[
R(s) = \mathbb{E}_{X,Y} [L(Y, s(X))],
\]

which is generally intractable. The ERM principle empirically approximates this risk by

\[
\frac{1}{n} \sum_i L(Y_i, s(X_i)),
\]

so an empirical optimizer \( \hat{s} \) is computed by minimizing this empirical risk w.r.t. \( s \). Clemençon et al. (2005); Clémençon et al. (2008) provided the theoretical statistical framework for empirical risk minimization in the ranking setting. The hard ranking risk, i.e., the risk function of the hard instance ranking problem, used in Clemençon et al. (2005) and essentially going back to (Herbrich et al. 1999a), is given by

\[
R^{\text{hard}}(r) := \mathbb{E}[I((Y - Y')r(X, X') < 0)]. \tag{1}
\]

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

\[
\min_{r \in \mathbb{R}} (L_n^{\text{hard}}(r))
\]

where

\[
L_n^{\text{hard}}(r) = \frac{1}{n(n-1)} \sum_{i \neq j} I((Y_i - Y_j)r(X_i, X_j) < 0) \tag{2}
\]
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 function are suppressed. For discrete-valued responses, some summands will be zero if \( Y_i = Y_j \). A more natural standardization of this loss would therefore be the number of pairs for which \( Y_i \neq Y_j \) instead of \( n(n-1) \). Note that \( L_n^{\text{hard}} \), i.e., the hard empirical risk, is also the hard ranking loss and not a sum of individual instance-wise losses as in regression or classification settings, which reflects the global nature of hard ranking problems. One can artificially identify the indicator function in Eq. 1 as a common loss function which, however, is operating on pairs.

In the instance ranking setting, ranking rules induced by scoring rules are self-evident due to the natural ordering existing on \( \mathcal{Y} \). In the special case of a parametric scoring function, one considers some parameter space \( \Theta \subset \mathbb{R}^p \) which means that it suffices to empirically find the best parametric scoring function (and with it, the empirically optimal induced ranking rule) from the family

\[
\mathcal{S} := \{ s_\theta : \mathcal{X} \to \mathbb{R} \mid \theta \in \Theta \}
\]

of such scoring functions by solving the parametric optimization problem

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

with

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

(3)

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

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

(4)

In fact, this loss function can be identified with the ranking loss used in Fahandar and Hüllermeier (2017) in the context of object ranking where the training data consists of sets of features including the corresponding true permutations representing the ordering on the respective subset.

Ai et al. (2019) argue that univariate scoring functions may not reflect possible dependencies of individual scores to other instances and propose group-wise scoring functions that operate on instance groups of fixed size to which a score vector of the same size is assigned. Since the number of potential groups grows factorially, they suggest Monte Carlo sampling in order to approximate the aggregated individual scores.

### 2.3.2 Weak ranking

For the weak instance 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, we consider the transformed responses
\[ \bar{Y}_i^{(K)} := 2I(\text{rk}(Y_i) \leq K) - 1 \]
where the ranks come from a descending ordering, i.e.,
\[ \text{rk}(Y_i) = \sum_j I(Y_i \leq Y_j). \]

Then the misclassification risk corresponding to the weak instance ranking problem in the sense of Clémençon and Vayatis (2007) is given by
\[ R_{\text{weak},u}(s) := P(\bar{Y}(s) \leq Q(s, 1 - u)) < 0 \]
with the empirical counterpart
\[ L_{\text{weak},K,n}(s) = \frac{1}{n} \sum_{i=1}^{n} I(\bar{Y}_i^{(K)}(s(X_i)) < 0) \]
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 level \((1 - u)\), one looks at the top \( K \) instances that represent this upper quantile.

**Remark 1** Due to the mass constraint, each false positive generates exactly one false negative, so the loss can be equivalently written as
\[ L_{\text{weak},K}(s) = \frac{2}{n} \sum_{i \in \text{Best}_K} I(\bar{Y}_i^{(K)}(s(X_i)) < 0). \]

Note that the weak ranking loss is not standardized, i.e., it is not necessarily able to take the value 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 “top half” and predicted to be in the “bottom half” and vice versa. For better comparison of the losses, Werner (2019) propose the standardized weak ranking loss
\[ L_{\text{weak},K,norm}(s) = \frac{1}{K} \sum_{i \in \text{Best}_K} I(\bar{Y}_i^{(K)}(s(X_i)) < 0). \]

**Remark 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 \( \text{Best}_K \) did not have been recovered by the model.

### 2.3.3 Localized ranking

A suitable loss function for the localized instance ranking problem was proposed in Clémençon and Vayatis (2007). In our notation, it is given by
\( L_{n,\text{loc},K}^n(s) := \frac{n}{n} L_{n,\text{weak},K}^n(s) + \frac{1}{n(n-1)} \sum_{i \neq j} I((s(X_i) - s(X_j))(Y_i - Y_j) < 0) \cap \{ \min(s(X_i), s(X_j)) \geq \hat{Q}(s, 1 - u^{(K)}) \} \) \tag{6}

In the first summand, \( n_\) indicates the number of negatives, so the quotient is just an estimate 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

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

In principle, the sum may also run over all indices in \( \text{Best}_K \) which indicates that not the ordering of the predicted top \( K \) instances but the ordering of the true top \( K \) instances has to be correct. Werner (2021) studied the global robustness of localized ranking problems, among others, for both versions and revealed that, depending on \( K \), there can indeed be a little difference in the robustness of the corresponding localized instance ranking problems. 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

\[
\max(L_{n,\text{loc},K}^n(s)) = \frac{n-K}{n} \cdot \frac{2K}{n} + \frac{K(K-1)}{n(n-1)} =: m_K,
\]

so a standardized version is simply

\[
L_{n,\text{loc},K,\text{norm}}^n(s) := \frac{1}{m_K} L_{n,\text{loc},K}^n(s).
\]

**Remark 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

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

A simple example for clarification is given below in Example 1 which we borrow from (Werner 2019).

**Example 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

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

$$L_{n}^{\text{weak},A}(\pi, \hat{\pi}) = \frac{2}{10} = 0.2.$$ 

The standardized weak ranking loss is then

$$L_{n}^{\text{weak},A,\text{norm}}(\pi, \hat{\pi}) = \frac{10}{8} \cdot \frac{2}{10} = 0.25,$$

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

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

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

$$L_{n}^{\text{loc},A}(\pi, \hat{\pi}) = \frac{2}{45} + \frac{6}{10} \cdot 0.2 = \frac{37}{225}.$$ 

The standardized localized ranking loss is then

$$L_{n}^{\text{loc},A,\text{norm}}(\pi, \hat{\pi}) = \frac{75}{46} \cdot \frac{37}{225} \approx 0.268.$$ 

Finally, the hard ranking loss is

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

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

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

The standardized localized ranking loss is

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

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.

Summarizing, we gave an overview of the natural ranking losses for hard, weak and localized instance ranking problems. However, these loss functions are non-continuous, so optimizing them directly would be very difficult, which is a well-known issue from classification where a natural classification loss function is the 0/1-loss function which is also an indicator function. A common technique is to optimize a sufficiently regular
surrogate loss. This principle has already entered instance ranking problems and the algorithms that we review in Sect. 4 operate on particular surrogate losses.

2.4 Quality criteria for ranking

So far, we presented loss functions for instance ranking problems that lead to algorithms in the spirit of the ERM 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** 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.

(a) The **true positive rate** (TPR) and the **false positive rate** (FPR) are given by

\[
\text{TPR} = \frac{1}{n_+} \sum_i I(\hat{Y}_i = 1)I(Y_i = 1), \quad \text{FPR} = \frac{1}{n_-} \sum_i I(\hat{Y}_i = 1)I(Y_i = -1).
\]

(b) The **Receiver Operation Characteristic curve** (ROC curve) is the plot of the true positive rate against the false positive rate.

(c) The AUC is defined as the **area under the ROC curve**.

For theoretical aspects of the empirical AUC and its optimization, we refer to Agarwal et al. (2005); Cortes and Mohri (2004) and Calders and Jaroszewicz (2007). 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 computing the TPR and the FPR. Since the goal is to achieve a TPR as high as possible for the price of an 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.

The connection of the AUC and bipartite ranking has been elaborated in (Cortes and Mohri 2004, Lem. 1) where they proved that the AUC for a binary classifier can be written as

\[
\text{AUC} = \frac{1}{n_- n_+} \sum_{i : Y_i = 1} \sum_{j : Y_j = -1} I(\hat{Y}_i > \hat{Y}_j)
\]

for \( \hat{Y}_i \in \{-1, 1\} \) for all \( i = 1, \ldots, n \). This is the Mann-Whitney U-statistic which is equivalent to \( 1 - L_n^{\text{hard}} \) except for the standardization, therefore, maximizing the AUC is equivalent to minimizing the hard ranking loss.

For bipartite localized instance ranking problems, Clémençon and Vayatis (2007) 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 2007).
Definition 3  The localized AUC is defined as

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

As for \( d \)-partite ranking problems, i.e., \( Y \) can take \( d \) different values, w.l.o.g. \( Y = \{1, \ldots, d\} \) with ordinal classes, Clémençon et al. (2013c) proposed the VUS (volume under the ROC surface) as quality criterion.

Definition 4  Let w.l.o.g. \( Y \) take values in \( \{1, \ldots, d\} \) and let again \( X \) take values in \( X \subset \mathbb{R}^p \).

(a) The ROC surface is the "continuous extension" (Clémençon et al. 2013c) of the plot for \( t_1 < t_2 < \cdots < t_{d-1} \).

(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 \( (k+1) \).

The VUS is not the only possible way how to assess the quality of multi-partite ranking models. Fürnkranz et al. (2009) considered the C-index

\[ C(s, X) = \frac{1}{\prod_i n_i} \sum_{i<j} \sum_{(X_k, X_l) \in C_i \times C_j} I(s(X_l) > s(X_k)) \]

where \( C_i \) denotes the set of all class-\( i \)-instances with \( n_i = |C_i| \), measuring the probability that a randomly selected class-\( j \)-instance is (correctly) ranked above a randomly chosen class-\( i \)-instance, which is equivalent to the hard ranking loss in Eq. 2 if no ties are apparent but which is standardized to \([0, 1]\) even if ties are observed while the hard ranking loss cannot attain the value 1 in this case. Note that the Mann-Whitney U-statistic in Eq. 7 is a special case of the C-index for bipartite ranking. The extension

\[ U(s, X) = \frac{2}{d(d-1)} \sum_{i<j} \text{AUC}(s, C_i \cup C_j), \]

of the AUC, which in Fürnkranz et al. (2009) is identified with a weighted version of the C-index. Waegeman et al. (2008) proposed the metric

\[ W(f(X)) = \frac{1}{\prod_i n_i} \sum_{X_1 \in C_1, \ldots, X_d \in C_d} I(s(X_1) < \cdots < s(X_d)) \]

which, however, as discussed in Fürnkranz et al. (2009), neglects how severely the ordering is violated, i.e., if there is only one misranking between two of the \( d \) instances or if the ordering is even reverted. Fürnkranz et al. (2009) therefore concentrate on \( C(s, X) \) and \( U(s, X) \).

Other well-known quality criteria for ranking problems which put more weight on the top instances are for example the MAP (mean average precision) and the NDCG.
(normalized discounted cumulative gain), which can even enter object ranking (see Cheng et al. 2010). An overview of different ranking metrics and corresponding loss functions is provided by Wang et al. (2018). Note that there are already even deep models that optimize such metrics [called deep metric learning, see e.g. Cakir et al. (2019)].

2.5 Evaluation of ranking losses

We want to emphasize that a common problem when considering ranking losses or quality criteria for ranking are the computational costs since one usually has to perform some kind of ordering. Moreover, the ranking losses defined in Sect. 2.3 would require \( \mathcal{O}(n^2) \) comparisons when evaluating them thoughtlessly.

To this end, there have been proposed strategies that reduce this complexity to \( \mathcal{O}(n \ln(n)) \) by considering quick-sort algorithms as done in Ailon and Mohri (2007) (note that they do not evaluate a ranking loss but generate a predicted ordering). In general, they rely on order statistics trees which can be computed in logarithmic time (Cormen et al. (2009)). More precisely, the height of such a tree is \( \mathcal{O}(\ln(n)) \) which allows for evaluating the rank of a given element resp. identifying the element with a given rank in logarithmic time as elaborated in Ailon and Mohri (2007). This is used in (Ailon and Mohri 2007, Alg. 2) to compute the number of misorderings in a predicted ordering. Note that for binary responses and scores, one only has to sort the data w.r.t. the original responses and, for each positive, sum up the number of negatives that have been ranked higher than it. One can also relate the hard ranking loss with Kendall’s \( \tau \) as done in Werner (2019, Lem. 6.1.1) for continuous responses without ties so that a fast computation method for Kendall’s \( \tau \) essentially going back to Knight (Knight 1966), which is implemented for example as \texttt{cor}, \( \text{fk} \) in the \texttt{pcaPP} (Filzmoser et al. 2018), becomes applicable, reducing the complexity also to \( \mathcal{O}(n \ln(n)) \). Even in the presence of ties and for general real-valued scores, one can compute the C-index with a complexity of again \( \mathcal{O}(n \ln(n)) \) where again order statistics trees are invoked.\(^1\)

A sampling strategy has been proposed in Sculley (2009) who suggests sampling pairs in order to statistically approximate the true loss. They describe how to execute this strategy even for more complex settings like structured data or streaming data but the strategy is of course applicable to the standard setting. Although one may drastically reduce the evaluation complexity, it is questionable whether one should take the approximation error into account since one can doubt that significantly reducing the complexity of \( \mathcal{O}(n \ln(n)) \) would be possible with that strategy.

3 Paradigms of ranking algorithms

In the previous section, we recapitulated natural pair-wise loss functions for different types of instance ranking problems. Minimizing such a loss function (or better a surrogate of it) is not the only way how (instance) ranking problems can be tackled. In this section, we review and discuss other approaches like reducing the ranking problems to a family of pair-wise classification problems or making a probabilistic ranking prediction. Moreover, even a pair-wise loss is not always necessary since there exist approaches that optimize a common

\(^1\) http://staff.cs.utu.fi/~aatapa/software/RLScore/modules/cindex.html.
point-wise loss function or that even require a list-wise loss function for structured data. We also discuss the connection of object resp. label ranking problems to instance ranking problems based on these aspects and finally emphasize that even for a given problem and a given data set, choosing which type of (instance) ranking problem has to be formulated may not be trivial.

3.1 Ranking and pair-wise classification

The regret of a scoring function \( s \) is defined as

\[
\mathbb{E}_{X,Y}[L(s(X), Y)] - \mathbb{E}_{X,Y}[L(s^*(X), Y)]
\]

where the expectation is taken over the joint distribution of \( X \) and \( Y \) and where \( s^* \) is the optimal model. Balcan et al. (2008) and Ailon and Mohri (2007) showed that bipartite instance ranking problems can be reduced to pair-wise classification problems so that instead of training a ranking model by optimizing a ranking loss, they suggest training a binary classifier on all (or a subset of all) pairs of positive and negative instances. Balcan et al. (2008, Thm. 1) prove that the ranking regret, induced by the AUC loss (which is 1 minus the AUC), is bounded by two times the classification regret, induced by the 0/1-classification loss. Ailon and Mohri (2007, Thm. 3) refine their result and prove that the hard ranking risk is bounded by the misclassification risk itself. They transfer this result to multi-partite instance ranking where they define a generalized AUC loss, i.e.,

\[
\sum_i \sum_{j 
eq i} I(Y_i > Y_j) \max(0, \hat{Y}_i - \hat{Y}_j) / \sum_i \sum_{i < j} |\hat{Y}_i - \hat{Y}_j|.
\]

The result from Balcan et al. (2008, Thm. 1) translates to this case, i.e., the ranking regret induced by this loss function is bounded by two times the classification regret.

Fürnkranz et al. (2009) consider an idea of Frank and Hall (2001) for reducing ordinal classification to a family of binary classification problems, more precisely, to \((d - 1)\) classification problems if \( d \) is the number of different labels. This amounts to aggregate the predictions during inference. Fürnkranz et al. (2009) suggest aggregating the scores rather than the predicted orderings themselves. Let for a class \( k \) the meta-classes \( C_k^- = \{1, \ldots, k\} \) and \( C_k^+ = \{k + 1, \ldots, d\} \) be given. Then, for each \( k \), a binary classifier \( c^{(k)} \) is trained on the corresponding data set with the meta-classes as responses. They refer to this strategy as Frank-Hall approach (Frank and Hall 2001). As an alternative, they consider round robin learning (Fürnkranz 2002) where pair-wise comparisons are made, i.e., a classifier \( c^{(k,l)} \) is trained on a pair of samples where on instance belongs to class \( k \) and the other one to class \( l \). Fürnkranz et al. (2009) point out that the pair-wise approach is problematic since each model \( c^{(k,l)} \) is only valid for such pairs but they also argue the error of the individual models may be compensated by aggregation. The complexity of the Frank-Hall strategy is \( \mathcal{O}(dn^2) \) and that of the pair-wise strategy is \( \mathcal{O}(d^2n^4) \) for a base learner with complexity \( \mathcal{O}(n^4) \).

Rajaram et al. (2003) suggest solving multi-partite instance ranking problems using ordinal regression, i.e., one predicts a real-valued pseudo-response for each instance which are discretized according to \((d + 1)\) thresholds that define \( d \) intervals. If the pseudo-response falls into the \( k \)-th interval, class \( k \) is predicted. The thresholds are usually learned during training. They propose the difference approach where for \( Y_i \neq Y_j \) the feature vector \( X_{ij} = X_i - X_j \) and the response \( Y_{ij} = \text{sign}(Y_i - Y_j) \) are computed on which a binary classifier is trained so that aggregating these classifiers leads to a ranking model. The embedding
approach in contrast identifies the ranking problem with an \((n + d - 2)\)-dimensional binary classification problem. Both approaches can also invoke feature transformations by kernels. Such an approach is also given by Li and Lin (2007) where magnitude-sensitive losses are considered, more precisely, they focus on \(V\)-shaped losses, i.e., for true class label \(Y\) and predicted class label \(\hat{Y}\), one has \(L(Y, \hat{Y} - 1) \geq L(Y, \hat{Y})\) for \(\hat{Y} \leq Y\) and \(L(Y, \hat{Y}) \leq L(Y, \hat{Y} + 1)\) for \(\hat{Y} \geq Y\), which, for example, is true for \(L(Y, \hat{Y}) = |Y - \hat{Y}|\). They argue that the Frank-Hall approach (Frank and Hall 2001) may lead to a poor generalization performance and propose a method to solve all binary classification problems jointly by essentially binarizing the responses and considering instance-individual weights. Li and Lin (2007, Thm. 1) show that the weighted 0/1-classification loss is an upper bound for the multi-class classification loss. See also Li and Lin (2007, Thm. 3) and Li and Lin (2007, Thm. 4) for bounds of the generalization error. Kotlowski et al. (2011) show that the minimization of a pairwise 0/1-classification loss is no appropriate strategy for ranking by providing an example where the Bayes classifier (having a regret of zero) does not induce a Bayes ranker, causing a positive ranking regret, so upper-bounding the ranking regret by the classification regret is impossible. They therefore reduce themselves to the losses instead of the regrets and show in Kotlowski et al. (2011, Thm. 3.1) that the ranking loss can be bounded from above by a weighed 0/1-classification loss, which, however, can be rather loose. Considering margin-based loss of the form \(L(Y, \hat{Y}) = L(\hat{Y}^I)\), Kotlowski et al. (2011, Thm. 4.1) prove ranking regret bounds in terms of the exponential and the logistic classification regret. Dembczynski et al. (2012) extend the results of Kotlowski et al. (2011) for multi-partite ranking. Dembczynski et al. (2012, Thm. 3.1) prove regret bounds for bipartite ranking in terms of the exponential and the logistic classification regret and generalize them in Dembczynski et al. (2012, Thm. 3.2) to \(d\)-partite ranking, also extending the work of Gao and Zhou (2011).

Agarwal (2014) generalize the results of Kotlowski et al. (2011) for a broader class of loss functions, more precisely, for proper loss functions where the term “proper” is meant in the spirit of Gneiting and Raftery (2007). They concentrate on the subclass of proper composite losses, for which \(L(Y, \hat{Y}) = \tilde{L}(\hat{Y}, \psi^{-1}(\hat{Y}))\) for a proper loss function \(\tilde{L}\) and a link function \(\psi : [0, 1] \rightarrow \mathcal{Y}\). They identify the exponential, the logistic, the quadratic and the squared Hinge loss as such loss functions and also show how to construct proper composite loss functions.

Summarizing, there exist theoretical results that prove that casting an instance ranking problem but also an object ranking problem as for example done in Fahandar and Hüllermeier (2017) as a family of binary classification problems is a valid approach. It can be applied to any type of ranking problem since the ties that arise in bipartite and multi-partite ranking problems are ignored while continuous ranking problems usually do not lead to ties (which would be ignored when applying this approach). Due to the reduction of the response information to preference information, it could happen that valuable information, which is provided especially by continuous responses, gets lost. Moreover, during testing, some approaches like the one in Balcan et al. (2008) rather predict rankings instead of ranking scores which makes it questionable at least for continuous instance ranking. On the other hand, reducing a ranking problem to a family of binary classification problems is a formidable strategy for object ranking and label ranking (see also Fürnkranz and Hüllermeier 2010) where solely preference information are given in the training data.

We want to emphasize that in principle, any binary classification algorithm can be applied here which makes this approach versatile and applicable to many cases. For example, a high-dimensional ranking problem with \(p > n\) may be solved as a sequence of binary classification problems where a classification algorithm providing a sparse model by
structural risk minimization (SRM, Vapnik 1998) which amounts to optimizing as regularization empirical risk can be applied. This however also highlights a limitation of some approaches that suggest learning different classifiers for the individual classification problems since these models have to be aggregated. Even though the aggregation may enter by aggregating the predicted scores instead of the models themselves, it would be difficult to interpret those ensembles of classifiers which can be based on different sets of selected variables.

Reviewing classification algorithms is not part of this paper, so we concentrate on ranking algorithms that fit a scoring function to each individual instance in Sect. 4.

### 3.2 Probabilistic models

Many ranking algorithms learn a scoring function resp. perform binary classification for instance pairs. These scores resp. the binary classifications induce a permutation. However, such a permutation is one single element of the symmetric group $\text{Perm}(1:n)$, in other words, a Dirac distribution on $\text{Perm}(1:n)$. Casting a ranking problem as a pair-wise classification problem could be interpreted as a soft prediction if every single classifier makes a logit or another type of probabilistic prediction which one could aggregate to get a probabilistic overall ranking prediction. There already exist very popular approaches which directly learn a distribution on the symmetric group $\text{Perm}(1:n)$ of permutations.

The Mallows model (Mallows 1957) defines

$$P_{\pi, \theta}(\hat{\pi}) = \frac{1}{Z} e^{\sigma D(\pi, \hat{\pi})}$$

with a normalization factor $Z$, a deviation parameter $\sigma$ and a reference ranking $\pi$. $D$ is a distance between permutations, in general, Kendall’s $\tau$, but other distances like Spearman’s $\rho$ are also possible. The parameter $\sigma$ is inferred by a maximum likelihood estimation. The Plackett–Luce model (Luce 1959; Plackett 1975) in contrast is an iterative approach where one has

$$P_{\nu}(\hat{\pi}) = \prod_{i=1}^{n} \frac{\nu_{k^{-1}(i)}}{\sum_{j=1}^{n} \nu_{k^{-1}(j)}}$$

with weights $\nu_i$ which are inferred with a Bayes estimation. These models have successfully entered object ranking (Szörényi et al. 2015) and label ranking (Busa-Fekete et al. 2014; Cheng et al. 2009). See also (Qin et al. 2010) for an application in rank aggregation.

### 3.3 Point-, pair- and list-wise approaches

The ranking loss functions that we presented so far were pair-wise ones, i.e., they require a pair of true responses and the corresponding pair of predicted responses as input. There are two other paradigms of formulating loss functions for ranking.

Surprisingly, there are approaches to solve an instance ranking problem by minimizing a common point-wise loss function which only requires the true responses and the corresponding predicted responses as input like a standard quadratic loss function used for regression, see for example (Cossock and Zhang 2006).

The third paradigm originates from the setting of query-based document retrieval (cf. Joachims (2002)) where one has a set $Q$ of queries and a list of documents $d^{(q)}_1, \ldots, d^{(q)}_{\mu(q)}$
associated with a particular query $q \in Q$. Usually, a joined feature representation $X_j^{(q)}$ is computed for each query-instance pair $(q, d_i^{(q)})$ so that for each query $q$, a whole list $((X_1^{(q)}, y_1^{(q)}), \ldots, (X_n^{(q)}, y_n^{(q)}))$ is treated as an instance. Cao et al. (2007) introduce the list-wise approach for ranking where one aims at minimizing a list-wise loss function which directly operates on these lists. We will refer to this setting as **structured instance ranking**. They point out that even though (Joachims 2002) formulate the ranking problem in a pair-wise fashion for this setting, this approach is not applicable in general due to the i.i.d. assumption and since a classification loss is minimized instead of a real ranking loss function. Moreover, the ranking will be biased towards queries with many related documents. Qin et al. (2008b) argue that minimizing pair-wise loss functions does not necessarily imply that the ranking performance is improved and provide a simple example where the number of documents is highly imbalanced across the queries, so the ranking for a specific query with only a few related documents may be poor while the total ranking performance is high. This resembles the well-known issue from classification which may also be biased towards classes with a high frequency in the data set. Qin et al. (2008b) call the list-wise loss functions **query-level loss functions** and show that the usual pair-wise surrogate loss functions can hardly be extended to query-level. See also (Xia et al. 2008) for theoretical properties of selected list-wise loss functions.

Important insights about the relation of quality criteria like (N)DCG and (M)AP are given in Joachims (2002), Li et al. (2007), Cossock and Zhang (2006) and Chen et al. (2009). Li et al. (2007) and Cossock and Zhang (2006) show that the squared loss function for regression is an upper bound for the MAP resp. the DCG. Chen et al. (2009) prove that particular pair-wise and list-wise ranking loss functions are upper bounds for (1-MAP) and (1-NDCG), generalizing a result from (Joachims 2002) who provides an upper bound for (1-AP) in terms of a pair-wise ranking loss function. Summarizing, minimizing these ranking loss functions is equivalent to maximizing the MAP resp. the NDCG. Xu et al. (2008) also showed that maximizing a performance measure amounts on minimizing a pair-wise ranking loss. Bruch et al. (2019) show that minimizing the negative cross-entropy loss where the responses are transformed by the softmax (expit) function is equivalent to maximize the mean DCG and the mean reciprocal rank, both for binary responses.

### 3.4 Connection of object and instance ranking

We already distinguished between instance, object and label ranking problems and restricted ourselves to instance ranking problems. However, there is one interesting approach where the algorithms that we review in this work for instance ranking are applicable for object ranking.

Being inherently different from instance ranking problems, there indeed exists a paradigm which relates object ranking problems to instance ranking problems. Fahandar and Hüllermeier (2017) point out that when using a scoring function approach in object ranking, i.e., ranking $X_i$ before $X_j$ if $s(X_i) > s(X_j)$, and if the training data are of the form $(X_i^{(m)}, \pi^{(m)})_{m=1}^M$ for sets $X^{(m)}$ of objects and corresponding true orderings $\pi^{(m)} \in \text{Perm}(1 : |X^{(m)}|)$, one may follow a point-wise paradigm. This paradigm amounts to replacing each object with a set of labeled instances where the labels depend on the permutation value and the size of the actual set $X^{(m)}$ (see Kamishima et al. 2010 for this so-called expected rank regression approach), so one artificially defines responses for each instance that are used for a regression approach.
The main difference between instance and object ranking problems is that in instance ranking problems, a ranking score, i.e., a discrete- or continuously-valued response is given for each instance in contrast to object ranking which only provides pair-wise preferences. Evidently, the responses in instance ranking also provide such ranks so that object ranking techniques like probabilistic models would be applicable, but this approach has several limitations. In the case of bipartite or $d$-partite instance ranking, only the class labels are given, so translating the responses into ranks would produce many ties (see also Zhou et al. 2008) which, especially in binary instance ranking problems, could render this technique to apply object ranking algorithms to this problem nearly meaningless, see also Zhu and Klabjan 2020, who pointed out that the Plackett–Luce model produces imprecise permutation probabilities in the presence of ties. Hence, the approaches directly designed for bipartite and multi-partite instance ranking that maximize the AUC or variants of it are much more appropriate. Continuous instance ranking problems generally do not suffer from ties, but boiling down the response information to mere ranks would delete precious information (although it depends on the quality of the response values, in document retrieval with given absolute judgements, transforming the responses into preference information might be beneficial, see e.g. Zheng et al. (2007)). See again Zhu and Klabjan (2020) who criticize that the Plackett–Luce model cannot respect high relevance grades even for multi-partite ranking problems. For example, if the true underlying scoring function that maps the features onto the responses is highly non-linear, this fact would be completely ignored when just considering the ranks of the instances in contrast to approaches that learn such a scoring function.

In our opinion, the response type (observed and predicted) is indeed the strongest difference between object and instance ranking that clearly distinguishes between them. From the perspective of a particular ranking algorithm however, the difference is indeed opaque since, depending on whether ranking scores (instance ranking) or ranks (object ranking) are provided by the training data, the algorithm may be applied to both problems. In this review, we include algorithms that stem from object ranking but that can directly be translated to the instance ranking responses, but we abstain from the less meaningful way to reduce the responses in instance ranking to ranks in order to apply an object ranking algorithm, excluding those algorithms from this work.

### 3.5 Connection of structured instance ranking and label ranking

Label ranking refers to the case that instances which consist of a feature as well as of preference information for its discrete label are given. The goal is to learn a ranking function that maps each feature $X_i$ onto an ordering $\pi(X_i) \in \text{Perm}(1 : d)$ for the number $d$ of categories of the response variable. In contrast, in structured instance ranking where query-document pairs are given, the goal is to predict the relevance for each document w.r.t. the given query. Note the strong resemblance to label ranking since the document set is also a discrete set.

However, there are some inherent differences between instance ranking based on query-document pairs and label ranking. In label ranking, the number of categories for the response variable is equal for each instance. In contrast, each query can potentially be related to a different number of documents. Moreover, label ranking can, apart from ranking scores, output an ordering of the categories for each feature in terms of a permutation or in terms of a probability for each class. This is the main difference to structured instance ranking since for each query, there are usually multiple relevant documents which makes
a probabilistic output on the document set inappropriate, so one either predicts a discrete relevance label for each document w.r.t. the specific query or a real-valued relevance score.

3.6 Applications: Which ranking problem is to be addressed?

Due to the variety of different ranking problems, it is not always evident which of these problems fits to which application. As already mentioned, questions like if a hard or a localized ranking problem have to be solved have to be answered by the analyst and depend on the application domain. For example, a localized ranking problem seems to be more appropriate for document retrieval since the predicted order of the documents at the bottom of the list may not be relevant at all, but even the decision whether a binary or continuous instance ranking problem or whether an instance or an object ranking problem is faced may not be evident.

In some applications in the chemistry or medicine context, it can be rather simple to make this decision, for example if one aims at ordering instances according to the probability of breast cancer as done in Sapir (2011). One can also transpose the matrix $X$ as done in Agarwal and Sengupta (2009) where one does not want to rank patients but genes according to their relevance for some disease. More precisely, one has binary responses, i.e., the disease is observed or not, while the features $X_i$ contain the expressions $X_{ij}$ of gene $i$ over $p$ patients, but one is again clearly in the bipartite instance ranking setting. In Agarwal et al. (2010), chemical structures are ranked in the context of drug discovery according to the real-valued $pIC_{50}$-value that acts as response, leading to a continuous instance ranking problem. Other data sets multi-class data sets like the cardiocotography data set analyzed in Clémençon et al. (2013c) implies that a multi-partite instance ranking problem is faced.

On the other hand, multi-class data is sometimes boiled down to binary responses like for example done in Sapir (2011) where the multi-class heart disease data set is considered which contains not only the information whether a heart disease occurred but also the degree. Condensing the different positive classes to one positive class evidently leads to a loss of information which is generally not desirable. A similar argumentation has been made in Werner (2019) in the context of risk-based auditing (see e.g. Alm et al. 1993; Gupta and Nagadevara 2007; Hsu et al. 2015). One can formulate the problem as a binary ranking problem which is the common way 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 (see also Fürnkranz et al. 2009), 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. In contrast, if one defines 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.

In contrast to the tax auditing setting where the amount of damage can be observed, the design of the problem becomes far more difficult in applications where the continuous responses themselves are just implicit measurements. This is true for methods that invoke
pseudo-responses when considering measurements in sports (Langville and Meyer 2012), intelligence (Borsboom et al. 2003), personality (Anand et al. 2011), familiar background (Dickerson and Popli 2016), but also for measurements of image quality done by human annotators (e.g. Ma et al. (2016)). As for the responses in document retrieval, a natural relevance measure is the number of clicks (Joachims 2002; Cao et al. 2007). However, the number of clicks can be misleading as highlighted in Joachims (2002) and Joachims (2005) since the number of clicks is influenced by the quality of the other links and the order in which the documents are presented by the search engine. Therefore, although responses are available, one sometimes indeed translates the instance ranking problem to an object ranking problem by condensing the response information to preference information, also for example done in Joachims (2002), in Gao et al. (2015) for image quality assessment and in Karmaker Santu et al. (2017) for E-commerce.

4 Current techniques to solve ranking problems

This section is divided into six parts. While the first subsection reviews a so-called plug-in approach, each of the four subsequent subsections is devoted to a particular underlying machine learning algorithm for the discussed ranking approach, i.e., Support Vector Machines (SVM), Boosting, trees and Neural Networks resp. Deep Learning. The last subsection contains algorithms that have not been covered by the subsections before.

4.1 Plug-in approaches

Plug-in approaches refer to the strategy to cast a ranking problem as a classification or regression problem and to train a classifier or regressor. Note that in principle any classification or regression algorithm can enter here, depending on the chosen classification or regression loss function.

In the case of bipartite ranking, one 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

\[
\frac{1}{n} \sum_i \log_2(1 + \exp(-2Y_i s_\theta(X_i))).
\]

The resulting function \( s_\theta \) 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, 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 (see also Fürnkranz et al. 2009), which is another aspect why this approach cannot be optimal.

A plug-in approach has also been considered for continuous instance ranking problems, but it suffers from similar weaknesses as the plug-in approach for bipartite instance ranking as pointed out in Sculley (2010) who emphasize that, due to the minimization of a regression loss which does not respect any ordering, the predictions are essentially scattered around the true values which also has been pointed out in Fürnkranz et al. (2009). Agarwal et al. (2010) compare their ranking-based SVM-strategy with a simple plug-in
approach based on SVR and derive that the ranking performance of SVR is inferior to that of their ranking algorithm (although in some cases SVR is indeed superior). Another plug-in approach has been proposed in Mohan et al. (2011) who train gradient-boosted regression trees or random forests for hard multi-partite instance ranking according to the squared loss function, i.e., the outputs should be close to the discrete-valued responses.

4.2 SVM-type approaches

Joachims (2002) provided the RankingSVM algorithm for document retrieval which is essentially based on the seminal approach for ordinal regression introduced in Herbrich et al. (1999a). In the situation of Herbrich et al. (1999a), a set of pairs \((X_i, Y_i)\) is given. Their goal is to solve a hard bipartite ranking problem, but they do not optimize the hard ranking loss directly but formulate 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 Joachims (2002) argue, trying to find a scoring function such that every inequality is satisfied would be NP-hard, so (Herbrich et al. 1999a; Joachims 2002) 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 \(K\) (Herbrich et al. 1999a). Due to the equivalence of SVM problems and structural risk minimization problems with a Hinge loss, the criterion in Joachims (2002) can be translated 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 \(K\) (see for example Schölkopf et al. 2001). They call their algorithm RankingSVM. Note that (Joachims 2002) have a structured instance ranking problem with a query-document data set and aim at fitting a scoring function \(s^{(q)}\) for every query \(q\) 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. They point out that this setting is more flexible than the one in Herbrich et al. (1999a) since it allows for different rankings for different queries.

An implementation of RankingSVM is given in the SVM\textsuperscript{light} software package (Joachims 1999) in C language.\(^2\) (Chen et al. 2017) accelerate the computation of the kernel matrix for the case \(n > p\) by invoking the kernel approximation \(K(x, x') = \langle \Phi(x), \Phi(x') \rangle\) which generates an approximate kernel Hilbert space and provides an SVM solution of the form

\[
s(x) = w^T \Phi(x).
\]

They propose two methods to get a suitable kernel approximation. The first is a Nyström approximation where \(m \ll n\) rows of \(X\), say, \(\hat{X}_1, \ldots, \hat{X}_m\), are sampled uniformly, followed by a singular value decomposition of the matrix \((K(\hat{X}_i, \hat{X}_j))\), followed by a singular value decomposition of the matrix \((K(\hat{X}_i, \hat{X}_j))\). Truncating the SVD by taking just the first \(k\) columns of the orthonormal matrix and the upper left \(k \times k\)

\(^2\) [http://svmlight.joachims.org/](http://svmlight.joachims.org/).
-submatrix of the diagonal matrix, one gets a rank-\( k \)-approximation, reducing the complexity to \( \mathcal{O}(npk + k^3) \). 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 fact that the approximation error is higher than for the Nyström approximation (for equal \( m \)), the complexity is just \( \mathcal{O}(nmp) \). Chen et al. (2017) provide publicly available MATLAB code.3

An improved version of SVMlight for the special case of linear kernels is given in the software package SVMrank relying on the cutting-plane algorithm from (Joachims 2006) for bipartite instance ranking.4 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 pair-wise differences \( X_i - X_j \) which leads to a complexity of \( \mathcal{O}(n^2) \). 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, essentially reducing the respective complexity to \( \mathcal{O}(np) \) for \( n > p \).5 A further improvement for linear SVMs has been proposed by Airola et al. (2011) which results in a linear time complexity of \( \mathcal{O}(np_0 + n \ln(n)) \) for the average number \( p_0 \) of non-zero features per sample, generalizing the method from (Joachims 2006), which was tailored to bipartite instance ranking to continuous instance ranking problems. They employ a cutting-plane algorithm that iteratively computes a linear lower bound for the empirical risk. Considering the pair-wise Hinge loss, they build up on a fast evaluation strategy from (Joachims 2006) for the empirical risk and the subgradients introduced which, however, is only efficient if the label space consists of a few discrete labels. They apply order statistics trees, i.e., binary search trees where the ordering of a pair is represented by the left resp. the right children node and which can be computed in \( \mathcal{O}(\ln(n)) \) time. Using these trees, the coefficients in the representation of the pair-wise Hinge loss from (Joachims 2006) are computed which allows for an efficient computation of the subgradients. Airola et al. (2011) point out that their approach could in principle also be used for kernelized ranking SVMs. Their Python code is publicly available.6 (Lee and Lin 2014) even improve this strategy further by considering trees where the instances are stored in the leaves so the global ranking is represented at the leaf level in contrast to the method of Airola et al. (2011), which employs binary rankings at the nodes. A Python implementation is publicly available.7

Qin et al. (2008a) rank relational objects for web search, i.e., there is a relationship of the objects like hierarchies of URLs. The ranking function therefore does not only respect the features of the objects but also their relations. Let \( R \in \mathbb{R}^{n \times n} \) represent the relationships between the objects so that the training set consists of training samples \((X_i, R_i, Y_i)\) that are i.i.d. realizations of some joint distribution. Let \( h \) be the feature function, so, for a scoring function \( s \), one has \( Y = s(h(X), R) \), i.e., it maps whole matrix-type samples onto the real line. Then one has three different settings, i.e., \( h, s \) or both functions have to be learned, but

---

3 https://github.com/KaenChan/rank-kernel-appr.
4 https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html.
5 http://olivier.chapelle.cc/primal/.
6 http://staff.cs.utu.fi/~aatapa/software/RankSVM/.
7 https://www.csie.ntu.edu.tw/~cjlin/liblinear/exp.html.

© Springer
they address only the first setting, i.e., the inner feature function has to be learned, so the problem is

$$\min_w \left( \sum_i L(s(h(X_i, w), R_i), Y_i) \right).$$

They however discuss how to define $s$ and let

$$s(h(X, w), R) = \arg\min_z (|h(X, w) - z|^2 + \lambda J(R, z))$$

for possible ranking scores $z$, i.e., the first part of the objective encourages the feature representation to already be close to the ranking scores and the second part is a penalty that encourages that similar documents get similar ranking scores. They propose to set

$$J(R, z) = \frac{1}{2} \sum_i \sum_j R_{i,j} (z_i - z_j)^2.$$  

In other words, the first part measures the local inconsistency and the penalty term encourages global consistency w.r.t. $R$. Assuming that $h(X, w) = Xw$, the problem can be solved using an SVM. A different setting is topic distillation where $R$ does not represent similarities but hierarchies, i.e., $R_{i,j} = 1$ if object $i$ is a parent of object $j$, which leads to the loss

$$J(R, z) = \sum_i \sum_j R_{i,j} \exp(z_j - z_i)$$

indicating that a high loss is suffered if a child gets a higher score than a parent. Both objectives can be solved in principle using optimization techniques like SVM, Boosting or NNs, and they detail out the application of RankingSVM for this task, resulting in the algorithm Relational RankingSVM.

Rakotomamonjy (2004) and Ataman and Street (2005) use the fact that the binary instance 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 $X^+_i$ resp. $X^-_i$ for the features. The empirical AUC can be estimated by

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

for $\xi_{ij} := s(X^+_i) - s(X^-_j)$. Using this 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. The computational complexity is essentially $O(n_{+} n_{-})$. Ataman and Street (2005) used a MATLAB and a WEKA implementation.
and the algorithm from (Rakotomamonjy 2004) can be found in a MATLAB toolbox (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

\[
\frac{1}{2}||w||^2 + \frac{C}{2} \sum_{i} \sum_{j} \xi_{ij}^q
\]

for the target function of the SVM, where \( q \in \{1, 2\} \), and the corresponding solutions. A recommendation for the choice of \( q \) 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 provide some suggestions how to reduce the complexity to \( O(n^2) \).

Cortes et al. (2007a) extend the RankingSVM algorithm by directly respecting the magnitudes of the pair-wise differences of the responses in continuous instance ranking. In their MPRank algorithm, the objective is

\[
||s||^2 + \frac{\lambda}{n(n-1)} \sum_{j \neq i} \sum_{j \neq i} ((s(X_i) - s(X_j)) - (Y_i - Y_j))^2,
\]

so the penalty term encourages the predicted difference of the scores to be close to the true differences. They point out that the main computational cost is the matrix inversion required for computing the solution which is in theory of complexity \( O(p^{2+\alpha}) \) in the primal resp. \( O(n^{2+\alpha}) \) in the dual case when applying fast matrix inversion, where \( \alpha \approx 0.376 \). However, due to the size of real data sets, there are essentially no practical implementations of such algorithms so that the true complexity remains to be \( O(p^3) \) resp. \( O(n^3) \). In an extended version (Cortes et al. 2007b), they provide an online variant of MPRank. They also consider the case of an \( \epsilon \)-insensitive penalty which results in the problem

\[
\min_s \left( ||s||^2 + \frac{\lambda}{n(n-1)} \sum_{j \neq i} \sum_{j \neq i} (s_{ij}^e + s_{ij}^e^*) \right),
\]

s.t. \((s(X_i) - s(X_j)) - (Y_i - Y_j) \leq \epsilon + s_{ij}, \quad (Y_i - Y_j) - (s(X_i) - s(X_j)) \leq \epsilon + s_{ij}^*, \quad s_{ij}^e, s_{ij}^e^* \geq 0,\)

which they call SVRank which, however, requires quadratic programming and is therefore more expensive than MPRank. Note that in the reference, one defines \( s(x) = wh(x) \) for a mapping \( h \) from \( \mathcal{X} \) to the RKHS and considers \( ||w||^2 \) in the objective, which is equivalent to our formulation due to \( h \) being fixed. These approaches are tailored to the hard continuous ranking problem and not to the localized variant since there is no evidence that the response differences would be large at the top of the list. They apply their algorithms to movie, joke and book ranking.

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

\[
\frac{1}{n(n-1)} \sum_{i<j} \frac{1}{2} \left| \text{sign} (s(X_i) - s(X_j)) - \text{sign} (Y_i - Y_j) \right| + \lambda ||s||^2_{H,K}
\]

\[\text{http://asi.insa-rouen.fr/enseignants/~arakoto/toolbox/}.\]
for some kernel $\mathcal{K}$ 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 criterion

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

which resembles the one from MPRank. The solution has, by the representer theorem, the form $f(X) = \sum_{i=1}^{n} a_i \mathcal{K}(X, X_i)$. The algorithm is called RankRLS ("regularized least squares"). The complexity of the algorithm is of order $O(p^3 + np^2)$ resulting from matrix inversion and matrix multiplication. Pahikkala et al. (2009) pointed out that RankRLS and MPRank were proposed independently, however, RankRLS is also applicable for query-document data. Moreover, RankRLS is extended in Pahikkala et al. (2009) so that it can directly learn from pair-wise preferences in the sense that one has sequences $(X_1, \ldots, X_n)$ but no absolute responses but that for each $(X_i, X_j)$ where $X_i$ is preferred over $X_j$ a number $Y_b \in \mathbb{R}_{\geq 0}$ is provided which quantifies the magnitude with which $X_i$ is preferred over $X_j$. Representing all such available preference information by vectors $e_h = (X_i, X_j, Y_b)$, they consider the optimization problem

$$\min_{f \in \mathcal{H}_f} \left( \sum_{h} w_h^2 (e_h - g(e_h))^2 + \lambda ||f||^2_{\mathcal{H}_f} \right),$$

where $w_h^2 (e_h - g(e_h))^2$ for $w_h, e_h \geq 0$ approximates the numerically intractable loss function $(1 - \text{sign} (g(e_h)))$ where $g(e_h) := s(X_i) - s(X_j)$. The preference magnitude is not yet incorporated in this loss function, so Pahikkala et al. (2009) propose variants that respect these magnitudes. The complexity of their method is $O(p^3 + np^2)$ as in Pahikkala et al. (2007). 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 RankRLS algorithm of $O(knp)$. They applied their algorithms to document retrieval data. A Python implementation of RankRLS and greedy RankRLS is publicly available.9

Cao et al. (2006) argue that a major weakness of RankingSVM is that misrankings 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.

Wu et al. (2017) propose SVR-based approach for blind image quality assessment by solving

$$\min_{w, \xi, \xi^*, \eta} \left( \frac{1}{2} ||w||^2 + C_1 \sum_{i} \xi_i + C_2 \sum_{i} \xi^*_i + C_3 \sum_{i} \sum_{j} \eta_{ij} \right)$$

s.t. $Y_i - w^T h(X_i) - b \leq e - \xi_i$, $w^T h(X_i) + b - Y_i \leq e + \xi^*_i$, $w^T h(X_i) - w^T h(X_j) \geq e - \eta_{ij}$, $\xi_i, \xi^*_i, \eta_{ij} \geq 0$.

9 https://github.com/aatapa/RLScore.
for all \(i, j\), so \(s(x) = w^T h(x) + b\) for a mapping \(h\) from \(\mathcal{X}\) into the feature space. As usual, this problem is translated into the dual problem which is solved using quadratic programming and which can be kernelized.

Jung et al. (2011) provide an ensemble approach called Ensemble RankingSVM by combining different RankingSVM models in structured instance ranking. More precisely, they train RankingSVM for each query-specific training set and select the best models using a validation set and invoking MAP as quality criterion. Due to the ensemble training process, the computational complexity is remarkably reduced to \(O(n^2/|Q|)\) for the query set \(Q\). Experiments are done on the OHSUMED data set. A similar idea has been suggested in Qin et al. (2007), who point out that RankingSVM constructs one hyperplane which is insufficient in complex settings, so they propose to learn multiple hyperplanes and to perform rank aggregation afterwards. The idea is to learn a base ranker for each instance pair, directly exploiting the correspondence of pair-wise ranking and binary classification, and to perform rank aggregation using the BordaCount model which assigns a score to each instance which equals the number of instances in the ranked list (generated by each base ranker) which are ranked lower. Their algorithm is applied to the OHSUMED and a definition search data set.

An approach for hard continuous instance ranking problems has been introduced in Agarwal et al. (2010). They argue similarly as (Cortes et al. 2007a) that the magnitude of the response differences should enter the loss function and consider the loss

\[
\frac{1}{n(n-1)} \sum_i \sum_{j \neq i} (Y_i - Y_j) (I(s(X_i) < s(X_j)) + 0.5I(s(X_i) = s(X_j))) I(Y_i > Y_j).
\]

Since this loss cannot be minimized directly, they propose an SVM-strategy where

\[
\frac{1}{n(n-1)} \sum_i \sum_{j \neq i} \max(0, (Y_i - Y_j - s(X_i) + s(X_j))) I(Y_i > Y_j) + \frac{1}{2C} ||w||_q^2 + \frac{1}{2C} ||w||_p^2 + C \sum_i \sum_j \xi_{ij}
\]

is minimized which is done by solving the dual quadratic program. The solution can be expressed similarly as for a standard SVM due to the representer theorem. The complexity is \(O(n^2)\). They rank chemical structures for drug discovery where the \(pIC_{50}\)-value serves as response.

Since SVM-type solutions are not sparse, there are several approaches to construct SVM-type ranking functions with feature selection.

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

\[
||w||_q^q + C \sum_i \sum_j \xi_{ij}
\]

for \(0 < q < 1\), 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 \(l_q\)-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 representants, reducing the number of constraints and therefore the complexity to \(O(n^2)\), similarly as in Brefeld and Scheffer (2005). They apply their algorithm to a heart disease and two credit risk data sets.
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. Their experiments are run on document retrieval data sets.

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 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.\(^{10}\)

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. Most of these SVM-type ranking algorithms are tailored to bipartite ranking problems but some of them are designed for continuous ranking problems. Cortes et al. (2007a) essentially also applied their algorithm to hard multi-partite instance ranking, treating the class labels as numerical values. The magnitude-preserving approaches from (Pahikkala et al. 2007; Cortes et al. 2007a) and (Agarwal 2010) may be appropriate for localized instance ranking provided that the distribution of the response values is positively skewed which would encourage a better prediction of the scores of the top instances, but it would in fact concentrate on the bottom of the list in the case of a negatively skewed distribution. Note that 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 mostly a pair-wise Hinge or pair-wise squared loss or a modification or either one.

### 4.3 Boosting-type approaches

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) benefiting from the binarity of the response variable. First, they propose a distribution $F$ on the space $\mathcal{X} \times \mathcal{X}$ which, for data $D$, is represented as a matrix that essentially contains weights. These weights can be thought of as 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

$$\sum_{i \neq j : X_i > X_j} F(X_i, X_j)(\tilde{s}(X_i) - \til{s}(X_j))$$

\(^{10}\) [http://remi.flamary.com/soft/soft-ranksvm-nc.html](http://remi.flamary.com/soft/soft-ranksvm-nc.html).
is maximized. 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. Freund et al. (2003) show that the complexity of RankBoost for bipartite instance ranking can essentially be reduced to $O(m(n_p + n_p))$ for the number $m$ of Boosting iterations. 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

$$\frac{2}{n(n-1)} \sum_i \left( \sum_{j>i} \exp((Y_i - Y_j)(s(X_i) - s(X_j))) \right)^q$$

for some $q \geq 1$. The argument behind this power loss given in Rudin (2009) is that the higher $q$ 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 $q = \infty$ has been studied in Rakotomamonjy (2012). As for the complexity, Rudin (2009) point out that any AdaBoost or RankBoost implementation may directly be used.

So, while RankBoost is tailored to hard bipartite instance ranking problems (and may be used for $d$-partite instance ranking problems in the sense of Clémençon et al. (2013c)), the p-Norm-Push is closest to handle localized bipartite instance 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.

Zheng et al. (2008b) assume that both pair-wise preferences as well as ranking scores are given, arguing that, for example, in document retrieval it can be difficult to translate relevance scores (categorical responses) into an ordering. Their combined loss function intends to both recover the true ordering as well as the ranking scores, so it combines a ranking loss with a regression loss, resulting in

$$L(x, y, s) = \frac{w}{2} \sum_{(i,j): X_i > X_j} \sum_j (\max(0, s(X_j) - s(X_i) + \xi))^2 + \frac{1-w}{2} \sum_i (Y_i - s(X_i))^2$$

for a margin parameter $\xi > 0$ and preference weights $w$ that can be selected using cross validation. Applying a Taylor expansion and distinguishing two cases for the non-differentiable Hinge surrogate, they derive a sufficiently regular quadratic upper bound of this
empirical risk which is minimized using Gradient Boosting (Friedman 2001; Bühlmann and Hothorn 2007) with regression base-learners. Their algorithm is called QBRank. They apply their algorithm to document retrieval data. They extend their work (Zheng et al. 2007) where the GBRank algorithm based on Gradient Boosting without approximation and without combining the regression and the ranking objective was introduced. The complexity is given by \( O(n^2 p m) \) for the number \( m \) of Boosting iterations.

GBRank is extended in Chen et al. (2010) who provide the GBlend algorithm. They address the problem that instances, here documents, stem from \( m_{\text{max}} \) different domains so that one would usually learn a scoring function \( s^{(m)} \) for each domain individually. If the range of the predicted ranking scores however strongly differs across the domains, the overall ranking loss can indeed be high although the domain-specific ranking losses are low. They propose to jointly optimize the scoring functions, regularized with a penalty term that encourages instances with similar response value to receive a similar score, leading to the overall objective

\[
\sum_{X_i \in X^{\text{train}}} \sum_{X_j \in X^{\text{train}}: Y_i > Y_j} \max(0, (s^{(m)}(X_i) - s^{(m')}(X_j) - \xi_{ij})^2) + \lambda \sum_{X_i \in X^{\text{train}}} \sum_{X_j \in X^{\text{train}}: Y_i = Y_j} (s^{(m)}(X_i) - s^{(m')}(X_j))^2
\]

for margins \( \xi_{ij} > 0 \) which is tailored to \( d \)-partite ranking due to the equality constraint in the regularizer. In potentially could extended to continuous instance ranking by invoking the magnitude-preserving paradigm of Cortes et al. (2007a), but the unbounded domain lets the Hinge loss appear to be a poor surrogate for the indicator ranking loss. The Boosting algorithm is generally of complexity \( O(mn^2 p) \) for the number \( m \) of iterations.

Zhou et al. (2008) explicitly include ties in their surrogate loss functions. Based on the Bradley–Terry model and the Thurstone—Mosteller model, they propose surrogate loss functions so that the objective becomes

\[
\sum_{i} \sum_{j} \ln((1 + \rho \exp(s(Y_i) - s(Y_j))) I(Y_j > Y_i)
\]

\[
+ \ln((1 + \rho \exp(s(Y_i) - s(Y_j))) + \ln((1 + \rho \exp(s(Y_j) - s(Y_i))) - \ln(\rho^2 - 1) I(Y_i = Y_j)
\]

for some \( \rho \geq 1 \) resp.

\[
\sum_{i} \sum_{j} -\ln(\Phi(s(Y_j) - s(Y_j) - \epsilon)) I(Y_j > Y_i)
\]

\[
+ [-\ln(\Phi(s(X_j) - s(X_j) + \epsilon) - \ln(\Phi(s(X_j) - s(X_j) - \epsilon)))] I(Y_i = Y_j)
\]

for some threshold \( \epsilon \geq 0 \) which relates to the parameter \( \rho \) by \( \rho = \exp(\epsilon) \) and the cumulative distribution function \( \Phi \) of the standard normal distribution. The empirical risk is minimized using Gradient Boosting with regression base learners whose complexity is \( O(n^2 mp) \) for the number \( m \) of Boosting iterations. The approach is mainly designed for multi-partite or bipartite hard instance ranking problems, but they experimentally derive that the performance on the top documents improves which at least partially makes their approach applicable to localized multi-partite instance ranking problems.

Qin et al. (2008b) consider structured instance ranking. For the ground truth list \( Y^{(q)} \) for a query \( q \), the cosine surrogate is given by

\[
L(Y^{(q)}, \hat{Y}^{(q)}) = \frac{1}{2} (1 - \cos(Y^{(q)}, \hat{Y}^{(q)}))
\]
and the overall objective is the sum of these query-wise losses which leads to the **RankCosine** algorithm which is a Boosting-type algorithm combining weak rankers (which may be the same as in **RankBoost**). Although they specify their algorithm for permutation outputs that would correspond to object ranking, having real-valued ranking scores available, their algorithm is directly applicable to hard continuous instance ranking. The complexity is $O(mkQn_{\max}p)$ for the number $k$ of candidate weak learners, the number $m$ of training iterations and the maximal number $n_{\max}$ of documents per query.

Tsai et al. (2020) criticize that the cross-entropy loss as used in Burges et al. (2005) or Cao et al. (2007) is unbounded and propose the fidelity loss which leads to the objective for the number $#q$ of document pairs for query $q$. They develop an AdaBoost-type algorithm for minimizing this objective. This FRank algorithm can be applied to hard binary and multi-partite instance ranking problems and seems to be better suited for hard continuous instance ranking problems than **RankBoost** due to the boundedness of the loss function. The complexity of FRank is $O(n^2Qmp)$ for the number of iterations.

An approach based on isotonic regression for ranking with query-dependent data has been introduced in Zheng et al. (2008a). Assuming an initial guess $s_0$ for the scoring function, they aim at iteratively modifying the current scoring function $s_m$ by computing

$$s_{m+1} = s_m + \eta g_m$$

with learning rate $\eta$ and baselearner $g_m$. The baselearner is a regression tree which minimizes

$$\sum_i (g_m(X_i) - \delta_i^m)^2$$

where the $\delta_i^m$ are computed by isotonic regression, minimizing

$$\sum_i (\delta_i^{(m)})^2 + \lambda \varepsilon^2,$$

s.t. $s_m(X_i) + \delta_i^m \geq s_m(X_j) + \delta_j^m + \Delta G_{ij}(1 - \zeta), \zeta \geq 0$, for the margin $\Delta G_{ij}$ between the responses for $X_i$ and $X_j$. The problem can be solved in $O(n^3)$ steps, but they mention that the complexity may be reduced to $O(n^2)$. The approach is called **IsoRank**. They further outline the application of this approach in the presence of ties.

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 instance 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 (note that the unboundedness has already been pointed out in Qin et al. (2008b)). Another weakness would be the necessity to evaluate the gradients of the pairwise loss (which are sums themselves) in each Boosting iteration, making the algorithm computationally expensive.

To handle these issues, Werner (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 componentwise linear baselearners (Bühlmann and Yu 2003; 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 (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 strong model most is selected.

However, 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. 2015) would be necessary which is outlined in Werner (2019) where a modified Stability Selection is proposed. This approach is, to the best of our knowledge, the first one that tries to find a stable predictor set for ranking. While the original approach has a complexity of $O(mn \ln(n)p)$, the aggregation of $B$ such Boosting models in a Stability Selection leads to $B$ times the respective complexity. However, in its current implementation in the R-package gfboost, it is mainly designed for the hard continuous instance ranking problem. Nevertheless, there is no restriction to apply their strategy to bipartite and $d$-partite continuous ranking problems if the underlying $L_2$-Boosting algorithm is replaced by a suitable variant like LogitBoost or MultiLogitBoost. Note that a Bagging-type approach has already been proposed in Ganjisaffar et al. (2011) and Burges et al. (2011) who combine LambdaMart models (Wu et al. 2008; Burges et al. 2007) which, however, are tailored to object ranking since the virtual gradient requires the ranks as input. Note that their Bagging algorithm is a classical Bagging (Breiman 1996) that aggregates the predictions and not the models as Stability Selection does.

The strategy of Fürnkranz et al. (2009), building up on the work of Fürnkranz (2002), for multi-partite instance ranking problems is not a Boosting-type approach at the first glance, but they essentially combine different weak (classification) models to get a suitable ranking model (see Sect. 3.1 for more details). Alternatively, Fürnkranz et al. (2009) suggest learning a binary classification model for each pair of classes and to sum up the individual scores, maybe even in a weighted fashion, to get the final score. As binary classifier, they use logit models. In principle, there is no restriction that prohibits an application of classification algorithms that perform model selection which, however, would lead to the question how to get a suitable aggregated predictor set.

### 4.4 Tree-type approaches

Clémençon and Vayatis (2008, 2010, 2009), for instance, also concentrate on AUC maximization to solve binary instance ranking problems as for example (Rakotomamonjy 2004), but in a stricter and more sophisticated way. Given the true conditional probability
\[ \eta(x) = P(Y = 1|X = x) \] and a scoring function \( s \), they introduce metrics on the ROC space which are

\[ d_1(s, \eta) \coloneqq \int_0^1 |ROC^*(\alpha) - ROC_s(\alpha)| d\alpha \]

and

\[ d_\infty(s, \eta) \coloneqq \sup_{\alpha \in [0, 1]} (|ROC^*(\alpha) - ROC_s(\alpha)|) \]

where \( ROC^* \) is the optimal ROC curve and \( ROC_s \) the ROC curve induced by the scoring function \( s \). Note that the absolute value in the supremum is not necessary since per definition the optimal ROC curve dominates every competitor ROC curve. The idea in the cited references is to optimize the ROC curve according to \( d_\infty \), i.e., in an \( L_\infty \)-sense due to the disadvantage that an \( L_1 \)-optimization is nothing but an AUC-optimization due to

An AUC-optimization is not appropriate according to the authors since different ROC curves can have the same AUC.

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 instance ranking problems as well as to localized binary instance ranking problems.

As for binary instance ranking problems, they provided TreeRank and RankOver (Clémençon and Vayatis 2008, 2010). The idea behind the TreeRank algorithm is to divide the feature space \( \mathcal{X} \) into disjoint parts \( \mathcal{P}_j \) and to construct a piece-wise constant scoring function

\[ s_N(x) = \sum_{j=1}^N a_j I(x \in \mathcal{P}_j) \]

for \( a_1 > \cdots > a_N \). 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 TreeRank algorithm then recursively adds nodes between all existing nodes such that the ROC curve approximates the optimal ROC curve by splitting each region \( \mathcal{P}_j \) in two parts. More precisely, one starts with the region \( \mathcal{P}_{0,0} = \mathcal{X} \) and the coefficients \( a_{0,1} = \beta_{0,1} = 1 \). In each stage \( b = 0, \ldots, D - 1 \) of the tree and in every iteration \( k = 0, \ldots, 2^b - 1 \), one computes the estimates

\[ \hat{\alpha}(\mathcal{P}_{b,k}) \coloneqq \frac{1}{n_-} \sum_{i} I(X_i \in \mathcal{P}_{b,k}, Y_i = -1) \]

\[ \hat{\beta}(\mathcal{P}_{b,k}) \coloneqq \frac{1}{n_+} \sum_{i} I(X_i \in \mathcal{P}_{b,k}, Y_i = 1) \]

and optimizes the entropy measure

\[ Ent(\mathcal{P}_{b,k}) \coloneqq (a_{b,k+1} - a_{b,k}) \hat{\beta}(\mathcal{P}_{b,k}) - (\beta_{b,k+1} - \beta_{b,k}) \hat{\alpha}(\mathcal{P}_{b,k}) \]
by finding a subset of $P_{b,k}$ which maximizes this empirical entropy. The coefficients are updated recursively. Clémençon and Vayatis (2008) already mention that TreeRank may be used as weak ranker for a Boosting-type approach.

Extensions by combining the TreeRank algorithm in combination with bagging resp. in a RandomForest-like sense are given in Clémençon et al. (2009), Clémençon et al. (2013a). A crucial question is how to combine the rankings predicted by the $B$ different trees. This leads to a so-called Kemeny aggregation (Kemeny 1959, see also (Korba et al. 2017) for theoretical aspects of rank aggregation) where a consensus ranking is computed.

Having some distance measure $D$ which in Clémençon et al. (2009) and Clémençon et al. (2013a) may be a Spearman correlation or Kendall’s $\tau$, the consensus ranking, represented by a permutation $\pi^* \in \text{Perm}(1 : n)$, is the solution of

$$\sum_{b=1}^{B} D(\hat{\pi}_b, \pi) = \min_{\pi}$$

for the predicted permutations $\hat{\pi}_b$ for tree $b$, respectively. As for the RandomForest-type approach (“Ranking forest”), Clémençon et al. (2013a) make two suggestions how to randomize the features in each node.

As for the pruning of ranking trees, we refer to (Clémençon et al. 2011) and (Clémençon et al. 2013a), 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

$$\frac{\text{AUC}_{T_{\text{sub}}}}{\lambda |T_{\text{sub}}|}$$

where $s_T$ denotes the scoring function corresponding to tree $T$.

The TreeRank algorithm has been available in the R-package TreeRank, but it had been removed. Nevertheless, the source code is still available.\(^{11}\)

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.

These tree-type algorithms are tailored to bipartite instance ranking problems. However, as pointed out in Clémençon et al. (2013b), they can be used for local AUC optimization (see Definition 3), so they are applicable for both hard and localized bipartite instance ranking problems while the AUC-maximizing competitors show inferior local ranking performance in the simulation studies of Clémençon et al. (2013b).

As for the $d$-partite instance ranking problems, Clémençon et al. (2013c) build up on the strategy of Fürnkranz (2002); Fürnkranz et al. (2009) that they can be regarded as collection of bipartite Ranking problems if one considers approaches like one-versus-all or one-versus-one. In Clémençon et al. (2013c), they apply different algorithms tailored to bipartite ranking problems like TreeRank, RankBoost or RankingSVM and evaluate their performance in the VUS criterion.

\(^{11}\) https://github.com/cran/TreeRank.
However, since the algorithms are not designed for VUS-optimization, Clémençon and Robbiano (2015b) modify their TreeRank algorithm such that the splits of 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 TreeRankTournament algorithm therefore is applicable to the hard $d$-partite instance ranking problem. Clémençon and Robbiano (2015a) provide a bagged and a RandomForest-type version of this algorithm, analogously to the bagged trees for the bipartite case.

Clémençon and Achab (2017) propose a tree-type algorithm for hard continuous instance ranking problems. Let w.l.o.g. $Y \in [0, 1]$. Then each subproblem

$$\max_{\alpha}(P(s(X) > t | Y > y) - P(s(X) > t | Y < y))$$

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 instance ranking problem, so the continuous instance ranking problem can be regarded as a so-called "continuum" of bipartite instance ranking problems (Clémençon and Achab 2017).

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

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

where ROC 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 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 forth, so the rightmost leaf indicates the bottom instance. Clémençon and Achab (2017) already announced a forthcoming paper where a RandomForest-type approach for 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 instance ranking problems.

4.5 Approaches with neural networks and deep learning

Burges et al. (2005) suggest defining 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
and where $p_{ij}$ is the analog for the theoretical differences based on the observed ranking scores. 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:

$$
 s(X_i) := h^{(3)} \left( \sum_j w^{(32)}_{ij} h^{(2)} \left( \sum_k w^{(21)}_{jk} X_k + b^{(2)}_j \right) + b^{(3)}_i \right).
$$

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. RankNet is tailored to the hard bipartite instance ranking problem and the experiments in Burges et al. (2005) are based on document retrieval data. It is available at the RankLib library (Dang 2013). A speedup has been proposed in Burges et al. (2007). Köppel et al. (2019) generalize RankNet with the algorithm DirectRanker which allows for any strictly monotone output activation function and which guarantees that a reflexive, anti-symmetric and transitive ranking function is learned. Since the complexity of neural networks depends on the architecture (for example, the number of hidden layers and hidden nodes, the optimizer, the application of Dropout etc.), we omit the complexity throughout this subsection.

Cao et al. (2007) extend RankNet to ListNet for query lists as instances and enable to concentrate on the top-$K$-probability, i.e., the probability that an instance belongs to the top-$K$-ones. For query $q$, the objective is

$$
 - \sum_i \prod_{g \in G_k} \sum_{j=1}^{n(g)} \frac{\exp(Y^{(q)}_{k})}{\prod_{j=1}^{n(g)} \exp(Y^{(q)}_{k})} \cdot \ln \left( \frac{1}{\sum_{j=1}^{n(g)} \exp(s(X^{(q)}_{k}))} \prod_{j=1}^{n(g)} \exp(s(X^{(q)}_{k})) \right)
$$

where $G_k$ is the set of all permutations which are correct on the top-$K$-instances. The objective is minimized via gradient descent where $s$ is a linear neural network. Their algorithm can be applied to hard instance ranking problems of all types and seems to be applicable even for localized versions due to considering top-$K$-probabilities. Ai et al. (2018) modified the approach of Cao et al. (2007) and replace the exponential functions by rectified exponential units $\exp(x) I(x > 0)$.

For hard binary instance ranking, Severyn and Moschitti (2015) minimize the regularized point-wise cross-entropy loss

$$
 - \sum_i Y_i \ln \left( \frac{\exp(s^{-1}(X_i))}{\sum_k \exp(s^{-1}(X_i))} \right) + (1 - Y_i) \ln \left( \frac{\exp(s(X_i))}{\sum_k \exp(s(X_i))} \right) + \lambda \| \theta \|^2
$$

where $\theta_k$ is the parameter for class $k$ and where the underlying model is a convolutional neural network. The loss is minimized via stochastic gradient descent. They also discuss to enhance the training procedure with Dropout (Srivastava et al. 2014).

The approach in Guo et al. (2016) originates from object ranking but can directly be translated to hard binary, multi-partite and partially to hard continuous instance ranking (due to the unbounded loss function). The loss is
which is minimized by a neural network architecture which first learns local query-document interactions using matching histogram mapping, matching patterns via a feed-forward neural network and query importance using a term gating network. A similar approach is given in Xiong et al. (2017), who use a different architecture based on kernel pooling and Pang et al. (2017), who learn query-specific feature representations for the instances.

Zhang et al. (2019) propose a graph-embedding approach for document ranking based on click-graph features for an application in product search. Given a hard bipartite ranking setting, they jointly train a multilevel perceptron-based encoder which transforms the semantic features $X(q)$ into graph embeddings $\hat{V}(q)$ and a scoring function by minimizing

$$
\frac{1}{n(n-1)} \sum_i \sum_{j \neq i} \max(0, I(Y_i > Y_j)(1 - s(X_j) - s(X_i)))
$$

for the true graph embeddings $V(q)$ and the smoothed Hinge loss $L^{\text{smooth Hinge}}$ from (Rennie 2005).

Pei et al. (2019) consider the hard binary ranking problem for personalized re-ranking. For binary relevance responses $Y_i$, they minimize the loss

$$
- \sum_q \sum_{i \in I_q} Y_i \ln(P(Y = Y_i|X, PV, \theta))
$$

w.r.t. the parameter $\theta$ of the NN and for a personalized matrix $PV$ that encodes the mutual influences between the instances. The probabilities $P(Y = Y_i|X, PV, \theta)$ are predicted using softmax activation in the output layer.

Song et al. (2016) introduce an approach based on gradients of the expected loss. Their work is based on Hazan et al. (2010) who proved that

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

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 these results for non-linear and non-convex functions. In fact, Song et al. (2016) apply their results to bipartite hard instance ranking problems by setting

\[\nabla_\theta \mathbb{E}[L(Y, s_\theta(X))] = \lim_{\epsilon \to 0} \left( \frac{1}{\epsilon} \mathbb{E}[\nabla_\theta F(X, Y_{\text{direct}}, \theta) - \nabla_\theta F(X, Y_{\theta}, \theta)] \right)\]
for the ranking rule introduced in Definition 1 and by invoking the loss function

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

where \( \hat{s}_\theta \) 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.

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 and argue that since they are all rank-based, i.e., depend on \( \text{rk} (Y) \) and \( \text{rk} (\hat{Y}) \), 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), \( s^{(1)} \) needs to be trained on synthetic training data, using a sorting Deep Neural Network. Having real-valued scores, they propose the surrogate loss

\[ \sum_i ||s^{(1)}(s^{(2)}(X_i)) - \text{rk} (Y_i)||^2_2 \]

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

\[ \sum_{k=1}^d \langle s^{(1)}(s^{(2)}(Y)_k), Y_k \rangle \]

based on the Mean Average Precision, where \( Y_k \) is a binary vector with ones where the respective component of \( Y \) is from class \( k \) and where \( s^{(2)}(Y)_k \) is considered to be the score vector for class \( k \). They also propose a surrogate for the Recall criterion. The scoring function \( s^{(2)} \) 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) ranking problems and does not make requirements for \( Y \).

All these approaches share the advantages of Deep Neural Networks, i.e., the automatic feature abstraction as well as a potentially high predictive performance, but suffer from the common disadvantages of these algorithms, i.e., they do not perform variable selection and are very difficult to interpret.
4.6 Other approaches

Yeh et al. (2007) consider query-document pairs \((q, d_i^{(q)})\) and aim at finding a scoring function \(s\) that assigns a relevance score to each pair, indicating how relevant document \(d_i^{(q)}\) is for query \(q\). Assuming that relevance scores \(Y^{(q)}\) are available for each pair \((q, d_i^{(q)})\), they apply a genetic algorithm to infer \(s\) according to the MAP as goodness criterion. Since they restrict themselves to binary-valued responses \(Y^{(q)}\), their algorithm, called RankGP, is tailored to hard binary instance ranking.

Crammer and Singer (2001) cast the ranking problem as ordinal regression problem where the number of ordered classes equals the number of instances. The real-valued scores predicted by a Perceptron are binned into \(n\) categories according to thresholds that are learned during training. Although their algorithm PRanking is tailored to object ranking since it assumes a permutation as ground-truth information, there would be no hindrance to apply it to hard continuous instance ranking by adapting the number of categories.

Zha et al. (2006) consider the list-wise approach and point out that a global scoring function that assigns a relevance score to each pair, indicating how relevant document \(d_j\) is for query \(q\), is tailored to object ranking since it

Moon et al. (2010) criticize that the usual list-wise paradigm weights the ranking performance solely according to the number of documents associated with a query but does not take the relevance of the queries themselves into account. They extend the isotone regression approach from (Zheng et al. 2008a). Given discrete-valued responses for each query-document pair, they relax the separation constraint from IsoRank and encourage...
the scoring function to map instances with similar responses to similar ranking scores (as already outlined in Zheng et al. (2008a)). More precisely, they solve

$$\min_{\delta, \epsilon, \xi} \left( \frac{1}{2} ||\delta||^2 + \frac{\lambda_1}{2} ||\epsilon||^2 + \frac{\lambda_2}{2} ||\xi||^2 \right)$$

s.t. $s(X_i) + \delta_i - s(X_j) - \delta_j \geq \Delta G_{ij} - \epsilon_{c_i}$, $|s(X_k) + \delta_k - s(X_l) - \delta_l| \leq \xi_{c_k}$

for relevance classes $c_i$ and index pairs $(i, j)$ corresponding to different classes and index pairs $(k, l)$ corresponding to the same class. They prove in Moon et al. (2010, Thm. 5) an equivalent problem formulation and solve it using a logarithmic barrier function which is of complexity $O(n \ln(n))$. Optionally, they consider an additional pointwise penalty encouraging the $s(X_i)$ to be close to the $Y_i$. The algorithm is called IntervalRank.

Sculley (2010) optimize a loss function that combined a ranking and a regression objective which they derive to be beneficial for the regression performance when facing rare events. The combined regularized loss function is

$$\frac{\alpha}{n} \sum_i L^{reg}(Y_i, s_\theta(X_i)) + \frac{1 - \alpha}{n(n-1)} \sum_i \sum_{j \neq i} L^{rank}(Y_i, Y_j, s_\theta(X_i), s_\theta(X_j)) + \frac{\lambda}{2} ||\theta||^2$$

for a regression loss function $L^{reg}$, a ranking loss function $L^{rank}$ and $\alpha \in [0, 1]$. They consider the squared or the logistic loss for regression and convex surrogates for the indicator ranking loss like the squared loss, the Hinge loss or the logistic loss and apply the SGD strategy from Sculley (2009) in order to avoid the computational costs to evaluate the objective function. The overall training cost is of order $O(mp_0 + n^2)$ for the number $m$ of training iterations and the maximal number $p_0$ of non-zero entries in the feature vectors. Their algorithm is call CRR and publicly available.12

Xu et al. (2016b) apply a RankBoost variant to image quality assessment. Assuming a linear scoring function $s_\theta(x) = \theta^T x$, they optimize

$$\frac{1}{n(n-1)} \sum_i \sum_{j \neq i} \max(0, Y_i - Y_j) \exp(s_\theta(X_j) - s_\theta(X_i))$$

which is however done directly by gradient descent and does not invoke base learners. Ma et al. (2016) similarly optimize

$$\frac{1}{n(n-1)} \sum_i \sum_{j \neq i} I(Y_i < Y_j) \exp(s_\theta(X_j) - s_\theta(X_i))$$

This approach has been extended in Xu et al. (2016a) where the data are clustered in advance and where cluster-individual scoring functions are computed.

12 http://code.google.com/p/sofia-ml.
5 Discussion

5.1 Discussion of the different ranking problems

In this subsection, we discuss 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 multi-partite 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 multi-partite 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, see also Fürnkranz et al. (2009).

Note that, on the other hand, due to the discretization, ordinal regression problems can indeed be perfectly solved even if the rankings provided by the scoring function are not perfect. For example, consider observations with indices \( i_1, \ldots, i_{n_k} \) that belong to class \( k \). If for a scoring rule \( s \) we had the predicted ordering \( s(X_{i_1}) < s(X_{i_2}) < \cdots < s(X_{i_{n_k}}) \) but the true ordering is different, then we can still choose thresholds such that all \( n_k \) instances that belong to class \( k \) (and no other instance) are classified into this class, provided that \( s(X_i) \notin [s(X_{i_1}), s(X_{i_{n_k}})] \forall i \notin \{i_1, \ldots, i_{n_k}\} \). Fürnkranz et al. (2009) argued that the class labels could in principle also be used as ranking scores but that this strategy would clearly lead to many ties. Though, as Robbiano (2013) already pointed out, the ordinal regression is based on another loss function.

Concerning informativity, one can state that multi-partite ranking problems are more informative than ordinal regression problems due to the chunking operation that is performed for the latter ones. But in fact, in an intermediate step, i.e., when having computed the scoring function, the ordinal regression problem is as informative as multi-partite 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 at least mathematically 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. As for informativity, see also Fürnkranz et al. (2009), who point out that classification is less informative than ordinal regression (and therefore ranking) but that regression may make too strong assumptions like requiring that one can compute meaningful differences between the numerical class values.

A similar discussion can be found in Hüllermeier and Fürnkranz (2010) in the context of label ranking. In contrast to classification where a model picks one of the labels, the goal in label ranking is to predict an ordering on the label set for each instance. They point out that a classification model predicts the most probable class but sorting the labels according to the predicted probabilities that a particular instance belongs to the respective class similarly induces a ranking on the label set.

The continuous instance 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 instance ranking problems, are easy to distinguish and are all meaningful. Hard bipartite and hard $d$-partite instance ranking problems are essentially solved by most of the algorithms that we described in Sect. 4 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 if class $d$ represents the ”rubbish class”.

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 instance 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 a bina-

try which cannot be reasonably translated to 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.

6 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 analyst. We analyzed and discussed the corresponding loss functions resp. quality criteria and carefully discussed different types of instance ranking problems and distinguished instance ranking problems from object and label ranking problems.

Section 4 contains a detailed review of existing learning algorithms for instance ranking based on the empirical resp. structural risk minimization principle in a unified notation, grouped by the underlying machine learning algorithm.

We discussed, from a qualitative perspective, the different combinations of ranking problems and emphasized that it is usually not trivial how to select the type of ranking problem which is most appropriate for a given application.
6.1 Open problems

Despite a vast variety of approaches to solve instance 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 in the sense of robust statistics (e.g. Huber and Ronchetti 2009; Hampel et al. 2011). 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 is categorical or continuous 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 interpretability of the model is still an ongoing research question.

An even more difficult situation arises once the response variable is multivariate, i.e., one has $Y \subset \mathbb{R}^k$, $k \geq 2$. Then one can clearly get partial rankings (not to be confused with partial orders in Cheng et al. (2010) which reflect the uncertainty that prohibit a clear ordering) which are rankings for each column of $Y$ separately. However, since one is actually interested in the ranking of the rows $X_i$ which in the case of univariate responses just equals the ranking of the $Y_i$, it remains to find an overall ranking for the $X_i$ in the case that each response column corresponds potentially to a different ranking. There are many situations where one has partial rankings and wants to get a suitable combined ranking based on these partial rankings. Such situations range from the ranking of web-sites by different search engines (Dwork et al. 2001) to the combination of judge grades in competitions (Davenport and Lovell 2005) and even to applications in nanotoxicology (Patel et al. 2013). The aggregation of the partial rankings gets even more difficult if the quality of the partial rankers is different (Deng et al. 2014). A standard approach is to compute an consensus ranking using for example the Kemeny aggregation (Kemeny 1959), an excellent reference for ranking scores in competitions and also ranking aggregation is given by Langville and Meyer (2012). However, if additionally sparse (and stable) model selection is desired, one has to find a suitable predictor set w.r.t. all response columns which for regression has already been done by Lutz et al. (2008)). A first idea to solve this problem has been outlined in Werner (2019).

Acknowledgements I thank three anonymous referees for their valuable comments that helped to significantly improve the quality of the paper.

Author Contributions Not applicable (single author).

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflicts of interest The author’s declare that they have no conflict of interest.

Code availability Not applicable (the code in the footnotes are just references but not my own code, therefore, I am not responsible for these sources).

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long
as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Agarwal, S. (2010). Learning to rank on graphs. *Machine Learning, 81*(3), 333–357.

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

Agarwal, S., & Sengupta, S. (2009). Ranking genes by relevance to a disease. In *Proceedings of the 8th annual international conference on computational systems bioinformatics*.

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

Agarwal, S., Dugar, D., & Sengupta, S. (2010). Ranking chemical structures for drug discovery: a new machine learning approach. *Journal of Chemical Information and Modeling, 50*(5), 716–731.

Ai, Q., Bi, K., Guo, J., & Croft, W. B. (2018). Learning a deep listwise context model for ranking refinement. In *The 41st international ACM SIGIR conference on research & development in information retrieval* (pp. 135–144).

Ai, Q., Wang, X., Bruch, S., Golbandi, N., Bendersky, M., & Najork, M. (2019). Learning groupwise multivariate scoring functions using deep neural networks. In *Proceedings of the 2019 ACM SIGIR conference on theory of information retrieval* (pp. 85–92).

Ailon, N., & Mohri, M. (2007). An efficient reduction of ranking to classification. *arXiv preprint arXiv: 0710.2889*.

Airola, A., Pahikkala, T., & Salakoski, T. (2011). Training linear ranking SVMs in linearithmic time using red-black trees. *Pattern Recognition Letters, 32*(9), 1328–1336.

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

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

Ataman, K., & Street, W. N. (2005). Optimizing area under the ROC curve using ranking SVMs. In *Proceedings of international conference on knowledge discovery in data mining*.

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

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

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

Breiman, L., & Scheffer, T. (2005). AUC maximizing support vector learning. In *Proceedings of the ICML 2005 workshop on ROC analysis in machine learning*.

Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140.

Bruch, S., Wang, X., Bendersky, M., & Najork, M. (2019). An analysis of the softmax cross entropy loss for learning-to-rank with binary relevance. In *Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval* (pp. 75–78).

Bühlmann, P. (2006). Boosting for high-dimensional linear models. *The Annals of Statistics, 34*(2), 559–583.

Bühlmann, P., & Hothorn, T. (2007). Boosting algorithms: regularization, prediction and model fitting. *Statistical Science, 22*(4), 477–505.

Bühlmann, P., & Van De Geer, S. (2011). *Statistics for high-dimensional data: methods, theory and applications*. Berlin: Springer.

Bühlmann, P., & Yu, B. (2003). Boosting with the $l_1$ loss: regression and classification. *Journal of the American Statistical Association, 98*(462), 324–339.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on machine learning* (pp. 89–96). ACM.
Burges, C., Svore, K., Bennett, P., Pastusiak, A., & Wu, Q. (2011). Learning to rank using an ensemble of lambda-gradient models. In Proceedings of the learning to rank challenge (pp. 25–35). PMLR.

Burges, C. J., Ragno, R., & Le, Q. V. (2007). Learning to rank with nonsmooth cost functions. In Advances in neural information processing systems (pp. 193–200).

Busa-Fekete, R., Hüllermeier, E., & Szörényi, B. (2014). Preference-based rank elicitation using statistical models: the case of mallows. In International conference on machine learning (pp. 1071–1079). PMLR.

Cakir, F., He, K., Xia, X., Kulis, B., & Sclaroff, S. (2019). Deep metric learning to rank. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 1861–1870).

Calders, T., & Jaroszewicz, S. (2007). Efficient AUC optimization for classification. In European conference on principles of data mining and knowledge discovery (pp. 42–53). Springer.

Canu, S., Grandvalet, Y., Guigue, V., & Rakotomamonjy, A. (2005). SVM and kernel methods Matlab toolbox. Perception systemes et information, INSA de Rouen, Rouen, France.

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

Chen, J., Chu, W., Kou, Z., & Zheng, Z. (2010). Learning to blend by relevance. arXiv preprint arXiv:1001.4597.

Chen, K., Li, R., Dou, Y., Liang, Z., & Lv, Q. (2017). Ranking support vector machine with kernel approximation. Computational Intelligence and Neuroscience, 2017, 4629534.

Chen, W., Liu, T.-Y., Lan, Y., Ma, Z.-M., & Li, H. (2009). Ranking measures and loss functions in learning to rank. Advances in Neural Information Processing Systems, 22, 315–323.

Cheng, W., Hüllermeier, E., Waegeman, W., & Welker, V. (2012). Label ranking with partial abstention based on thresholded probabilistic models. Advances in Neural Information Processing Systems, 25, 2501–2509.

Clémençon, S., Lugosi, G., & Vayatis, N. (2008). Ranking and empirical minimization of U-statistics. The Annals of Statistics, 36(2), 844–874.

Clémençon, S., & Achab, M. (2017). Ranking data with continuous labels through oriented recursive partitions. In Advances in neural information processing systems (pp. 4603–4611).

Clémençon, S., & Robbiano, S. (2015a). An ensemble learning technique for multipartite ranking. In Proceedings (pp. 397–402). Presses universitaires de Louvain.

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

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

Clémençon, S., & Vayatis, N. (2008). Tree-structured ranking rules and approximation of the optimal ROC curve. In Proceedings of the 2008 conference on algorithmic learning theory. Lecture Notes Art. International (vol 5254, pp. 22–37).

Clémençon, S., & Vayatis, N. (2009). On partitioning rules for bipartite ranking. In Proceedings of the twelfth international conference on artificial intelligence and statistics (Vol. 5, pp. 97–104). PMLR.

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

Clémençon, S., Lugosi, G., & Vayatis, N. (2005). Ranking and scoring using empirical risk minimization. In International conference on computational learning theory (pp. 1–15). Springer.

Clémençon, S., Depecker, M., & Vayatis, N. (2009). Bagging ranking trees. In 2009 International conference on machine learning and applications (pp. 658–663). IEEE.

Clémençon, S., Depecker, M., & Vayatis, N. (2011). Adaptive partitioning schemes for bipartite ranking. Machine Learning, 83(1), 31–69.
Clémençon, S., Depecker, M., & Vayatis, N. (2013a). Ranking forests. *Journal of Machine Learning Research, 14*(Jan), 39–73.

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

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

Cohen, W. W., Schapire, R. E., & Singer, Y. (1999). Learning to order things. *Journal of Artificial Intelligence Research, 10*, 243–270.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. Cambridge: MIT Press.

Cortes, C., & Mohri, M. (2004). AUC optimization vs. error rate minimization. In *Advances in neural information processing systems* (pp. 313–320).

Cortes, C., Mohri, M., & Rastogi, A. (2007a). Magnitude-preserving ranking algorithms. In *Proceedings of the 24th international conference on machine learning* (pp. 169–176).

Cortes, C., Mohri, M., & Rastogi, A. (2007b). An alternative ranking problem for search engines. In *International workshop on experimental and efficient algorithms* (pp. 1–22).

Cossock, D., & Zhang, T. (2006). Subset ranking using regression. In *International conference on computational learning theory* (pp. 605–619).

Crammer, K., & Singer, Y. (2001). Pranking with ranking. In *Advances in neural information processing systems 14*. Citeseer.

Dang, V. (2013). Ranklib—a library of ranking algorithms.

Davenport, A., & Lovell, D. (2005). Ranking pilots in aerobatic flight competitions. Technical report, IBM Research Report RC23631 (W0506-079), TJ Watson.

Dembczynski, K., Kotlowski, W., & Hüllermeier, E. (2012). Consistent multilabel ranking through univariate losses. arXiv preprint arXiv:1206.6401.

Deng, K., Han, S., Li, K. J., & Liu, J. S. (2014). Bayesian aggregation of order-based rank data. *Journal of the American Statistical Association, 109*(507), 1023–1039.

Dickerson, A., & Popli, G. K. (2016). 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.

Dwork, C., Kumar, R., Naor, M., & Sivakumar, D. (2001). Rank aggregation methods for the web. In *Proceedings of the 10th international conference on world wide web* (pp. 613–622).

Engilberge, M., Chevallier, L., Pérez, P., & Cord, M. (2019). Sodeep: a sorting deep net to learn ranking loss surrogates. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 10792–10801).

Fahandar, M. A., & Hüllermeier, E. (2017). Learning to rank based on analogue reasoning. arXiv preprint arXiv:1711.10207.

Filzmoser, P., Fritz, H., & Kalcher, K. (2018). *pcaperp: Robust PCA by Projection Pursuit*. https://CRAN.R-project.org/package=pcaperp. R package version 1.9-73.

Frank, E., & Hall, M. (2001). A simple approach to ordinal classification. In *European conference on machine learning* (pp. 145–156). Springer.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences, 55*(1), 119–139.

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

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics, pp. 1189–1232.

Fürnkranz, J. (2002). Round robin classification. *Journal of Machine Learning Research, 2*(Mar), 721–747.

Fürnkranz, J., & Hüllermeier, E. (2010). Preference learning and ranking by pairwise comparison. In J. Fürnkranz & E. Hüllermeier (Eds.), *Preference learning* (pp. 65–82). Springer.

Fürnkranz, J., Hüllermeier, E., Mencía, E. L., & Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning, 73*(2), 133–153.

Fürnkranz, J., Hüllermeier, E., & Vanderlooo, S. (2009). Binary decomposition methods for multipartite ranking. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 359–374). Springer.

Fürnkranz, J., & Hüllermeier, E. (01 2011). *Preference Learning*, volume 19. ISBN 978-3-642-14124-9. https://doi.org/10.1007/978-3-642-14125-6.
Khanna, V. K. (2008). Risk-based internal audit in Indian banks: A modified and improved approach for conduct of branch audit. *ICFAI Journal of Audit Practice*, 5(4), 35–56.

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

Köppel, M., Segner, A., Wagener, M., Pensel, L., Karwath, A., & Kramer, S. (2019). Pairwise learning to rank by neural networks revisited: Reconstruction, theoretical analysis and practical performance. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 237–252). Springer.

Korba, A., Clémençon, S., & Sibony, E. (2017). A learning theory of ranking aggregation. In *Proceedings of the 20th conference on artificial intelligence and statistics* (AISTATS 2017).

Kotlowski, W., Dembczynski, K., & Hüllermeier, E. (2011). Bipartite ranking through minimization of univariate loss. In *International conference on machine learning*.

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

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

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

Langville, A. N., & Meyer, C. D. (2012). *Who's # 1?: the science of rating and ranking*. Princeton University Press.

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

Lee, C.-P., & Lin, C.-J. (2014). Large-scale linear RankSVM. *Neural computation*, 26(4), 781–817.

Li, H. (2011a). A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10), 1854–1862.

Li, H. (2011b). Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 4(1), 1–113.

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

Liu, T.-Y. (2011). *Learning to rank for information retrieval*. Berlin: Springer.

Luce, R. D. (1959). *Individual choice behavior*. New York: Wiley.

Lutz, R. W., Kalisch, M., & Bühlmann, P. (2008). Robustified $L_2$-boosting. *Computational Statistics & Data Analysis*, 52(7), 3331–3341.

Ma, L., Xu, L., Zhang, Y., Yan, Y., & Nga, K. N. (2016). No-reference retargeted image quality assessment based on pairwise rank learning. *IEEE Transactions on Multimedia*, 18(11), 2228–2237.

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

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

Mohan, A., Chen, Z., & Weinberger, K. (2011). Web-search ranking with initialized gradient boosted regression trees. In *Proceedings of the learning to rank challenge* (pp. 77–89). PMLR.

Moon, T., Smola, A., Chang, Y., & Zheng, Z. (2010). Intervalrank: isotonic regression with listwise and pairwise constraints. In *Proceedings of the 3rd ACM international conference on web search and data mining* (pp. 151–160).

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

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

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

Pahikkala, T., Tsivtsivadze, E., Airola, A., Järvinen, J., & Boberg, J. (2009). An efficient algorithm for learning to rank from preference graphs. *Machine Learning*, 75(1), 129–165.

Pahikkala, T., Airola, A., Naula, P., & Salakoski, T. (2010). Greedy RankRLS: a linear time algorithm for learning sparse ranking models. In *SIGIR 2010 workshop on feature generation and selection for information retrieval* (pp. 11–18). ACM.
Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., & Cheng, X. (2017). Deeprank: A new deep architecture for relevance ranking in information retrieval. In Proceedings of the 2017 ACM on conference on information and knowledge management (pp. 257–266). ACM.

Patel, T., Telesca, D., Rallo, R., George, S., Xia, T., & Nel, A. E. (2013). Hierarchical rank aggregation with applications to nanotoxicology. Journal of Agricultural, Biological, and Environmental Statistics, 18(2), 159–177.

Pei, C., Zhang, Y., Zhang, Y., Sun, F., Lin, X., Sun, H., et al. (2019). Personalized re-ranking for recommendation. In Proceedings of the 13th ACM conference on recommender systems (pp. 3–11).

Pickett, K. S. (2006). Audit planning: a risk-based approach. New York: Wiley.

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

Qin, T., Zhang, X.-D., Wang, D.-S., Liu, T.-Y., Lai, W., & Li, H. (2007). Ranking with multiple hyperplanes. In Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval (pp. 279–286).

Qin, T., Liu, T.-Y., Zhang, X.-D., Wang, D.-S., Xiong, W.-Y., & Li, H. (2008a). Learning to rank relational objects and its application to web search. In Proceedings of the 17th international conference on world wide web (pp. 407–416).

Qin, T., Zhang, X.-D., Tsai, M.-F., Wang, D.-S., Liu, T.-Y., & Li, H. (2008b). Query-level loss functions for information retrieval. Information Processing & Management, 44(2), 838–855.

Qin, T., Geng, X., & Liu, T.-Y. (2010). A new probabilistic model for rank aggregation. In Advances in neural information processing systems (pp. 1948–1956).

Rajaram, S., Garg, A., Zhou, X. S., & Huang, T. S. (2003). Classification approach towards ranking and sorting problems. In European conference on machine learning (pp. 301–312). Springer.

Rakotomamonjy, A. (2004). Optimizing area under Roc curve with SVMs. In Proceedings of the ECAI-2004 workshop on ROC analysis in AI (pp. 71–80).

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

Rennie, J. D. (2005). Smooth hinge classification. Proceeding of Massachusetts Institute of Technology.

Robbiiano, S. (2013). Méthodes d’apprentissage statistique pour le ranking théorie, algorithmes et applications. PhD thesis, Télécom ParisTech.

Rudin, C. (2009). 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.

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

Sapir, M. (2011). Bipartite ranking algorithm for classification and survival analysis. arXiv preprint arXiv:1112.1966.

Schölkopf, B., Herbrich, R., & Smola, A. (2001). A generalized representer theorem. In Computational learning theory (pp. 416–426). Springer.

Sculley, D. (2009). Large scale learning to rank. In NIPS 2009 workshop on advances in ranking.

Sculley, D. (2010). Combined regression and ranking. In Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 979–988).

Severyn, A., & Moschitti, A. (2015). Learning to rank short text pairs with convolutional deep neural networks. In Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval (pp. 373–382).

Song, Y., Schwing, A., & Urtasun, R. (2016). Training deep neural networks via direct loss minimization. In International conference on machine learning (pp. 2169–2177).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929–1958.

Szörényi, B., Busa-Fekete, R., Paul, A., & Hüllermeier, E. (2015). Online rank elicitation for plank-ett-luce: A dueling bandits approach. In Advances in neural information processing systems (pp. 604–6120).

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

Tsai, T., Yang, K., Ho, T.-Y., & Jin, Y. (2020). Robust adversarial objects against deep learning models. In Proceedings of the AAAI Conference on Artificial Intelligence, 34, 954–962.

Vapnik, V. (1998). Statistical learning theory (Vol. 1). New York: Wiley.

Vembu, S., & Gärtner, T. (2010). Label ranking algorithms: a survey. In J. Fünnkranz & E. Hüllermeier (Eds.), Preference learning (pp. 45–64). Springer.

Waegeman, W., De Baets, B., & Boullart, L. (2008). ROC analysis in ordinal regression learning. Pattern Recognition Letters, 29(1), 1–9.
Wang, X., Li, C., Golbandi, N., Bendersky, M., & Najork, M. (2018). The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM international conference on information and knowledge management* (pp. 1313–1322).

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

Werner, T. (2021). Global quantitative robustness of instance ranking problems. *Available on arXiv*, arXiv:2103.07198

Wu, Q., Burges, C. J., Svore, K. M., & Gao, J. (2008). Ranking, boosting, and model adaptation. Technical report, Microsoft Research.

Wu, Q., Li, H., Wang, Z., Meng, F., Luo, B., Li, W., & Ngan, K. N. (2017). Blind image quality assessment based on rank-order regularized regression. *IEEE Transactions on Multimedia*, 19(11), 2490–2504.

Xia, F., Liu, T.-Y., Wang, J., Zhang, W., & Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on machine learning* (pp. 1192–1199).

Xiong, C., Dai, Z., Callan, J., Liu, Z., & Power, R. (2017). End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval* (pp. 55–64).

Xu, J., Liu, T.-Y., Lu, M., Li, H., & Ma, W.-Y. (2008). Directly optimizing evaluation measures in learning to rank. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 107–114).

Xu, L., Li, J., Lin, W., Zhang, Y., Ma, L., Fang, Y., & Yan, Y. (2016a). Multi-task rank learning for image quality assessment. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(9), 1833–1843.

Xu, L., Li, J., Lin, W., Zhang, Y., Zhang, Y., & Yan, Y. (2016b). Pairwise comparison and rank learning for image quality assessment. *Displays*, 44, 21–26.

Yeh, J.-Y., Lin, J.-Y., Ke, H.-R., & Yang, W.-P. (2007). Learning to rank for information retrieval using genetic programming. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval (LR4IR 2007)*.

Zhai, G., & Min, X. (2020). Perceptual image quality assessment: a survey. *Science China Information Sciences*, 63, 1–52.

Zhang, Y., Wang, D., & Zhang, Y. (2019). Neural IR meets graph embedding: A ranking model for product search. In *The World Wide Web conference* (pp. 2390–2400).

Zheng, Z., Chen, K., Sun, G., & Zha, H. (2007a). A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 287–294).

Zheng, Z., Zha, H., & Sun, G. (2008a). Query-level learning to rank using isotonic regression. In *2008 46th annual allerton conference on communication, control, and computing* (pp. 1108–1115). IEEE.

Zheng, Z., Zha, H., Zhang, T., Chapelle, O., Chen, K., & Sun, G. (2007b). A general boosting method and its application to learning ranking functions for web search. In *NIPS’07: Proceedings of the 20th international conference on neural information processing systems, December 2007* (pp. 1697–1704).

Zhou, K., Xue, G.-R., Zha, H., & Yu, Y. (2008). Learning to rank with ties. In *Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval* (pp. 275–282).

Zhu, X., & Klabjan, D. (2020). Listwise learning to rank by exploring unique ratings. In *Proceedings of the 13th international conference on web search and data mining* (pp. 798–806).

**Publisher’s Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.