Ranking with Ordered Weighted Pairwise Classification

Nicolas Usunier NICOLAS.USUNIER@LIP6.FR
David Buffoni DAVID.BUFFONI@LIP6.FR
Patrick Gallinari PATRICK.GALLINARI@LIP6.FR
Laboratoire d'Informatique de Paris 6, 104, avenue du Président Kennedy, 75016 Paris, France

Abstract

In ranking with the pairwise classification approach, the loss associated to a predicted ranked list is the mean of the pairwise classification losses. This loss is inadequate for tasks like information retrieval where we prefer ranked lists with high precision on the top of the list. We propose to optimize a larger class of loss functions for ranking, based on an ordered weighted average (OWA) (Yager, 1988) of the classification losses. Convex OWA aggregation operators range from the max to the mean depending on their weights, and can be used to focus on the top ranked elements as they give more weight to the largest losses. When aggregating hinge losses, the optimization problem is similar to the SVM for interdependent output spaces. Moreover, we show that OWA aggregates of margin-based classification losses have good generalization properties. Experiments on the Letor 3.0 benchmark dataset for information retrieval validate our approach.

1. Introduction

We address the problem of learning scoring functions, motivated by ranking tasks in Information Retrieval (IR) (but not limited to them). Considering the example of document retrieval, when given a query, the scoring function associates a scalar to each document (the score). The documents are then presented to the user, in decreasing order of scores. The quality of this sorted list of documents depends on the position (the *rank*) of the documents that are relevant to the query; and since the user considers only the few first documents, it is desirable to have a high precision on top scored documents. Learning to rank is the problem of choosing an effective scoring function using a training set of queries for which relevant documents are known.

These last few years, there has been a large amount of work on learning to rank, especially on the problem of obtaining a high precision on the top of the list. Existing approaches fall into three categories. In the first category, the methods define smooth approximations of quality measures used in IR, in a regression framework (Cossock & Zhang, 2006) or using probability distributions on predicted ranks (Taylor et al., 2008). In the second category, the methods use convex upper bounds on these metrics, using boosting (Xu & Li, 2007), or as a large-margin structured output prediction problem (Yue et al., 2007; Xu et al., 2008; Le & Smola, 2007). Finally, the third category of approaches define new multivariate loss functions, easier to optimize, that consider the vector of scores produced for a given query (Burges et al., 2006; Cao et al., 2007). All these methods perform well in practice.

These recent works share the same goal: going beyond the historical ranking framework, the *pairwise classification* approach (see e.g. (Freund et al., 2003; Har-Peled et al., 2002)). In our context, this approach can be explained as follows: the position of a given relevant element in the sorted list can be computed by counting the number of irrelevant elements that have a higher score. This count can be carried out by building the pairs of (relevant, irrelevant) elements, and checking the sign of the difference of scores.

In the context of information retrieval, convex losses for ranking were created by taking the mean of the pairwise classification losses (Joachims, 2002; Freund et al., 2003; Cao et al., 2006). Learning with such a loss function is equivalent to optimizing the mean rank of the relevant documents. In this context, the limit of the pairwise approach is that the mean rank does not provide information on the top of the list. The pairwise approach has also been used in the context of multiclass classification, considering, for each exam-

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

ple, pairs of the form (target class label, other label). The 0/1-loss can then be upper bounded by taking the maximum of the pairwise losses (see e.g. (Crammer & Singer, 2001)). This loss focuses on the top ranked element, but it is still inadequate for IR tasks because it does not provide any rank information when a relevant element is not in the first position. Overall, the limits of the pairwise approach in ranking does not seem to be a consequence of considering each pair individually, but to the small expressivity of the max and mean aggregation operators.

In this paper, we present an extension of the pairwise classification approach. Similarly to the original setting, we start from pairwise losses, and aggregate them. However, instead of using the max or the mean, we use convex Ordered Weighted Averaging (OWA) operators (Yager, 1988). These operators are parameterized by a set of decreasing weights α_j, and make a weighted sum of the classification losses, where the weight α_j is associated to the j-th highest loss. Depending on their weights, convex OWA operators range from the max to the mean operators, thus strictly extending existing pairwise classification approaches. Choosing an OWA operator between these two extreme cases leads to optimizing an error that focuses on the top-ranked elements, as required in IR tasks.

The rest of the paper is organized as follows. We present basic definitions in section 2, and define the ordered weighted pairwise classification (OWPC) approach in section 3. We consider in section 4 a large-margin formulation of OWPC, and show that existing optimization algorithms for structured output predictions can be used for training linear scoring functions. We show in section 5 that the notion of margin can be extended to our framework, yielding margin-based generalization error bounds. We present the experimental evaluation of our algorithm in section 6 on the Letor 3.0[1] benchmark dataset for information retrieval. The related work is discussed in section 7.

2. Framework

2.1. Notations and Definitions

In the rest of the paper, bold characters like $\mathbf{X}$, $\mathbf{x}$ and $\mathbf{y}$ denote sequences. Normal fonts and subscript indexes like X_j and x_j denote the j-th element of the corresponding sequence ($\mathbf{X}$ and $\mathbf{x}$). When the context is clear, we will consider sequences as sets, and use normal fonts without subscript index to denote a value (as in $x \in \mathbf{x}$). The length of a sequence and the cardinal of a set are denoted with brackets (like in $[\mathbf{y}]$). $\{i..j\}$ denotes the interval of integers $\{i, ..., j\}$. The indicator function is denoted I, $\mathrm{sign}(t) \stackrel{def}{=} 2\,\mathrm{I}(t > 0) - 1$ is the sign function, $[t]_+ \stackrel{def}{=} \max(t, 0)$ is the positive part, $\mathfrak{S}_n$ is the set of permutations of $\{1..n\}$, and $\langle ., . \rangle$ denotes the dot product. $\mathbb{E}_{x\sim\mu} f(x)$ denotes the expectation of some function f for x drawn according to μ, and $\hat{\mathbb{E}}_{x\sim S} f(x) \stackrel{def}{=} \frac{1}{[S]} \sum_{s\in S} f(x)$ denotes the mean of f over the finite set S.

We consider a standard setting for learning to rank. The ranking system receives as input an *observation* z, which defines a sequence of *candidates* (or simply elements) $\mathbf{X}(z) \stackrel{def}{=} \big(\mathbf{X}_1(z), ..., \mathbf{X}_{[z]}(z)\big)$ (where $[z]$ is used as a shortcut for $[\mathbf{X}(z)]$). In document retrieval, z corresponds to the user query, and $\mathbf{X}_j(z)$ to the feature representation of the j-th document. For clarity in the discussion, we assume a binary relevance of the candidates: a sequence $\mathbf{y}$ contains the indexes of the *relevant candidates* ($\bar{\mathbf{y}}$ contains the indexes of the *irrelevant* ones). A score (i.e. real-valued) function f takes as input the feature representation of a document, thus $f\big(X_j(z)\big)$, denoted $f_j(z)$ for simplicity, is the score of the j-th document. The output of the ranking system is the list of the candidates sorted by decreasing scores. Given a training set $S = (z_i, \mathbf{y}_i)_{i=1}^m$ of m examples drawn i.i.d. according to some fixed (but unknown) distribution μ, learning to rank consists in choosing a score function f in some predefined set $\mathcal{F}$ with low *generalization error*, $R(f) \stackrel{def}{=} \mathbb{E}_{(z,\mathbf{y})\sim\mu}\, \mathtt{err}(f, z, \mathbf{y})$, for a given *ranking error function* $\mathtt{err}$.

2.2. Ranking error functions

Given an observation z, its relevant candidates $\mathbf{y}$, and a score function f, the predicted *rank* of a relevant element is defined by:

$$\forall y \in \mathbf{y}, rank_y(f, z, \mathbf{y}) \stackrel{def}{=} \sum_{\bar{y}\in\bar{\mathbf{y}}} \mathrm{I}\left(f_y(z) \leq f_{\bar{y}}(z)\right) \quad (1)$$

This definition of rank[2] has lead to the *pairwise classification* approach (see e.g. (Har-Peled et al., 2002; Freund et al., 2003)): $\mathrm{I}\left(f_y(z) \leq f_{\bar{y}}(z)\right)$ is equal to $\mathrm{I}\left(\mathrm{sign}\big(f_y(z) - f_{\bar{y}}(z)\big) \neq 1\right)$. It is thus also the error of the *pairwise classifier* $(y, \bar{y}) \mapsto \mathrm{sign}\big(f_y(z) - f_{\bar{y}}(z)\big)$, which should predict the class label 1. Given our definition of rank, we will consider the following general form of ranking error functions:

Definition 1 *The* ranking error *of a real valued func-*

[1] http://research.microsoft.com/en-us/um/beijing/projects/letor/index.html

[2] We can notice that this definition of rank ignores the relative position of relevant elements. It however has little importance in practice, an is the common definition in the pairwise classification approach.

tion f on $(z, \mathbf{y})$ is defined as follows:

$$\texttt{err}(f, z, \mathbf{y}) \stackrel{def}{=} \frac{1}{[\mathbf{y}]} \sum_{y \in \mathbf{y}} \Phi_{[\bar{\mathbf{y}}]}\left(rank_y(f, z, \mathbf{y})\right)$$

Where, for all $n \in \mathbb{N}^*$, there exists a vector $\boldsymbol{\alpha}^n \in [0,1]^n$ of positive, non-increasing values $\alpha_1^n \geq \alpha_2^n \geq ... \geq \alpha_n^n \geq 0$ with $\sum_{j=1}^n \alpha_j^n = 1$ such that:

$$\forall k \in \{0..n\}\, \Phi_n(k) \stackrel{def}{=} \sum_{j=1}^{k} \alpha_j^n \tag{2}$$

The function Φ_n defines the penalty incurred by placing k irrelevant elements before a relevant one. It is a non-decreasing functions (as a cumulated sum of positive values), and the penalty is 0 when the relevant element is before all irrelevant ones. It takes as special case the $O/1$-loss of multiclass classification (set $\alpha_1 = 1$ and all other αs to 0), as well as the mean pairwise error (set $\alpha_j^n = 1/n$ for all j) usually considered in the pairwise approach for learning to rank. More generally, Φ_n can define errors that focus more or less on the top of the list: the penalty incurred by losing a rank $\Phi_n(k+1) - \Phi_n(k) = \alpha_{k+1}^n$ decreases with k. Thus, setting the first weights to high values means that losing a rank costs a lot at the top of the list, while losing a rank at the bottom costs less.

3. Ordered Weighted Classification

In this section, we present our framework of Ordered Weighted Pairwise Classification, and make the link between the aggregation of pairwise losses and the ranking errors of definition 1. We start with some background on Ordered Weighted Averaging Operators (Yager, 1988).

3.1. Ordered Weighted Averaging

Definition 2 (OWA operator (Yager, 1988)) *Let $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)$ be a sequence of n non-negative numbers with $\sum_{j=1}^n \alpha_j = 1$. The* Ordered Weighted Averaging (OWA) Operator *associated to $\boldsymbol{\alpha}$, is the function* $\text{owa}^{\boldsymbol{\alpha}} : \mathbb{R}^n \to \mathbb{R}$ *defined as follows:*

$$\forall \mathbf{t} = (t_1, ..., t_n) \in \mathbb{R}^n, \text{owa}^{\boldsymbol{\alpha}}(\mathbf{t}) = \sum_{j=1}^{n} \alpha_j t_{\sigma(j)}$$

where $\sigma \in \mathfrak{S}_n$ such that $\forall j, t_{\sigma(j)} \geq t_{\sigma(j+1)}$.

In other words, an OWA operator makes a weighted sum of the t_js. The weight given to t_j does not depend on j, but on the position of t_j in the list $t_{\sigma(1)} \geq t_{\sigma(2)} \geq ... \geq t_{\sigma(n)}$. Thus the max and the mean are special cases of OWA operators: for the max, we set $\alpha_1 = 1$ and $\alpha_j = 0$ for $j > 1$; for the mean, we set $\alpha_j = \frac{1}{n}$ for all j. In fact, both the max and the mean operators are OWA with *non-increasing weights* which, in general, have the following properties:

Proposition 1 *Let $\boldsymbol{\alpha}$ such that $\sum_{j=1}^n \alpha_j = 1$ and $\alpha_1 \geq \alpha_2 \geq ... \geq \alpha_n \geq 0$. Let $\mathbf{t} \in \mathbb{R}^n$. Then:*

1. $\text{owa}^{\boldsymbol{\alpha}}(\mathbf{t}) = \max_{\sigma \in \mathfrak{S}_n} \sum_{j=1}^{n} \alpha_j t_{\sigma(j)} = \max_{\sigma \in \mathfrak{S}_n} \sum_{j=1}^{n} \alpha_{\sigma(j)} t_j$
2. $\mathbf{t} \to \text{owa}^{\boldsymbol{\alpha}}(\mathbf{t})$ *is* convex
3. *If $\mathbf{t}'$ is such that $\forall j, t_j \geq t'_j$, then* $\text{owa}^{\boldsymbol{\alpha}}(\mathbf{t}) \geq \text{owa}^{\boldsymbol{\alpha}}(\mathbf{t}')$

The first point is due to the well-known rearrangement inequality, and directly implies the two other points. We can now make the link between aggregation of pairwise classification losses and the ranking errors. From now on, when the context is clear, we drop the upperscript $\boldsymbol{\alpha}$, and denote $\text{owa}^{\boldsymbol{\alpha}}(\mathbf{t})$ by $\underset{j \in \{1..n\}}{\text{owa}}\, t_j$.

Proposition 2 *Let Φ_n as in equation (2), and let $\boldsymbol{\alpha} = (\alpha_j)_{j=1}^n$ be its associated non-increasing sequence of weights. Let* owa *be the OWA operator with weights $\boldsymbol{\alpha}$. Then, for all $\mathbf{t} \in \mathbb{R}^n$ and any convex function $\ell : \mathbb{R} \to \mathbb{R}_+$ such that $\mathrm{I}(t \leq 0) \leq \ell(t)$, we have:*

1. $\underset{j \in \{1..n\}}{\text{owa}}\, \mathrm{I}(t_j \leq 0) = \Phi_n\Big(\sum_{j=1}^{n} \mathrm{I}(t_j \leq 0)\Big)$
2. $\mathbf{t} \mapsto \underset{j \in \{1..n\}}{\text{owa}}\, \ell(t_j)$ *is convex*
3. $\Phi_n\Big(\sum_{j=1}^{n} \mathrm{I}(t_j \leq 0)\Big) \leq \underset{j \in \{1..n\}}{\text{owa}}\, \ell(t_j)$

Proof The first point is the definition of Φ_n. Points 2 and 3 are direct consequences of proposition 1. □

3.2. Application to learning to rank

Proposition 2 suggests the use of the following convex upper bound on the empirical risk of any ranking error function that satisfies equation (2):

$$\begin{aligned} \hat{R}^{\Phi}(f, S) &\stackrel{def}{=} \hat{\mathbb{E}}_{(z,\mathbf{y}) \sim S} \frac{1}{[\mathbf{y}]} \sum_{y \in \mathbf{y}} \Phi_{[\bar{\mathbf{y}}]}\big(rank_y(f, z, \mathbf{y})\big) \\ &\leq \hat{\mathbb{E}}_{(z,\mathbf{y}) \sim S} \frac{1}{[\mathbf{y}]} \sum_{y \in \mathbf{y}} \underset{\bar{y} \in \bar{\mathbf{y}}}{\text{owa}}\, \ell\big(f_y(z) - f_{\bar{y}}(z)\big) \end{aligned} \tag{3}$$

where ℓ is a non-increasing convex upper bound on $\mathrm{I}(t \leq 0)$, $S = \big((z_1, \mathbf{y}_1), ..., (z_m, \mathbf{y}_m)\big)$ is the training

set, and owa denotes the OWA operator associated to the weights of the corresponding function $\Phi_{[\bar{\mathbf{y}}]}$[3].

We can immediately notice some interesting special cases when ℓ is the hinge loss $t \mapsto [1-t]_+$: if the owa operator is the max (i.e. $\alpha_1 = 1$ in the definition of Φ), the convex upper bound is exactly the one used in SVMs for multiclass classification (Crammer & Singer, 2001; Bordes et al., 2007). If the owa operator is the mean, we recover the standard pairwise setting (see e.g. (Joachims, 2002)).

In general, the convex loss associates the largest weights to the largest losses, and thus to the pairs containing the irrelevant elements with the greater scores (ℓ is non increasing). Consequently, if the weights α are strictly decreasing, losing ranks at the beginning of the list cost a lot (each lost rank is associated to a large weight), while losing a rank on the bottom of the list costs less, and the weights of the OWA operator control the weight given to each part of the list.

4. Large-margin formulation

SVM formulation We now show that a regularized version of the empirical risk of equation (3) using the hinge loss $\ell : t \mapsto [1-t]_+$ can be solved using existing algorithms for structured output spaces (Tsochantaridis et al., 2005). We first consider the following optimization problem, which generalizes SVMs for multiclass classification and pairwise ranking:

$$\min_w \frac{1}{2}||w||^2 + C\sum_{(z,\mathbf{y})\in S} \frac{1}{[\mathbf{y}]} \sum_{y\in\mathbf{y}} \operatorname*{owa}_{\bar{y}\in\bar{\mathbf{y}}} [1 - \langle w, X_y(z) - X_{\bar{y}}(z)\rangle]_+$$

To recover an SVM for interdependent output spaces, we first notice that $[1-t]_+ = \max_{b\in\{0,1\}} b(1-t)$. Thus, using proposition 1, we have:

$$\begin{aligned} &\frac{1}{[\mathbf{y}]} \sum_{y\in\mathbf{y}} \operatorname*{owa}_{\bar{y}\in\bar{\mathbf{y}}} \ell\big(\langle w, X_y(z) - X_{\bar{y}}(z)\rangle\big) \qquad (4)\\ &= \max_{\sigma,\mathbf{b}} \frac{1}{[\mathbf{y}]} \sum_{i=1}^{[\mathbf{y}]} \sum_{j=1}^{[\bar{\mathbf{y}}]} \alpha_j b_{i\sigma(j)} \big(1 - \Big\langle w, X_{y_i}(z) - X_{\bar{y}_{\sigma(j)}}(z)\Big\rangle\big)\\ &= \max_{\sigma,\mathbf{b}} \big\{\Delta_{(z,\mathbf{y})}(\sigma,\mathbf{b}) - \big\langle w, \Xi_{(z,\mathbf{y})}(\sigma,\mathbf{b})\big\rangle\big\} \end{aligned}$$

Where the max is taken over $\sigma \in \mathfrak{S}_{[\bar{\mathbf{y}}]}$, $\mathbf{b} = (b_{ij})_{i\le[\mathbf{y}],j\le[\bar{\mathbf{y}}]} \in \{0,1\}^{[\mathbf{y}]\times[\bar{\mathbf{y}}]}$, and:

$$\Delta_{(z,\mathbf{y})}(\sigma,\mathbf{b}) \stackrel{def}{=} \frac{1}{[\mathbf{y}]} \sum_{i=1}^{[\mathbf{y}]} \sum_{j=1}^{[\bar{\mathbf{y}}]} \alpha_j b_{i\sigma(j)} \qquad (5)$$

$$\Xi_{(z,\mathbf{y})}(\sigma,\mathbf{b}) \stackrel{def}{=} \frac{1}{[\mathbf{y}]} \sum_{i=1}^{[\mathbf{y}]} \sum_{j=1}^{[\bar{\mathbf{y}}]} \alpha_j b_{i\sigma(j)} \Big(X_{y_i}(z) - X_{\bar{y}_{\sigma(j)}}(z)\Big) \qquad (6)$$

[3]In order to simplify the notations, we deleted the upperscript $\boldsymbol{\alpha}$ in the owa operator, but the weights used for a given example $(z,\mathbf{y}) \in S$ depend on $[\bar{\mathbf{y}}]$.

We finally obtain:

$$\begin{aligned} \min_{w,\boldsymbol{\xi}} \quad & \frac{1}{2}||w||^2 + C \sum_{(z,\mathbf{y})\in S} \xi_{(z,\mathbf{y})}\\ \text{u.c.} \quad & \forall (z,\mathbf{y}) \in S, \xi_{(z,\mathbf{y})} \ge 0 \qquad (7)\\ & \forall (z,\mathbf{y}) \in S, \forall \sigma \in \mathfrak{S}_{[\bar{\mathbf{y}}]}, \forall \mathbf{b} \in \{0,1\}^{[\mathbf{y}]\times[\bar{\mathbf{y}}]} :\\ & \quad \big\langle w, \Xi_{(z,\mathbf{y})}(\sigma,\mathbf{b})\big\rangle \ge \Delta_{(z,\mathbf{y})}(\sigma,\mathbf{b}) - \xi_{(z,\mathbf{y})} \end{aligned}$$

Optimization Problem 7 is formally equivalent to an SVM for structured outputs with margin rescaling. Existing optimization algorithms like the cutting plane of (Tsochantaridis et al., 2005) or LaRank (Bordes et al., 2007; Bordes et al., 2008) can thus be applied. We used LaRank, since it is simple to implement and efficient. Due to lack of space, we omit the details of the algorithm. The interested reader can refer to the original LaRank papers for more details.

Computational cost LaRank optimizes the dual objective of problems similar to (7). Similarly to most structured output prediction problems, equation (7) leads to an intractable number of dual variables (at most $[\bar{\mathbf{y}}]! * 2^{[\mathbf{y}]*[\bar{\mathbf{y}}]}$ for each example $(z,\mathbf{y})$). The efficiency of LaRank is however guaranteed by maintaining the vector of dual variable extremely sparse: the authors showed that a (κ,τ)-approximate solution of the dual can be obtained with $O(\frac{Cm}{\kappa\tau})$ non-zero dual variables (Bordes et al., 2007) (this bound is independent on the true number of dual variables). Sparsity is achieved using a so-called arg max procedure that selects the set of non-zero dual variables. In our case, the *argmax* procedure is the search for a permutation σ^* and a binary vector $\mathbf{b}^*$ such that:

$$\sigma^*, \mathbf{b}^* \in \operatorname*{arg\,max}_{\sigma,\mathbf{b}} \big\{\Delta_{(z,\mathbf{y})}(\sigma,\mathbf{b}) - \big\langle w, \Xi_{(z,\mathbf{y})}(\sigma,\mathbf{b})\big\rangle\big\}$$

In practice, only the corresponding values $\Delta_{(z,\mathbf{y})}(\sigma,\mathbf{b})$ and $\Xi_{(z,\mathbf{y})}(\sigma,\mathbf{b})$ need to be computed. In our case, these computations can be done efficiently:

Proposition 3 *Fix an example $(z,\mathbf{y})$, a parameter vector w, and the weights $\boldsymbol{\alpha}$ of the OWA operator. Let:*

$$\begin{aligned} & b^*_{ij} = \mathrm{I}\left(1 + \big\langle w, X_{\bar{y}_j}(z)\big\rangle > \langle w, X_{y_i}(z)\rangle\right)\\ & \sigma^* \ \ s.t.\ \forall j, \big\langle w, X_{\bar{y}_{\sigma^*(j)}}(z)\big\rangle \ge \big\langle w, X_{\bar{y}_{\sigma^*(j+1)}}(z)\big\rangle \end{aligned} \qquad (8)$$

Then:

$$\sigma^*, \mathbf{b}^* \in \operatorname*{arg\,max}_{\sigma,\mathbf{b}} \big\{\Delta_{(z,\mathbf{y})}(\sigma,\mathbf{b}) - \big\langle w, \Xi_{(z,\mathbf{y})}(\sigma,\mathbf{b})\big\rangle\big\}$$

Moreover, the cost of computing $\Delta_{(z,\mathbf{y})}(\sigma^*,\mathbf{b}^*)$ and $\Xi_{(z,\mathbf{y})}(\sigma^*,\mathbf{b}^*)$ is in $O([z]\ln[z] + d[z])$, where d is the dimension of the feature space.

We omit the full proof of the proposition and the algorithm due to lack of space. To give an intuition of the result, we can see that equation (8) gives optimal $\mathbf{b}$ and σ using equation (4): b^*_{ij} is fixed to recover the hinge losses (using $[1-t]_+ = \max_b b(1-t)$), and σ^* orders the obtained losses in decreasing order. For the computational cost, $O([z]\ln[z])$ is due to sorting the elements by decreasing order of scores, and the term $O(d[z])$ is due to the computations of the dot products and the vector $\Xi_{(z,\mathbf{y})}(\sigma,\mathbf{b})$. Overall, the computational cost of the optimization is similar to other learning to rank algorithms like (Yue et al., 2007) and (Le & Smola, 2007).

5. Margin-based error bound

The large margin formulation we consider in problem (7) tries to achieve a constant margin in all pairs. The margin theory in binary and multiclass classification suggests that achieving a large margin on the training set yields good generalization guarantees (see e.g. (Schapire et al., 1998)). In this section, we extend these results to our framework.

To simplify the notations, we consider a simplified setting: $\mathcal{Z}$ is the input space, $\mathcal{H}$ is a set of real-valued measurable functions on $\mathcal{Z}$. μ is a probability measure on the product space $\mathcal{Z}^n$. The training set $S = (\mathbf{z}_1, ..., \mathbf{z}_m)$ contains m observations, assumed to be drawn i.i.d. according to μ (i.e. $\forall i, \mathbf{z}_i \in \mathcal{Z}^n$). Given $\mathbf{z} \in \mathcal{Z}^n$ a score function h makes an error on z_j if $h(z_j) \leq 0$. Given any loss function $\ell : \mathbb{R} \to \mathbb{R}^+$, the generalization error of h is defined by: $\mathbb{E}_{\mathbf{z}\sim\mu}\,\mathrm{owa}_{j\in\{1..n\}}\,\ell\circ h(z_j)$, and the empirical error on the set S is defined by: $\hat{\mathbb{E}}_{\mathbf{z}\sim S}\,\mathrm{owa}_{j\in\{1..n\}}\,\ell\circ h(z_j)$. To make the analogy with our original setting, $\mathbf{z}$ is the sequence of pairs, so z_j is a pair (relevant, irrelevant). $h(z_j)$ is the difference of scores between the relevant and the irrelevant elements of the pair, and μ generates the observations[4].

Our analysis is based on the L_∞ covering numbers (Cucker & Smale, 2002). Given a class of function $\mathcal{H}$ and $\eta > 0$, an η-cover of $\mathcal{H}$ is a finite set $\mathcal{G} \subset \mathcal{F}$ (if any) such that $\forall h \in \mathcal{H}, \exists g \in \mathcal{G}, ||h-g||_\infty \leq \eta$, where $||.||_\infty$ is the infinite norm. The covering number at η, denoted $\mathcal{N}(\mathcal{H},\eta)$ is the cardinal of the smallest η-cover of $\mathcal{F}$. Moreover, our bound is margin-based: considering the loss $c^\gamma(t) = \mathrm{I}\,(t \leq \gamma)$, the generalization error will be measured in terms of c^0 (the true cost), while the empirical error will be measured using a margin penalty $\gamma > 0$:

[4]Formally, the only difference with the original setting is that we have a single relevant element and a constant number of irrelevant elements per observation. Proving an error bound in the original setting is technically similar.

Theorem 4 *Let $\mathcal{H}$ be a class of real-valued functions on $\mathcal{Z}$. Let μ be a probability measure on the product space $\mathcal{Z}^n$. Let S be a training set drawn according to μ^m. Then, for any $\gamma > 0$ and $\delta \in]0,1]$, we have, with probability at least $1-\delta$:*

$$\forall h \in \mathcal{H},\ \mathop{\mathbb{E}}_{\mathbf{z}\sim\mu}\ \mathop{\mathrm{owa}}_{j\in\{1..n\}}\ c\circ h(z_j) \leq \mathop{\hat{\mathbb{E}}}_{\mathbf{z}\sim S}\ \mathop{\mathrm{owa}}_{j\in\{1..n\}}\ c^\gamma \circ h(z_j) + \sqrt{\frac{\ln\left(\mathcal{N}(\mathcal{H},\gamma/2)/\delta\right)}{2m}}$$

When the OWA operator is the max, $\mathrm{owa}_{j\in\{1..n\}}\, c^\gamma \circ h(z_j)$ corresponds to the notion of margin defined for multiclass classification (Schapire et al., 1998). We thus extend the existing results about the margin: since the covering numbers decrease with γ, the theorem shows that, when aggregating losses with an OWA operator, achieving a large margin on the individual pairs will lead to better generalization.

proof sketch Before proving the theorem, we notice that, for any $\mathbf{t}$ and $\mathbf{t'}$ in $\mathbb{R}^n$ (proof omitted):

$$\left[\mathop{\mathrm{owa}}_{j\in\{1..n\}} t_j - \mathop{\mathrm{owa}}_{j\in\{1..n\}} t'j\right]_+ \leq \sum_{j=1}^n \left[t_j - t'_j\right]_+ \qquad (9)$$

For any real-valued function r on $\mathcal{Z}$, we denote $C^\gamma_\mu(r) \stackrel{def}{=} \mathbb{E}_{\mathbf{z}\sim\mu}\,\mathrm{owa}_{j\in\{1..n\}}\, c^\gamma \circ r(z_j)$. We denote $C^\gamma_S(r)$ its empirical version. We first notice that for any $z \in \mathcal{Z}$, any $\theta, \gamma > 0$, and any functions h and g:

$$\Big[\mathrm{I}\,(h(x) \leq \theta) - \mathrm{I}\,(g(x) \leq \theta+\gamma)\Big]_+ \leq \mathrm{I}\,(|h(x)-g(x)| > \gamma) \qquad (10)$$

We now fix some $\gamma > 0$, and consider the quantities $C^0_\mu(h) - C^{\gamma/2}_\mu(g)$ and $C^{\gamma/2}_S(g) - C^\gamma_S(h)$. Applying to each of them sequentially $\mathbb{E}\,t - \mathbb{E}\,t' \leq \mathbb{E}\,[t-t']_+$, equation (9) and equation (10), we obtain:

$$\begin{aligned} C^0_\mu(h) - C^{\gamma/2}_\mu(g) &\leq \sum_{j=1}^n \mathop{\mathbb{P}}_{z\sim\mu_j}\left(|h(z)-g(z)| > \gamma/2\right) \\ C^{\gamma/2}_S(g) - C^\gamma_S(h) &\leq \sum_{j=1}^n \mathop{\mathbb{P}}_{z\sim\hat{\mu}^S_j}\left(|h(z)-g(z)| > \gamma/2\right) \end{aligned} \qquad (11)$$

Where μ_j is the marginal distribution of the j-th coordinate according to μ, and $\hat{\mu}^S_j$ the empirical marginal distribution on S. Let $h \in \mathcal{H}$ and let $\mathcal{G}$ be a $\gamma/2$ -cover of size finite class of size $\mathcal{N}(\mathcal{F},\gamma/2)$. Taking $g \in \mathcal{G}$ such that $||h-g||_\infty \leq \gamma/2$, the two right-hand terms of the inequalities (11) equal 0. We then add the two inequalities and apply the error bound on finite function classes to $C^{\gamma/2}_\mu(g) - C^{\gamma/2}_S(g)$, the result is obtained. □

Table 1. Test performance of OWPC in terms of MAP on Letor for different setting the the weights. $g_{=1}$ represents the constant weights, and $g_{=1}^{p\%}$ the constant weight on the top $p\%$ of the list (all weights equal 0 after the top $p\%$).

	TD03	TD04	HP03	HP04	NP03	NP04
owpc $g_{=1}^{5\%}$	0,278	0,213	**0,752**	**0,690**	**0,683**	**0,689**
owpc $g_{=1}^{10\%}$	**0,298**	0,229	0,743	0,681	**0,683**	0,678
owpc $g_{=1}$	0,244	**0,232**	0,738	0,678	0,676	0,675

Table 2. Test MAP of OWPC and Letor baselines. $g_{/}$ corresponds to linearly decreasing weights, g_{All} to weights chosen on the validation set.

	TD03	TD04	HP03	HP04	NP03	NP04
RSVM	0.263	0.224	0.741	0.668	**0.696**	0.659
SVM^{map}	0.245	0.205	0.742	0.718	0.687	0.662
Adarank	0.228	0.219	**0.771**	0.722	0.678	0.622
ListNet	0.275	0.223	0.766	0.690	0.690	0.672
owpc g_{All}	**0.282**	**0.227**	0.745	**0.726**	0.680	**0.690**
owpc $g_{/}$	**0.290**	**0.229**	0.757	**0.726**	0.685	**0.683**

6. Experiments

In this section, we present our experiments on the Letor 3.0 benchmark datasets[5] (Liu et al., 2007). We report results on the 6 .GOV datasets[6], named TD 2003, TD 2004, HP 2003, HP 2004, NP 2003 and NP 2004. The data was collected from various document retrieval competitions taking place in 2003 and 2004. Each dataset contains between 50 and 150 queries with at most 1000 labeled documents per query. For each query, each document has a vectorial representation of 60 features commonly used in information retrieval. A detailed description of the datasets can be found on Letor's Website. Each dataset contains five folds. Each fold contains a training set (3/5 of the queries), a validation set (1/5) and a test set (1/5). The queries used for training, validating and testing vary from one fold to the other. The evaluation is carried out with two standard metrics in IR: the Mean Average Precision (MAP) and the Normalized Discounted Cumulated Gain (NDCG) (see e.g. (Manning et al., 2008)).

Definition of the weights Using the Ordered Pairwise Classification (OWPC) approach requires the definition of the weights used in the OWA operator. We restricted ourselves to simple weighting schemes of the following form: $\alpha_j = g(j,n)/\sum_{k=1}^{n} g(k,n)$, where $g : \mathbb{N}^{*2} \rightarrow \mathbb{R}^+$ is called a *generator function*, and n is the number of irrelevant documents for the considered query[7]. We considered three kinds of generator functions: **(1)** constant weight on the top $p\%$ of the list, denoted $g_{=1}^{p\%}$: $g_{=1}^{p\%}(j,n) = 1$ if $j/n \leq p\%$ and 0 otherwise. **(2)** linearly decreasing weights: denoted $g_{/}$: $g_{/}(j,n) = 1/j$. **(3)** exponentially decreasing weights: denoted $g_{exp}^{p\%}$: $g_{exp}^{p\%}(j,n) = 2^{-100/p*j/n}$ (The weight is divided by 2 every $p\%$ of the list).

Experimental setup We present two kinds of experiments: **(1)** the weights are fixed before seeing the data, **(2)** the weights are considered as an hyperparameter of the algorithm and are selected on the validation set. In all cases, for a given weighting scheme and for each fold, the hyperparameter C (equation (7)) is chosen among $\{10^{-3}, 10^{-2}, ..., 10^{3}\}$ using the MAP score on the validation set. When the weights are considered as a hyperparameter, the optimal value of C is first chosen for each generator function, and the one achieving the best MAP on the validation set is chosen. In that situation, the following generator functions were used: the linearly decreasing weights ($g_{/}$), the constant weights on the top $p\%$ ($g_{=1}^{p\%}$) and the exponentially decreasing weights ($g_{exp}^{p\%}$) with $p \in \{5, 10, 20\}$, and an additional run $g_{=1} \stackrel{def}{=} g_{=1}^{100}$, corresponding to standard pairwise classification.

Influence of the weights We first present a simple experiment to show the effect of focusing on the top $p\%$ of the list (with a constant weight, i.e. the $g_{=1}^{p\%}$ generator functions). The test performance in terms of MAP on the 6 datasets are reported in table 1. For readability, we only included $p = 5\%$, $p = 10\%$ and the constant weights (the $g_{=1}$ generator function). $g_{=1}$ is our baseline since it corresponds to the standard mean pairwise error. As expected, focusing on the top of the list yields a significant improvement: focusing on the top 10% of the list yields a better MAP than $g_{=1}$ on 5 datasets out of 6, and focusing on the top 5% of the list yields even better results on 4 out of 6 datasets.

Comparison to other algorithms Tables 2, 3 and 4 show the test performance of OWPC in terms of MAP, NDCG@1 and NDCG@3 on the six datasets. Due to space limitations and for better readability, we only report the results of the linearly decreasing weights $g_{/}$, which tend to have the best overall validation scores, and the results of the run g_{All} where the weights are chosen on the validation set.

We also report results of state-of-the-art ranking algorithms: RankingSVM (Joachims, 2002),

[5] http://research.microsoft.com/en-us/um/beijing/projects/letor/index.html

[6] We do not report results for the OHSUMED dataset since we limited ourselves to binary relevance judgements.

[7] Recall that the weights depend on the number of irrelevant documents for the query due to the normalization constraint $\sum_{j=1}^{n} \alpha_j = 1$.

Table 3. Test NDCG@1 of OWPC and Letor baselines.

	TD03	TD04	HP03	HP04	NP03	NP04
RSVM	0.320	0.413	0.693	0.573	**0.580**	0.507
SVM^{map}	0.320	0.293	0.713	**0.627**	0.560	0.520
Adarank	0.360	0.427	0.713	0.587	0.560	0.507
ListNet	0.400	0.360	**0.720**	0.600	0.567	0.533
owpc g_{All}	**0.420**	0.333	**0.727**	**0.627**	0.567	**0.560**
owpc $g_{/}$	**0.440**	**0.453**	**0.720**	0.613	**0.580**	**0.560**

Table 4. Test NDCG@3 of OWPC and Letor baselines.

	TD03	TD04	HP03	HP04	NP03	NP04
RSVM	**0.344**	0.347	0.775	0.715	0.765	0.750
SVM^{map}	0.320	0.304	0.779	0.754	**0.767**	0.749
Adarank	0.291	**0.369**	0.790	0.751	0.716	0.672
ListNet	0.337	0.357	**0.813**	0.721	0.758	**0.759**
owpc g_{All}	0.342	0.349	0.782	**0.808**	0.737	**0.775**
owpc $g_{/}$	**0.361**	**0.371**	0.794	**0.800**	0.731	**0.759**

SVM^{MAP}(Yue et al., 2007), Adarank [8] (Xu & Li, 2007) and ListNet (Cao et al., 2007). The results of these algorithms are publicly available on Letor's Website[9]. In the tables, the results of OWPC are in bold if they are better than the baselines, and the result of the best baseline is in bold if outperforms at least one OWPC.

The run with linearly decreasing weights ($g_{/}$) outperforms the baselines (or give equal results) on 4 out of 6 datasets on the three measures (5/6 for NDCG@1). These results prove the relevancy of the OWPC for ranking, but also that the linearly decreasing weights may be a good default choice for the generator function. The run with weights chosen on the validation set (g_{All}) outperforms the baselines (or give equal results) on 4 out of 6 datasets in terms of MAP and NDCG@1 measures. It gives similar results as ListNet in terms of NDCG@3. This proves, once again, that the OWPC approach gives state-of-the-art results. In comparison to the linearly decreasing weights, the run g_{All} tends to be have a lower MAP and NDCG@1. This is due to some overfitting on the validation sets, since these sets are rather small and the number of compared models for the choice of g_{All} is very high.

7. Related work

The idea of weighting losses has already been used in (Cao et al., 2006) and (Cossock & Zhang, 2006). However, in these works, the weights are associated to individual losses prior to learning, and consequently do not depend on the ranks predicted by the function during training. We may also notice that in our context of binary relevance judgements, the weights used in (Cao et al., 2006) are the special case of our method when constant weights are used (i.e. $\alpha_j = \frac{1}{[\bar{\mathbf{y}}]}$, for all j, for a given query $(z, \mathbf{y})$).

The SVM for structured output has been used in the context of ranking in (Yue et al., 2007; Xu et al., 2008; Le & Smola, 2007). In these works, the margin rescaling function Δ (see equation (7)) is any function, and the feature map they use (Ξ in equation (7)) is developped independently of the rescaling factor. In contrast, in our approach, both the feature map and the rescaling factor are implied by the loss function we defined. They have the advantage of being extremely general (able to optimize a convex upper bound on any quality measure, using a suitable margin rescaling factor). On the other hand, this upper bound can be very loose (Do et al., 2008), and there is little theoretical evidence on how to make the feature map[10].

In contrast to the above works, we proposed to optimize a function which is not directly related to standard evaluation measures used in IR. In that sense, we follow (Burges et al., 2006; Cao et al., 2007), where the authors propose loss functions that mimic the basic behavior of standard evaluation measures, but are easier to handle (training can be efficiently carried out with gradient descent). Since LaRank has proved to be very effective on large scale structured prediction tasks (Bordes et al., 2008), it would also be interesting to investigate its large-scale behavior when optimizing our ranking loss. Unfortunately, the benchmark datasets of Letor are too small to explore this.

In terms of generalization, we obtained an interpretation in terms of the margin theory. Error bounds for ranking in the context of IR have been shown for regularized empirical risk minimization methods for some pairwise approaches (Lan et al., 2008) and importance weighted regression (Cossock & Zhang, 2006), and with Rademacher averages for ListNet (Liu & Lan, 2008). However, to the best of our knowledge, the notion of margin was not recovered for ranking algorithms in the context of IR.

[8] For the AdaRank algorithm, we report the results of AdaRank^{map} in terms of MAP, and AdaRank^{ndcg} when using the NDCG.

[9] The results for other algorithms are available on Letor's Website. Due to space limitations, we chose the 4 algorithms with the best overall performance.

[10] (Le & Smola, 2007) consider ranking as predicting a permutation, and use a weighting scheme similar to OWPC (in fact, the score associated to the predicted permutation is an OWA aggregation of the documents scores). Although the approaches are equivalent in the multi-class classification case, they are not related in general (e.g. our feature map Ξ depends on two discrete components σ and $\mathbf{b}$, while their feature map only depends on a permutation).

8. Conclusion

We presented the ordered weighted pairwise classification approach to ranking. It extends traditional approaches to ranking and multi-class classification, and allows to define convex loss functions that focus on the top of the list. The approach obtains state-of-the-art results on a benchmark dataset for learning to rank.

Acknowlegments

The authors thank Lucie Galand for the discussions on OWA operators. This work was partly funded by the ANR (CADI project), and by the EU PASCAL2 network of excellence.

References

Bordes, A., Bottou, L., Gallinari, P., & Weston, J. (2007). Solving multiclass support vector machines with larank. *Proc. of the Int. Conf. on Mach. Learn.* (pp. 89–96).

Bordes, A., Usunier, N., & Bottou, L. (2008). Sequence labelling svms trained in one pass. *Proc. of the Eur. Conf. on Mach. Learn. and Principles and Practice of Knowledge Discovery in Databases* (pp. 146–161).

Burges, C. J. C., Ragno, R., & Le, Q. V. (2006). Learning to rank with nonsmooth cost functions. *Proc. of Adv. in Neural Inf. Processing Syst.* (pp. 193–200).

Cao, Y., Xu, J., Liu, T.-Y., Li, H., Huang, Y., & Hon, H.-W. (2006). Adapting ranking svm to document retrieval. *Proc. of the 29th SIGIR Conf. on Res. and Devel. in Inf. Ret.* (pp. 186–193).

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. *Proc. of the Int. Conf. on Mach. Learn.* (pp. 129–136).

Cossock, D., & Zhang, T. (2006). Subset ranking using regression. *Proc. of Comp. Learn. Theory* (pp. 605–619).

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *J. of Mach. Learn. Res.*, *2*, 265–292.

Cucker, F., & Smale, S. (2002). On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, *39*, 1–49.

Do, C. B., Le, Q., Chapelle, O., & Smola, A. (2008). Tighter bounds for structured estimation. *Proc. of Adv. in Neural Inf. Processing Syst.* (pp. 281–288).

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *J. of Mach. Learn. Res.*, *4*, 933–969.

Har-Peled, S., Roth, D., & Zimak, D. (2002). Constraint classification for multiclass classification and ranking. *Proc. of Adv. in Neural Inf. Processing Syst.* (pp. 785–792).

Joachims, T. (2002). Optimizing search engines using clickthrough data. *Proc. of the Int. Conf. on Kwoledge Discovery and Data Mining* (pp. 133–142).

Lan, Y., Liu, T.-Y., Qin, T., Ma, Z., & Li, H. (2008). Query-level stability and generalization in learning to rank. *Proc. of the Int. Conf. on Mach. Learn.* (pp. 512–519).

Le, Q. V., & Smola, A. J. (2007). *Direct optimization of ranking measures* (Technical Report). NICTA.

Liu, T.-Y., & Lan, Y. (2008). *Generalization analysis of listwise learning-to-rank algorithms using rademacher average* (Technical Report MSR-TR-2008-155). Microsoft Res.

Liu, T.-Y., Xu, J., Qin, T., Xiong, W., & Li, H. (2007). Letor: Benchmark dataset for research on learning to rank for information retrieval. *Proc. of SIGIR'07 workshop on Learning to Rank for Inf. Ret.*.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval.* Cambridge University Press.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *Annals of Statistics*, *26*, 322–330.

Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008). Softrank: optimizing non-smooth rank metrics. *Proc. of the Int. Conf. on Web Search and Data Mining* (pp. 77–86). ACM.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *J. of Mach. Learn. Res.*, *6*, 1453–1484.

Xu, J., & Li, H. (2007). Adarank: a boosting algorithm for information retrieval. *Proc. of the 30th SIGIR Conf. on Res. and Devel. in Inf. Ret.* (pp. 391–398).

Xu, J., Liu, T.-Y., Lu, M., Li, H., & Ma, W.-Y. (2008). Directly optimizing evaluation measures in learning to rank. *Proc. of the 31st SIGIR Conf. on Res. and Devel. in Inf. Ret.* (pp. 107–114).

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multi-criteria decision making. *IEEE Transactions on Syst., Man and Cybernetics*, *18*, 183–190.

Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A support vector method for optimizing average precision. *Proc. of the 30th SIGIR Conf. on Res. and Devel. in Inf. Ret.* (pp. 271–278).