Generalization Analysis of Listwise Learning-to-Rank Algorithms

Yanyan Lan* LANYANYAN@AMSS.AC.CN
Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, 100190, P.R. China.
Microsoft Research Asia, 4F, Sigma Center, No. 49, Zhichun Road, Beijing, 100190, P. R. China.

Tie-Yan Liu TYLIU@MICROSOFT.COM
Microsoft Research Asia, 4F, Sigma Center, No. 49, Zhichun Road, Beijing, 100190, P. R. China.

Zhiming Ma MAZM@AMT.AC.CN
Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, 100190, P.R. China.

Hang Li HANGLI@MICROSOFT.COM
Microsoft Research Asia, 4F, Sigma Center, No. 49, Zhichun Road, Beijing, 100190, P. R. China.

Abstract

This paper presents a theoretical framework for ranking, and demonstrates how to perform generalization analysis of listwise ranking algorithms using the framework. Many learning-to-rank algorithms have been proposed in recent years. Among them, the listwise approach has shown higher empirical ranking performance when compared to the other approaches. However, there is no theoretical study on the listwise approach as far as we know. In this paper, we propose a theoretical framework for ranking, which can naturally describe various listwise learning-to-rank algorithms. With this framework, we prove a theorem which gives a generalization bound of a listwise ranking algorithm, on the basis of Rademacher Average of the class of compound functions. The compound functions take listwise loss functions as outer functions and ranking models as inner functions. We then compute the Rademacher Averages for existing listwise algorithms of ListMLE, ListNet, and RankCosine. We also discuss the tightness of the bounds in different situations with regard to the list length and transformation function.

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.
*The work was performed when the first author was an intern at Microsoft Research Asia.

1. Introduction

Ranking is an important problem in various applications, such as information retrieval, natural language processing, computational biology, and social sciences. In many real scenarios, the ranking problem is defined as follows. Given a group of objects, a ranking model sorts the objects with respect to each other in the group, according to their degrees of relevance, importance, or preferences. For example, in information retrieval, a "group" corresponds to a query, and "objects" corresponds to documents associated with the query. In recent years, machine learning technologies have been developed to learn ranking models, and a new research branch named "learning to rank" has emerged. In the training phase, several groups of objects are given to facilitate the creation of a ranking model, and in testing, the ranking model is used to predict the ranked list for a new group of objects. Many learning-to-rank algorithms have been proposed in the literature, which can be categorized into three groups: the pointwise, pairwise, and listwise approaches. The pointwise and pairwise approaches (Li et al., 2007; Herbrich et al., 1999; Crammer & Singer, 2001) respectively transform ranking into (ordinal) regression or classification on single object and object pairs. The listwise approach solves the problem of ranking by minimizing a loss function defined on object lists. Representative listwise ranking algorithms include ListMLE (Xia et al., 2008), ListNet (Cao et al., 2007), and RankCosine (Qin et al., 2007). According to previous studies (Cao et al., 2007; Qin et al., 2007; Xia et al., 2008), the listwise approach can outperform the other two approaches on benchmark datasets.

Although some works have discussed the generalization bound for pairwise ranking algorithms, such as (Agarwal & Niyogi, 2005) and (Freund et al., 2003), these studies are not sufficient especially for handling the real settings in applications like information retrieval. Ranking in these real applications employs a special data structure of two layers: group and object (e.g., in information retrieval, query and document). While the ranking model operates at the object level, the evaluation of a ranking model is usually performed at the group level. Due to these key differences from conventional learning tasks, many existing generalization theories in machine learning may not be directly applied.

Without loss of generality, we take information retrieval as an example application in this paper. In such a scenario, a meaningful generalization bound on a learning to rank algoirthm should be defined at query level. As far as we know, (Lan et al., 2008) was the only work that investigates the query-level generalization ability of learning-to-rank algorithms. The problem with their approach, however, was that the proposed framework could only be used in the generalization analysis of pairwise approach, but not the listwise approach[1]. Therefore, how to conduct generalization analysis of listwise ranking algorithms was still an open issue. The goal of this paper is exactly to tackle the challenge.

First, to conduct generalization analysis of listwise ranking algorithms, we extend the query-level ranking framework proposed in (Lan et al., 2008). In our framework, a query is directly assumed to be a random variable, represented as the set of documents associated with it. However, there is no further assumption on stochastic generation of documents from the set, once it is given and fixed. In this way, it can more naturally characterize the listwise ranking algorithms.

Second, with the extended framework, we further employ the Rademacher Average technique (Mendelson, 2001; Bartlett & Mendelson, 2002; Mendelson, 2003; Bousquet et al., 2004) to perform the generalization analysis. Specifically, in this paper, we prove that if we define a compound function, whose outer function is the listwise loss function and inner function is the ranking model of a listwise algorithm, then the generalization bound of the listwise algorithm will be a function of the Rademacher Average of this compound function class. We then demonstrate how to compute the Rademacher Average, by taking the existing algorithms of ListMLE, ListNet, and RankCosine as examples. We also discuss the tightness of the generalization bounds in different situations, in order to provide guidelines on algorithm design and parameter tuning. For instance, we discuss how to set the parameters of an algorithm, how to select a transformation function[2], and how to choose the most suitable algorithm in a given setting.

To our knowledge, this paper is the first study on the generalization ability of listwise ranking algorithms. Major contributions of the paper include:

1) the proposal of the extended query-level ranking framework, which enables theoretical analysis on the listwise approach;

2) the proof of the theorem that gives a generalization bound of listwise ranking algorithm on the basis of the Rademacher Average;

3) the derivation of the Rademacher Averages for three listwise ranking algorithms;

4) the investigations on the tightness of generalization bounds, with respect to different listwise loss functions and transformation functions.

2. Query-Level Ranking Framework

Given the special characteristics of ranking when compared with classification and regression (e.g., the notion of 'query' exists in learning and evaluation), a new framework is needed to facilitate theoretical research on the problem. The framework proposed in (Lan et al., 2008) was based on the same motivation. Unfortunately, it can only explain the pointwise and pairwise approaches, but not the listwise approach including ListNet (Cao et al., 2007) and ListMLE (Xia et al., 2008). In this work, we extend their framework, and provide a more reasonable formalization of the listwise ranking algorithms.

For ease of explanation, let us first look at the query-level ranking framework proposed in (Lan et al., 2008).

Let $\mathcal{Q}$ be the query space, $\mathcal{D}$ be the document space, and $\mathcal{X}(= R^d)$ be the d-dimensional feature space. We use the mapping $\Phi : \mathcal{Q} \times \mathcal{D} \to \mathcal{X}$ to create a feature vector from a query-document pair. Let q be a random variable defined on the query space with an unknown

[1] The authors claimed that their framework can handle the listwise case. However, their "listwise" formulation is more like an extension of the pairwise approach in which m-document subset is utilized, and thus it is not the same as the conventional listwise approach.

[2] In the existing listwise ranking algorithms, a transformation function is first applied to the ranking scores of documents, and then a liswise loss is defined based on the transformed scores and the ground truth.

probability distribution $P_{\mathcal{Q}}$. Let f denote the real-valued ranking function, which assigns each document a score $f(x)$. The scores of the documents associated with the same query are used to rank the documents. We measure the loss of ranking documents for query q using function f with a loss function $L(f;q)$.

The goal of ranking is to minimize the *expected risk* of f as defined below:

$$R(f) = \int_{\mathcal{Q}} L(f;q) P_{\mathcal{Q}}(dq).$$

As the distribution $P_{\mathcal{Q}}$ is unknown, we use the *empirical risk* to approximate the expected risk, which is defined as follows:

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} L(f;q_i),$$

where $q_1, \cdots, q_n$ are i.i.d observations of q.

Different ways of defining the loss function $L(f;q)$ lead to different formalizations. Lan et al.(2008) gives a definition of $L(f;q)$ for the pointwise approach, pairwise approach, and 'listwise' approach. Specifically, $L(f;q)$ is defined as a function based on a probability distribution of documents, document pairs, or m-document subset. We point out that this might not be appropriate, because for the existing listwise algorithms such as ListNet and ListMLE, once a query is given, the set of all the documents associated with the query is fixed. There is no further random sampling of documents from the set and the query-level loss function does not need to be dependent on such a random sampling. Therefore, it would be more accurate to define the loss function $L(f;q)$ as a function of the fixed set of all the associated documents. This is exactly the way in which we define the loss function $L(f;q)$ in this paper.

For each query, the number of all the documents associated is assumed to be m. This is for simplicity and it is exactly the case when we perform re-ranking of the top-ranked m documents in practice. The goal of learning in the listwise approach is to best predict the ranked list of m documents given a query.

We actually represent query q by (z, y), where $z = (x_1, \cdots, x_m)$ and y stands for the ground-truth permutation of m documents. Let $\mathcal{Z} = \mathcal{X}^m$ be the input space, whose elements are m feature vectors corresponding to the m documents, where $y(i)$ stands for the index of the document whose rank is i in the permutation y. We call m the list length and assume that $m \geq 3$. Let $\mathcal{Y}$ be the output space, whose elements are permutations of the m documents. Then we regard (z, y) as a random variable on the space $\mathcal{Z} \times \mathcal{Y}$ according to an unknown probability distribution $P(\cdot, \cdot)$.

Let $l(f; z, y)$ denote a listwise loss function (e.g., the likelihood loss in (Xia et al., 2008), the cross entropy loss in (Cao et al., 2007), and the cosine loss in (Qin et al., 2007)) defined on the random variable (z, y) and the ranking model $f \in \mathcal{F}$, where $\mathcal{F}$ is the real-valued function class. Then in our extended framework, $L(f;q)$ and $P_{\mathcal{Q}}$ can be set as $L(f;q) = l(f;z,y)$ and $P_{\mathcal{Q}} = P(\cdot,\cdot)$ respectively.

In this way, the *expected risk* of the listwise approach with respect to the loss function l is defined as:

$$R_l(f) = \int_{\mathcal{Z}\times\mathcal{Y}} l(f;z,y)\, P(dz, dy).$$

The *empirical risk* of the listwise approach with respect to the loss function l is defined as:

$$\widehat{R}_l(f;S) = \frac{1}{n}\sum_{i=1}^{n} l(f;z_i,y_i),$$

where $(z_i, y_i), i = 1, \cdots, n$ denotes the training data sampled i.i.d. with (z, y) from the space $\mathcal{Z} \times \mathcal{Y}$, $S = \{(z_1, y_1), \cdots, (z_n, y_n)\}$ and $z_i = (x_1^{(i)}, \cdots, x_m^{(i)})$.

With the theoretical framework above, one can perform various theoretical analysis on the listwise approach. One of the most important analyses is about the generalization ability of the listwise ranking algorithms. Mathematically, it is to find a tight upper bound of $\sup_{f\in\mathcal{F}}(R_l(f) - \widehat{R}_l(f;S))$, which is called generalization bound in this paper.

3. Generalization Analysis on Listwise Ranking Algorithms

In this section, we give the generalization bound of a listwise algorithm using the ranking framework and Rademacher Average[3]. To the best of our knowledge, this is the first generalization analysis on the listwise approach to learning to rank.

3.1. Rademacher Average based Generalization Bound

Rademacher Average measures how much the function class $\mathcal{F}$ can fit random noise. The definition of Rademacher Average is as follows.

Definition 1. *For a function class $\mathcal{G}$, the empirical Rademacher Average is defined as:*

$$\widehat{\mathcal{R}}(\mathcal{G}) = E_\sigma \sup_{g\in\mathcal{G}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i g(X_i),$$

[3]There are two types of Rademacher Averages: the empirical and expected Rademacher Averages (Bartlett & Mendelson, 2002). We use the empirical Rademacher Average in this paper.

where $X_i, i = 1, \cdots, n$ are i.i.d. random variables, and $\sigma_i, i = 1, \cdots, n$ are i.i.d. random variables, with probability $\frac{1}{2}$ to take value 1 or -1, σ stands for $\{\sigma_1, \cdots, \sigma_n\}$.

According to (Ledoux & Talagrand, 1991; Bartlett & Mendelson, 2002), there are three properties with Rademacher Average, as summarized below.

1) $\forall c \in \mathbb{R}$, where $\mathbb{R}$ stands for the space of real numbers, $c\mathcal{G} \triangleq \{h : \exists g \in \mathcal{G}, s.t. h = cg.\}$, then,

$$\widehat{\mathcal{R}}(c\mathcal{G}) = |c|\widehat{\mathcal{R}}(\mathcal{G}). \tag{1}$$

2) If λ is a Lipschitz function with Lipschitz constant L_λ (i.e., $\forall x_1, x_2$ within the domain of λ, $|\lambda(x_1) - \lambda(x_2)| \leq L_\lambda |x_1 - x_2|$) and $\lambda(0) = 0$, $\lambda \circ \mathcal{G} \triangleq \{h : \exists g \in \mathcal{G}, s.t. h = \lambda \circ g.\}$, where $\lambda \circ g$ stands for the compound function, whose outer function is λ and inner function is g. Then,

$$\widehat{\mathcal{R}}(\lambda \circ \mathcal{G}) \leq L_\lambda \widehat{\mathcal{R}}(\mathcal{G}). \tag{2}$$

3) Given $\mathcal{G}_i, i = 1, \cdots, n$, $\sum_{i=1}^n \mathcal{G}_i \triangleq \{g : \exists g_i \in \mathcal{G}_i, i = 1, \cdots, n, s.t. g = \sum_{i=1}^n g_i.\}$, then,

$$\widehat{\mathcal{R}}(\sum_{i=1}^n \mathcal{G}_i) \leq \sum_{i=1}^n \widehat{R}(\mathcal{G}_i). \tag{3}$$

Applying the theory of Rademacher Average (Bartlett & Mendelson, 2002)(Bousquet et al., 2004) to the listwise approach, we can get the generalization bound of the approach as shown in the following theorem. For the detailed proof, please refer to (Liu & Lan, 2008).

Theorem 1. *Let $\mathcal{A}$ denote a listwise ranking algorithm, and let $l_{\mathcal{A}}(f; z, y) \in [0, 1]$ be its listwise loss, given the training data $S = \{(z_i, y_i), i = 1, \cdots, n\}$, with probability at least $1 - \delta$, the following inequality holds:*

$$\sup_{f \in \mathcal{F}} (R_{l_{\mathcal{A}}}(f) - \widehat{R}_{l_{\mathcal{A}}}(f; S)) \leq 2\widehat{\mathcal{R}}(l_{\mathcal{A}} \circ \mathcal{F}) + \sqrt{\frac{2 \ln \frac{2}{\delta}}{n}},$$

where $\widehat{\mathcal{R}}(l_{\mathcal{A}} \circ \mathcal{F}) = E_\sigma \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \sigma_i l_{\mathcal{A}}(f; z_i, y_i)$.

Theorem 1 shows that the generalization bound of a listwise ranking algorithm is related to $\widehat{\mathcal{R}}(l_{\mathcal{A}} \circ \mathcal{F})$, i.e., the Rademacher Average of the class of compound functions, whose outer functions are the listwise loss functions and inner functions are the ranking models. In order to apply this theorem to a specific listwise ranking algorithm, we need to compute the Rademacher Average of the corresponding compound function class.

3.2. Rademacher Averages of Listwise Ranking Algorithms

We demonstrate how to compute the Rademacher Averages for three existing listwise ranking algorithms, ListMLE, ListNet and RankCosine.

3.2.1. Listwise Ranking Algorithms

We first show the listwise loss functions of the three algorithms:

$$\textbf{ListMLE}: l(f; z, y) = -\log P(y|z; f),$$
$$P(y|z; f) = \prod_{i=1}^m \frac{\phi(f(x_{y(i)}))}{\sum_{j=i}^m \phi(f(x_{y(j)}))}.$$

$$\textbf{ListNet}: l(f; z, y) = -\sum_{\forall \pi \in \mathcal{Y}} P(\pi|z; g_y) \log P(\pi|z; f),$$
$$P(\pi|z; g_y) = \prod_{i=1}^m \frac{\phi(g_y(x_{\pi(i)}))}{\sum_{j=i}^m \phi(g_y(x_{\pi(j)}))},$$
$$P(\pi|z; f) = \prod_{i=1}^m \frac{\phi(f(x_{\pi(i)}))}{\sum_{j=i}^m \phi(f(x_{\pi(j)}))}.$$

$$\textbf{RankCosine}: l(f; z, y) = \frac{1}{2}\left(1 - \frac{\phi(g_y(z))^T \phi(f(z))}{\|\phi(g_y(z))\| \|\phi(f(z))\|}\right).$$

Where $g_y(x)$ is the score of x given in the ground-truth, $\phi(\cdot)$ is the transformation function, which is an increasing and strictly positive function. In this paper, we assume ϕ to be differentiable[4].

These algorithms actually minimize the following empirical risk to learn the ranking model:

$$\widehat{R}_l(f; S) = \frac{1}{n} \sum_{i=1}^n l(f; z_i, y_i).$$

To conduct meaningful comparison among the algorithms, we normalize their listwise loss functions to the same range, e.g., $[0, 1]$. We further make two assumptions on the feature vector and the ranking model. Let x be the feature vector of a query-document pair, we assume that $\forall x \in \mathcal{X}, \|x\| \leq M$. Furthermore, following the common practice as in (Bartlett & Mendelson, 2002), we assume that the ranking model f to be learned is from the linear function class $\mathcal{F} = \{x \to w \cdot x : \|w\| \leq B.\}$[5]. Therefore, we have $\forall x \in \mathcal{X}, \forall f \in \mathcal{F}, |f(x)| \leq BM$. Then, for a listwise ranking algorithm $\mathcal{A}$ (e.g., ListMLE, ListNet or RankCosine), we normalize its original listwise loss function $l(f; z, y)$ to be $l_{\mathcal{A}}(f; z, y) = \frac{l(f;z,y)}{Z_{\mathcal{A}}}$, where $Z_{ListMLE} = Z_{ListNet} = m(\log m + \log \frac{\phi(BM)}{\phi(-BM)})$, and $Z_{RankCosine} = 1$. With the normalization, the algorithms actually minimize the following empirical risk in learning:

$$\widehat{R}_{l_{\mathcal{A}}}(f; S) = \frac{1}{n} \sum_{i=1}^m l_{\mathcal{A}}(f; z_i, y_i).$$

Note that since the normalizer is a constant, the normalization only changes the empirical and expected

[4]Most transformation functions used in the literature, such as the linear, exponential and sigmoid functions, satisfy this requirement.

[5]In this paper, we take the linear ranking model as an example, and leave the investigations on more complicated ranking models to future work.

risks of an algorithm, but does not change their optimal solutions.

3.2.2. Upper Bounds of $\widehat{\mathcal{R}}(l_{\mathcal{A}} \circ \mathcal{F})$

According to Theorem 1, the generalization ability of a listwise ranking algorithm depends on its Rademacher Average $\widehat{\mathcal{R}}(l_{\mathcal{A}} \circ \mathcal{F})$. Hereafter, we discuss the upper bound of $\widehat{\mathcal{R}}(l_{\mathcal{A}} \circ \mathcal{F})$ for each algorithm we are concerned with. The results are summarized in Theorem 2.

Theorem 2. *The upper bounds of $\widehat{\mathcal{R}}(l_{\mathcal{A}} \circ \mathcal{F})$ of ListMLE, ListNet, and RankCosine can be represented as the following form:*

$$\widehat{\mathcal{R}}(l_{\mathcal{A}} \circ \mathcal{F}) \leq C_{\mathcal{A}}(\phi) N(\phi) \widehat{\mathcal{R}}(\mathcal{F}),$$

where $\mathcal{A}$ stands for a listwise ranking algorithm, $N(\phi) = \sup_{x \in [-BM, BM]} \phi'(x)$, and $C_{\mathcal{A}}(\phi)$ is defined as follows:

$$C_{ListMLE}(\phi) = \frac{2}{\phi(-BM)(\log m + \log \frac{\phi(BM)}{\phi(-BM)})},$$
$$C_{ListNet}(\phi) = \frac{2m!}{\phi(-BM)(\log m + \log \frac{\phi(BM)}{\phi(-BM)})},$$
$$C_{RankCosine}(\phi) = \frac{\sqrt{m}}{2\phi(-BM)}.$$

Proof. Due to space limitations, we take ListMLE as an example to illustrate the proof sketch.

Substituting the listwise loss function of ListMLE into the definition of $\widehat{\mathcal{R}}(l_{\mathcal{A}} \circ \mathcal{F})$, and using the properties of Rademacher Average (Eq. 1 and Eq. 3), we obtain the following inequality:

$$\begin{aligned} &\widehat{R}(l_{ListMLE} \circ \mathcal{F}) \qquad (4) \\ &\leq \frac{\sum_{j=1}^{m} E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \log(\phi(f(x^{(i)}_{y_i(j)})))]}{m(\log m + \log \frac{\phi(BM)}{\phi(-BM)})} \\ &+ \frac{\sum_{j=1}^{m} E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \log(\sum_{s=j}^{m} \phi(f(x^{(i)}_{y_i(s)})))]}{m(\log m + \log \frac{\phi(BM)}{\phi(-BM)})}. \end{aligned}$$

First, let us consider the term $E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \log(\phi(f(x^{(i)}_{y_i(j)})))]$. Define $\varphi(t) = \log(1+t), t \in [\phi(-BM) - 1, \phi(BM) - 1]$, then $\varphi(0) = 0$, and $\varphi'(t) = \frac{1}{1+t} \leq \frac{1}{\phi(-BM)}$. With the property of Rademacher Average (Eq. 2), the following inequality holds:

$$\begin{aligned} &E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \log(\phi(f(x^{(i)}_{y_i(j)})))] \\ &\leq \frac{1}{\phi(-BM)} E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i (\phi(f(x^{(i)}_{y_i(j)})) - 1)]. \end{aligned}$$

Furthermore, we have $E\sigma_i = 0$ from the definition of σ_i. Therefore,

$$\begin{aligned} &E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \log(\phi(f(x^{(i)}_{y_i(j)})))] \\ &\leq \frac{1}{\phi(-BM)} E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi(f(x^{(i)}_{y_i(j)}))], \qquad (5) \end{aligned}$$

Second, let us consider the term $E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \log(\sum_{s=j}^{m} \phi(f(x^{(i)}_{y_i(s)})))]$. With a similar technique to that used for the first term and the property of Rademacher Average (Eq. 3), the following inequality holds:

$$\begin{aligned} &E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \log(\sum_{s=j}^{m} \phi(\{f(x^{(i)}_{y_i(s)})\}))] \\ &\leq \frac{1}{(m-j+1)\phi(-BM)} E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i (\sum_{s=j}^{m} \phi(f(x^{(i)}_{y_i(s)})))] \\ &\leq \frac{1}{\phi(-BM)} \sum_{s=j}^{m} E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi(f(x^{(i)}_{y_i(s)}))]. \qquad (6) \end{aligned}$$

Here $x^{(i)}_{y_i(s)}$ are i.i.d. since they are actually documents from different queries with same positions in the ground-truth permutations. Combining Eq. 4, Eq. 5 and Eq. 6, we obtain the following inequality:

$$\begin{aligned} &\widehat{\mathcal{R}}(l_{ListMLE} \circ \mathcal{F}) \\ &\leq \frac{2}{\phi(-BM)(\log m + \log \frac{\phi(BM)}{\phi(-BM)})} \widehat{\mathcal{R}}(\phi \circ \mathcal{F}) \\ &= C_{ListMLE}(\phi) \widehat{\mathcal{R}}(\phi \circ \mathcal{F}), \end{aligned}$$

where $\widehat{\mathcal{R}}(\phi \circ \mathcal{F}) = E_\sigma[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \phi(f(x_i))]$. Furthermore, using the fact that ϕ is differentiable, and the property of Rademacher Average (Eq.2), we have $\widehat{R}(\phi \circ \mathcal{F}) \leq N(\phi) \widehat{R}(\mathcal{F})$, where $N(\phi) = \sup_{x \in [-BM, BM]} \phi'(x)$. Therefore we can conclude the theorem. □

The theorem shows that $\widehat{\mathcal{R}}(l_{\mathcal{A}} \circ \mathcal{F})$ of an algorithm $\mathcal{A}$ is determined by the following three factors:

1. $C_{\mathcal{A}}(\phi)$, an algorithm-dependent factor, which is a function of both the listwise loss function and the transformation function ϕ. $C_{\mathcal{A}}(\phi)$ is also related to the range of $\phi(x), x \in [-BM, BM]$. More detailed discussions on $C_{\mathcal{A}}(\phi)$ will be given in the next section.

2. $N(\phi)$, an algorithm-independent factor, which measures the smoothness of the transformation function ϕ. We will discuss this factor in the next section.

3. $\widehat{\mathcal{R}}(\mathcal{F})$, the Rademacher Average of the ranking function class. According to (Bartlett & Mendelson, 2002), in the case of a linear ranking function, we have $\widehat{\mathcal{R}}(\mathcal{F}) \leq \frac{2BM}{\sqrt{n}}$.

3.3. Discussion

Combining Theorem 1 and Theorem 2, we obtain that with probability at least $1-\delta$, the following inequality holds[6]:

$$\sup_{f\in\mathcal{F}}(R_{l_{\mathcal{A}}}(f) - \widehat{R}_{l_{\mathcal{A}}}(f;S)) \leq \frac{4BM}{\sqrt{n}}C_{\mathcal{A}}(\phi)N(\phi) + \sqrt{\frac{2\ln\frac{2}{\delta}}{n}}.$$

From the above inequality, we can see that the generalization bound depends on $C_{\mathcal{A}}(\phi)$, $N(\phi)$, and number of training samples n. Since $C_{\mathcal{A}}(\phi)$ and $N(\phi)$ are independent of n (according to Theorem 2), when n approaches infinity, the generalization bound vanishes at the rate of $O(\frac{1}{\sqrt{n}})$.

Furthermore, in general, smaller $C_{\mathcal{A}}(\phi)N(\phi)$ leads to tighter generalization bound. With some derivations, we obtain the concrete forms of $N(\phi)$ and $C_{\mathcal{A}}(\phi)$ as shown in Table 1, with respect to different transformation functions. We first look at the bounds of different algorithms when the transformation function ϕ is fixed. Then we discuss the bound of a given algorithm when the following transformation functions are used,

- ◇ Linear Function:
  $\phi_L(x) = ax + b, x \in [-BM, BM]$.
- ◇ Exponential Function:
  $\phi_E(x) = e^{ax}, x \in [-BM, BM]$.
- ◇ Sigmoid Function:
  $\phi_S(x) = \frac{1}{1+e^{-ax}}, x \in [-BM, BM]$.

Note that in the above definitions, we assume that $a > 0$ and $b > aBM + \Delta$ (where $\Delta > 0$ is a constant), to guarantee that the transformation function is an increasing and strictly positive function. Furthermore, a should not be too small. Otherwise, $\forall x, ax \approx 0$, which will make the learning process unreasonable.

3.3.1. Generalization Bounds of Different Algorithms

For the same transformation function ϕ, the generalization bounds of the algorithms only depend on $C_{\mathcal{A}}(\phi)$. According to Table. 1, we have the following results with regard to the factor.

- *The generalization bound of ListMLE is much tighter than that of ListNet, especially when m, the length of list, is large.*

  Actually, the listwise loss for ListNet can be regarded as the weighted sum of the losses for ListMLE with respect to different targeted permutations. In this regard, it is easy to prove that $C_{ListNet}(\phi) = m!C_{ListMLE}(\phi)$.

- *The bound of ListMLE decreases monotonically, while the bounds of ListNet and RankCosine increase monotonically, with respect to m.*

  For RankCosine, m can be regarded as the dimension of list representation (each dimension corresponds to a document). When the dimension is high, there are many terms to sum up when computing the Rademacher Average. Considering Eq. 3, the bound of RankCosine will increase monotonically with respect to m. For ListMLE, m determines $Z_{ListMLE}$. From the property of Rademacher Average (Eq. 1), we can clearly see that the bound will decrease monotonically with respect to m. For ListNet, $C_{ListNet}(\phi) = m!C_{ListMLE}(\phi)$. Since $m!$ increases faster than $Z_{ListMLE}$, the bound eventually increases monotonically with respect to m.

- *When $m \geq 6$, the generalization bound of ListMLE is always tighter than that of RankCosine; otherwise, certain conditions need to be satisfied to ensure that the generalization bound of ListMLE is still tighter than that of RankCosine.*

  It has been shown above, when m becomes larger the generalization bound of ListMLE will become tighter while that of RankCosine will become looser. More precisely, when $m \geq 6$, ListMLE will have a tighter bound than RankCosine.

  If $m < 6$ and we still want the bound of ListMLE to be tighter, then we need to carefully choose ϕ and other parameters. The corresponding conditions[7] are as follows: for ϕ_L, $1 < \frac{b}{aBM} < \frac{\exp\frac{4}{\sqrt{m}}+m}{\exp\frac{4}{\sqrt{m}}-m}$; for ϕ_E, $aBM > \frac{1}{2}(\frac{4}{\sqrt{m}} - \log m)$; and for ϕ_S, $aBM > \frac{4}{\sqrt{m}} - \log m$. Otherwise, the bound of RankCosine will be tighter.

In practice, when we employ the listwise ranking algorithms, the average length of list (e.g. the number of training documents per query) is usually not small. Take the benchmark LETOR dataset as an example, the average length of lists in TD2003 and TD2004

[6] The notations are the same as in Theorem 1 and Theorem 2.

[7] Simpler but stronger conditions: for ϕ_L, $1 < \frac{b}{aBM} < \frac{3}{2}$; for ϕ_E, $aBM > \frac{3}{4}$; for ϕ_S, $aBM > \frac{3}{2}$.

Table 1. $N(\phi)$ and $C_{\mathcal{A}}(\phi)$ for Listwise Ranking Algorithms

ϕ	$N(\phi)$	$C_{ListMLE}(\phi)$	$C_{ListNet}(\phi)$	$C_{RankCosine}(\phi)$
ϕ_L	a	$\frac{2}{(b-aBM)(\log m+\log\frac{b+aBM}{b-aBM})}$	$\frac{2m!}{(b-aBM)(\log m+\log\frac{b+aBM}{b-aBM})}$	$\frac{\sqrt{m}}{2(b-aBM)}$
ϕ_E	ae^{aBM}	$\frac{2e^{aBM}}{\log m+2aBM}$	$\frac{2m!e^{aBM}}{\log m+2aBM}$	$\frac{\sqrt{m}e^{aBM}}{2}$
ϕ_S	$\frac{ae^{aBM}}{(1+e^{-aBM})^2}$	$\frac{2(1+e^{aBM})}{\log m+aBM}$	$\frac{2m!(1+e^{aBM})}{\log m+aBM}$	$\frac{\sqrt{m}(1+e^{aBM})}{2}$

Table 2. Results of Different ϕ for RankCosine (Where $K = e^{-aBM}$)

Condition	**Result**
$b - aBM > K^2(1+K)$	$\phi_L \succ \phi_S \succ \phi_E$
$K^2 < b - aBM < K^2(1+K)$	$\phi_S \succ \phi_L \succ \phi_E$
$0 < b - aBM < K^2$	$\phi_S \succ \phi_E \succ \phi_L$

Table 3. Results of Different ϕ for ListMLE and ListNet When $b > K^2(1+K) + aBM$ (Where $K = e^{-aBM}$, $N = \log\frac{b+aBM}{b-aBM}$)

Condition	**Result**
$\log m > \frac{2aBMK^2-(b-aBM)N}{b-aBM-e^{-2aBM}}$	$\phi_L \succ \phi_E$
$\log m < \frac{2aBMK^2-(b-aBM)N}{b-aBM-e^{-2aBM}}$	$\phi_E \succ \phi_L$
$\log m > \frac{aBMK^2(1+K)-(b-aBM)N}{b-aBM-K^2(1+K)}$	$\phi_L \succ \phi_S$
$\log m < \frac{aBMK^2(1+K)-(b-aBM)N}{b-aBM-K^2(1+K)}$	$\phi_S \succ \phi_L$
$\log m > aBM(e^{aBM}-1)$	$\phi_S \succ \phi_E$
$\log m < aBM(e^{aBM}-1)$	$\phi_E \succ \phi_S$

datasets is about 1000, and the average length of lists in OHSUMED dataset is about 150. In this case, the upper bound on the generalization ability of ListMLE will be the best among the three listwise ranking algorithms under investigation.

3.3.2. Generalization Bounds w.r.t. Different Transformation Functions

By jointly considering $N(\phi)$ and $C_{\mathcal{A}}(\phi)$, we get the results presented in Tables 2 and 3. Here we use $\phi_1 \succ \phi_2$ to represent the case in which the generalization bound of an algorithm with transformation function ϕ_1 is tighter than that with transformation function ϕ_2. Due to space limitations, we omit the detailed proofs and only make some discussions here.

- *For RankCosine, the sigmoid transformation function is always better than the exponential transformation function in terms of the generalization bound.*

  As seen from Table 1, $C_{RankCosine}(\phi_E)N(\phi_E) = (1+e^{aBM})C_{RankCosine}(\phi_S)N(\phi_S)$. Since $aBM > 0$, $1 + e^{aBM} > 2$. Therefore, we always have: for RankCosine, $\phi_S \succ \phi_E$.

- *For RankCosine, the linear transformation function is better than the sigmoid transformation function in terms of the generalization bound in most cases.*

  Note that B and M are fixed, and a is not very small (as discussed above). In this case, aBM will not be very small either. Since e^{-u} decreases rapidly as u increases, both e^{-2aBM} and $e^{-2aBM}(1 + e^{-aBM})$ will be very close to zero. Therefore, given $b > aBM + \Delta$, it is very likely that we also have $b > e^{-2aBM}(1+e^{-aBM})+aBM$. Referring to the results in Table 2, we obtain that for RankCosine $\phi_L \succ \phi_S$ holds in such a case.

- *For ListMLE and ListNet, the linear transformation function is the best choice in terms of generalization bound in most cases.*

  For ListMLE and ListNet, when $b > e^{-2aBM}(1 + e^{-aBM})+aBM$ (which is likely to be true in most cases as discussed above), we have the results as listed in Table 3. Generally speaking, when m is not very small (which is true in practice as discussed above), we have $\phi_L \succ \phi_S \succ \phi_E$. For example, suppose $a = 1, M = 1, B = 10, b = 11$, then as long as $m \geq 1$, we will have $\phi_L \succ \phi_S \succ \phi_E$.

In summary, the above discussions imply that for all the three algorithms the use of linear transformation function will result in tighter generalization bounds in most cases. Actually, the compound function of the transformation function ϕ and the ranking function f can be viewed as a new ranking function class. For the same f, ϕ determines the complexity of the new ranking function. Intuitively, the use of simpler ranking functions will result in better generalization ability. Obviously the linear transformation function is the simplest among all the three transformation functions. Therefore we can make the conclusion.

4. Conclusions and Future Work

In this paper, we have proposed a framework for ranking, and analyzed the generalization ability of listwise ranking algorithms with the framework. The analysis is based on the theory of Rademacher Average. We have proved a generalization bound for a listwise ranking algorithm that depends on the Rademacher Average of the class of compound functions. The compound functions take listwise loss functions as outer functions and ranking models as inner functions. We have computed the Rademacher Averages for three listwise ranking algorithms: ListMLE, ListNet, and RankCosine, and discussed the tightness of generalization bounds in different situations. To our knowledge, this is the first work that has formally discussed the generalization ability of listwise ranking algorithms.

The major findings in this work are as follows: (1) when the number of training samples approaches infinity, the generalization bounds of the three listwise ranking algorithms will all converge to zero at the same rate of $O(\frac{1}{\sqrt{n}})$; (2) when the length of the list is larger than or equal to six, the generalization ability of ListMLE is possibly the best among the three algorithms; (3) in most cases, a linear transformation function is the best choice, in terms of generalization ability.

As future work, we plan to investigate the approximation error, and thus the generalization bound between the true expected risk (based on a true loss of ranking) and the empirical risk (based on the listwise loss).

References

Agarwal, S., & Niyogi, P. (2005). Stability and generalization of bipartite ranking algorithms. *Proc. of COLT'05* (pp. 32–47).

Bartlett, P. L., & Mendelson, S. (2002). Rademacher and gaussian complexities risk bounds and structural results. *Journal of Machine Learning* (pp. 463–482).

Bousquet, O., Boucheron, S., & Lugosi, G. (2004). Introduction to statistical learning theory. *Advanced Lectures on Machine Learning* (pp. 169–207). Springer Berlin / Heidelberg.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. *ICML '07* (pp. 129–136).

Crammer, K., & Singer, Y. (2001). Pranking with ranking. *Proceedings of NIPS 2001* (pp. 641–647).

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

Herbrich, R., Obermayer, K., & Graepel, T. (1999). Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers.* (pp. 115–132).

Lan, Y., Liu, T.-Y., Qin, T., Ma, Z., & Li, H. (2008). Query-level stability and generalization in learning to rank. *Proceedings of ICML 2008* (pp. 512–519).

Ledoux, M., & Talagrand, M. (1991). *Probability in banach spaces: isoperimetry and processes.* Berlin: Springer-Verlag.

Li, P., Burges, C., & Wu, Q. (2007). Mcrank: Learning to rank using multiple classification and gradient boosting. *NIPS2007.*

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

Mendelson, S. (2001). Rademacher averages and phase transitions in glivenko-cantelli classes. *IEEE Trans. Inf. Theory* (pp. 251–263).

Mendelson, S. (2003). A few notes on statistical learning theory. *Advanced Lectures in Machine Learning* (pp. 1–40). LNCS 2600, Springer.

Qin, T., Zhang, X.-D., Tsai, M.-F., Wang, D.-S., Liu, T.-Y., & Li, H. (2007). Query-level loss functions for information retrieval. *Information Processing & Management*, 838–855.

Xia, F., Liu, T.-Y., Wang, J., Zhang, W., & Li, H. (2008). Listwise approach to learning to rank - theory and algorithm. *Proceedings of ICML 2008* (pp. 1192–1199).