Listwise Approach to Learning to Rank - Theory and Algorithm

Fen Xia* FEN.XIA@IA.AC.CN
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, P. R. China.

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

Jue Wang JUE.WANG@IA.AC.CN
Wensheng Zhang WENSHENG.ZHANG@IA.AC.CN
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, P. R. China.

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

Abstract

This paper aims to conduct a study on the listwise approach to learning to rank. The listwise approach learns a ranking function by taking individual lists as instances and minimizing a loss function defined on the predicted list and the ground-truth list. Existing work on the approach mainly focused on the development of new algorithms; methods such as RankCosine and ListNet have been proposed and good performances by them have been observed. Unfortunately, the underlying theory was not sufficiently studied so far. To amend the problem, this paper proposes conducting theoretical analysis of learning to rank algorithms through investigations on the properties of the loss functions, including consistency, soundness, continuity, differentiability, convexity, and efficiency. A sufficient condition on consistency for ranking is given, which seems to be the first such result obtained in related research. The paper then conducts analysis on three loss functions: likelihood loss, cosine loss, and cross entropy loss. The latter two were used in RankCosine and ListNet. The use of the likelihood loss leads to the development of a new listwise method called ListMLE, whose loss function offers better properties, and also leads to better experimental results.

1. Introduction

Ranking, which is to sort objects based on certain factors, is the central problem of applications such as information retrieval (IR) and information filtering. Recently machine learning technologies called 'learning to rank' have been successfully applied to ranking, and several approaches have been proposed, including the pointwise, pairwise, and listwise approaches.

The listwise approach addresses the ranking problem in the following way. In learning, it takes ranked lists of objects (e.g., ranked lists of documents in IR) as instances and trains a ranking function through the minimization of a listwise loss function defined on the predicted list and the ground truth list. The listwise approach captures the ranking problems, particularly those in IR in a conceptually more natural way than previous work. Several methods such as RankCosine and ListNet have been proposed. Previous experiments demonstrate that the listwise approach usually performs better than the other approaches (Cao et al., 2007)(Qin et al., 2007).

Existing work on the listwise approach mainly focused on the development of new algorithms, such as RankCosine and ListNet. However, there was no sufficient theoretical foundation laid down. Furthermore, the strength and limitation of the algorithms, and the relations between the proposed algorithms were still

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.
*The work was performed when the first author was an intern at Microsoft Research Asia.

not clear. This largely prevented us from deeply understanding the approach, more critically, from devising more advanced algorithms.

In this paper, we aim to conduct an investigation on the listwise approach.

First, we give a formal definition of the listwise approach. In ranking, the input is a set of objects, the output is a permutation of the objects[1], and the model is a ranking function which maps a given input to an output. In learning, the training data is drawn i.i.d. according to an unknown but fixed joint probability distribution between input and output. Ideally we would minimize the expected $0-1$ loss defined on the predicted list and the ground truth list. Practically we instead manage to minimize an empirical surrogate loss with respect to the training data.

Second, we evaluate a surrogate loss function from four aspects: (a) consistency, (b) soundness, (c) mathematical properties of continuity, differentiability, and convexity, and (d) computational efficiency in learning. We give analysis on three loss functions: likelihood loss, cosine loss, and cross entropy loss. The first one is newly proposed in this paper, and the last two were used in RankCosine and ListNet, respectively.

Third, we propose a novel method for the listwise approach, which we call ListMLE. ListMLE formalizes learning to rank as a problem of minimizing the likelihood loss function, equivalently maximizing the likelihood function of a probability model. Due to the nice properties of the loss function, ListMLE stands to be more effective than RankCosine and ListNet.

Finally, we have experimentally verified the correctness of the theoretical findings. We have also found that ListMLE can significantly outperform RankCosine and ListNet.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 gives a formal definition to the listwise approach. Section 4 conducts theoretical analysis of listwise loss functions. Section 5 introduces the ListMLE method. Experimental results are reported in Section 6 and the conclusion and future work are given in the last section.

2. Related Work

Existing methods for learning to rank fall into three categories. The pointwise approach (Nallapati, 2004) transforms ranking into regression or classification on single objects. The pairwise approach (Herbrich et al., 1999) (Freund et al., 1998) (Burges et al., 2005) transforms ranking into classification on object pairs. The advantage for these two approaches is that existing theories and algorithms on regression or classification can be directly applied, but the problem is that they do not model the ranking problem in a straightforward fashion. The listwise approach can overcome the drawback of the aforementioned two approaches by tackling the ranking problem directly, as explained below.

For instance, Cao et al. (2007) proposed one of the first listwise methods, called ListNet. In ListNet, the listwise loss function is defined as cross entropy between two parameterized probability distributions of permutations; one is obtained from the predicted result and the other is from the ground truth. Qin et al. (2007) proposed another method called RankCosine. In the method, the listwise loss function is defined on the basis of cosine similarity between two score vectors from the predicted result and the ground truth[2]. Experimental results show that the listwise approach usually outperforms the pointwise and pariwise approaches.

In this paper, we aim to investigate the listwise approach to learning to rank, particularly from the viewpoint of loss functions. Actually similar investigations have also been conducted for classification. For instance, in classification, consistency and soundness of loss functions are well studied. Consistency forms the basis for the success of a loss function. It is known that if a loss function is consistent, then the learned classifier can achieve the optimal Bayes error rate in the large sample limit. Many well known loss functions such as hinge loss, exponential loss, and logistic loss are all consistent (cf., (Zhang, 2004)(Bartlett et al., 2003)(Lin, 2002)). Soundness of a loss function guarantees that the loss can represent well the targeted learning problem. That is, an incorrect prediction should receive a larger penalty than a correct prediction, and the penalty should reflect the confidence of prediction. For example, hinge loss, exponential loss, and logistic loss are sound for classification. In contrast, square loss is sound for regression but not for classification (Hastie et al., 2001).

3. Listwise Approach

We give a formal definition of the listwise approach to learning to rank. Let $\boldsymbol{X}$ be the input space whose

[1] In this paper, we use permutation and ranked list interchangeably.

[2] In a broad sense, methods directly optimizing evaluation measures, such as SVM-MAP (Yue et al., 2007) and AdaRank (Xu & Li, 2007) can also be regarded as listwise algorithms. We will, however, limit our discussions in this paper on algorithms like ListNet and RankCosine.

elements are sets of objects to be ranked, Y be the output space whose elements are permutations of objects, and P_{XY} be an unknown but fixed joint probability distribution of X and Y. Let $\mathbf{h} : X \to Y$ be a ranking function, and H be the corresponding function space (i.e., $\mathbf{h} \in H$). Let $\mathbf{x} \in X$ and $\mathbf{y} \in Y$, and let $y(i)$ be the index of object which is ranked at position i. The task is to learn a ranking function that can minimize the expected loss $R(\mathbf{h})$, defined as:

$$R(\mathbf{h}) = \int_{X \times Y} l(\mathbf{h}(\mathbf{x}), \mathbf{y}) dP(\mathbf{x}, \mathbf{y}), \tag{1}$$

where $l(\mathbf{h}(\mathbf{x}), \mathbf{y})$ is the $0-1$ loss function such that

$$l(\mathbf{h}(\mathbf{x}), \mathbf{y}) = \begin{cases} 1, & \text{if } \mathbf{h}(\mathbf{x}) \neq \mathbf{y} \\ 0, & \text{if } \mathbf{h}(\mathbf{x}) = \mathbf{y}, \end{cases} \tag{2}$$

That is to say, we formalize the ranking problem as a new 'classification' problem on permutations. If the permutation of the predicted result is the same as the ground truth, then we have zero loss; otherwise we have one loss. In real ranking applications, the loss can be cost-sensitive, i.e., depending on the positions of the incorrectly ranked objects. We will leave this as our future work and focus on the $0-1$ loss in this paper first. Actually, in the literature of classification, people also studied the $0-1$ loss first, before they eventually moved onto the cost-sensitive case.

It is easy to see that the optimal ranking function which can minimize the expected loss $R(\mathbf{h}_B) = \inf R(\mathbf{h})$ is given by the Bayes rule,

$$\mathbf{h}_B(\mathbf{x}) = \arg\max_{\mathbf{y} \in Y} P(\mathbf{y}|\mathbf{x}), \tag{3}$$

Since P_{XY} is unknown, formula (1) cannot be directly solved and thus $\mathbf{h}_B(\mathbf{x})$ cannot be easily obtained. In practice, we are given independently and identically distributed (i.i.d) samples $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^m \sim P_{XY}$, we instead try to obtain a ranking function $\mathbf{h} \in H$ that minimizes the empirical loss.

$$R_S(\mathbf{h}) = \frac{1}{m} \sum_{i=1}^{m} l(\mathbf{h}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}). \tag{4}$$

Note that for efficiency consideration, in practice the ranking function usually works on individual objects. It assigns a score to each object (by employing a scoring function g), sorts the objects in descending order of the scores, and finally creates the ranked list. That is to say, $\mathbf{h}(\mathbf{x}^{(i)})$ is decomposable with respect to objects. It is defined as

$$\mathbf{h}(\mathbf{x}^{(i)}) = sort(g(x_1^{(i)}), \ldots, g(x_{n_i}^{(i)})). \tag{5}$$

where $x_j^{(i)} \in \mathbf{x}^{(i)}$, n_i denotes the number of objects in $\mathbf{x}^{(i)}$, $g(\cdot)$ denotes the scoring function, and $sort(\cdot)$ denotes the sorting function. As a result, (4) becomes:

$$R_S(\mathbf{g}) = \frac{1}{m} \sum_{i=1}^{m} l(sort(g(x_1^{(i)}), \ldots, g(x_{n_i}^{(i)})), \mathbf{y}^{(i)}). \tag{6}$$

Due to the nature of the sorting function and the $0-1$ loss function, the empirical loss in (6) is inherently non-differentiable with respect to g, which poses a challenge to the optimization of it. To tackle this problem, we can introduce a surrogate loss as an approximation of (6), following a common practice in machine learning.

$$R_S^{\phi}(\mathbf{g}) = \frac{1}{m} \sum_{i=1}^{m} \phi(\mathbf{g}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)}), \tag{7}$$

where ϕ is a surrogate loss function and $\mathbf{g}(\mathbf{x}^{(i)}) = (g(x_1^{(i)}), \ldots, g(x_{n_i}^{(i)}))$. For convenience in notation, in the following sections, we sometimes write $\phi_{\mathbf{y}}(\mathbf{g})$ for $\phi(\mathbf{g}(\mathbf{x}), \mathbf{y})$ and use bold symbols such as $\mathbf{g}$ to denote vectors since for a given $\mathbf{x}$, $\mathbf{g}(\mathbf{x})$ becomes a vector.

4. Theoretical Analysis

4.1. Properties of Loss Function

We analyze the listwise approach from the viewpoint of surrogate loss function. Specifically, we look at the following properties [3] of it: (a) consistency , (b) soundness, (c) continuity, differentiability, and convexity, and (d) computational efficiency in learning.

Consistency is about whether the obtained ranking function can converge to the optimal one through the minimization of the empirical surrogate loss (7), when the training sample size goes to infinity. It is a necessary condition for a surrogate loss function to be a good one for a learning algorithm (cf., Zhang (2004)).

Soundness is about whether the loss function can indeed represent loss in ranking. For example, an incorrect ranking should receive a larger penalty than a correct ranking, and the penalty should reflect the confidence of the ranking. This property is particularly important when the size of training data is small, because it can directly affect the training results.

4.2. Consistency

We conduct analysis on learning to rank algorithms from the viewpoint of consistency. As far as we know,

[3] In addition, convergence rate is another issue to consider. We leave it as future work.

this is the first work discussing the consistency issue for ranking.

In the large sample limit, minimizing the empirical surrogate loss (7) amounts to minimizing the following expected surrogate loss

$$R^{\phi}(\mathbf{g}) = E_{X,Y}\{\phi_{\mathbf{y}}(\mathbf{g}(\mathbf{x}))\} = E_X\{Q(\mathbf{g}(\mathbf{x}))\} \quad (8)$$
$$\text{where } Q(\mathbf{g}(\mathbf{x})) = \sum_{\mathbf{y}\in Y} P(\mathbf{y}|\mathbf{x})\phi_{\mathbf{y}}(\mathbf{g}(\mathbf{x})).$$

Here we assume $\mathbf{g}(\mathbf{x})$ is chosen from a vector Borel measurable function set, whose elements can take any value from $\Omega \subset R^n$.

When the minimization of (8) can lead to the minimization of the expected $0-1$ loss (1), we say the surrogate loss function is consistent. A equivalent definition can be found in Definition 2. Actually this equivalence relationship has been discussed in related work on the consistency of classification (Zhang, 2004).

Definition 1. *We define $\Lambda_{\mathbf{y}}$ as the space of all possible probabilities on the permutation space Y, i.e., $\Lambda_Y \triangleq \{\mathbf{p} \in R^{|Y|} : \sum_{\mathbf{y}\in Y} p_{\mathbf{y}} = 1, p_{\mathbf{y}} \geq 0\}$.*

Definition 2. *The loss $\phi_{\mathbf{y}}(\mathbf{g})$ is consistent on a set $\Omega \subset R^n$ with respect to the ranking loss (1), if the following conditions hold: $\forall \mathbf{p} \in \Lambda_Y$, assume $\mathbf{y}^* = \arg max_{\mathbf{y}\in Y} p_{\mathbf{y}}$ and $Y^c_{\mathbf{y}^*}$ denotes the space of permutations after removing $\mathbf{y}^*$, we have*

$$\inf_{\mathbf{g}\in\Omega} Q(\mathbf{g}) < \inf_{\mathbf{g}\in\Omega, sort(\mathbf{g})\in Y^c_{\mathbf{y}^*}} Q(\mathbf{g})$$

We next give sufficient conditions of consistency in ranking.

Definition 3. *A permutation probability space Λ_Y is order preserving with respect to object i and j, if the following conditions hold: $\forall \mathbf{y} \in Y_{i,j} \triangleq \{\mathbf{y} \in Y : y^{-1}(i) < y^{-1}(j)\}$ where $y^{-1}(i)$ denotes the position for object i in $\mathbf{y}$, denote $\sigma^{-1}\mathbf{y}$ as the permutation which exchanges the positions of object i and j while hold others unchanged for $\mathbf{y}$, we have $p_{\mathbf{y}} > p_{\sigma^{-1}\mathbf{y}}$.*

Definition 4. *The loss $\phi_{\mathbf{y}}(\mathbf{g})$ is order sensitive on a set $\Omega \subset R^n$, if $\phi_{\mathbf{y}}(\mathbf{g})$ is a non-negative differentiable function and the following two conditions hold:*

1. *$\forall \mathbf{y} \in Y$, $\forall i < j$, denote $\sigma\mathbf{y}$ as the permutation which exchanges the object on position i and that on position j while holds others unchanged for $\mathbf{y}$, if $g_{y(i)} < g_{y(j)}$, then $\phi_{\mathbf{y}}(\mathbf{g}) \geq \phi_{\sigma\mathbf{y}}(\mathbf{g})$ and with at least one $\mathbf{y}$, the strict inequality holds.*
2. *If $g_i = g_j$, then either $\forall \mathbf{y} \in Y_{i,j}$, $\frac{\partial\phi_{\mathbf{y}}(\mathbf{g})}{\partial g_i} \leq \frac{\partial\phi_{\mathbf{y}}(\mathbf{g})}{\partial g_j}$, or $\forall \mathbf{y} \in Y_{i,j}$, $\frac{\partial\phi_{\mathbf{y}}(\mathbf{g})}{\partial g_i} \geq \frac{\partial\phi_{\mathbf{y}}(\mathbf{g})}{\partial g_j}$, and with at least one $\mathbf{y}$, the strict inequality holds.*

Theorem 5. *Let $\phi_{\mathbf{y}}(\mathbf{g})$ be an order sensitive loss function on $\Omega \subset R^n$. $\forall n$ objects, if its permutation probability space is order preserving with respect to $n-1$ objective pairs $(j_1, j_2), (j_2, j_3), \cdots, (j_n - 1, j_n)$. Then the loss $\phi_{\mathbf{y}}(\mathbf{g})$ is consistent with respect to (1).*

Due to space limitations, we only give the proof sketch. First, we can show if the permutation probability space is order preserving with respect to $n-1$ objective pairs $(j_1, j_2), (j_2, j_3), \cdots, (j_n - 1, j_n)$, then the permutation with the maximum probability is $\mathbf{y}^* = (j_1, j_2, \cdots, j_n)$. Second, for an order sensitive loss function, for any order preserving object pairs (j_1, j_2), the vector $\mathbf{g}$ which minimizes $Q(\mathbf{g})$ in (8) should assign a larger score to j_1 than to j_2. This can be proven by the change of loss due to exchanging the scores of j_1 and j_2. Given all these results and Definition 2, we can prove Theorem 5 by means of contradiction.

Theorem 5 gives sufficient conditions for a surrogate loss function to be consistent: the permutation probability space should be order preserving and the function should be order sensitive. Actually, the assumption of order preserving has already been made when we use the scoring function and sorting function for ranking. The property of order preserving has also been explicitly or implicitly used in previous work, such as Cossock and Zhang (2006). The property of order sensitive shows that starting with a ground truth permutation, the loss will increase if we exchange the positions of two objects in it, and the speed of increase in loss is sensitive to the positions of objects.

4.3. Case Studies

We look at the four properties of three loss functions.

4.3.1. Likelihood Loss

We introduce a new loss function for listwise approach, which we call likelihood loss. The likelihood loss function is defined as:

$$\phi(\mathbf{g}(\mathbf{x}), \mathbf{y}) = -\log P(\mathbf{y}|\mathbf{x}; \mathbf{g}) \quad (9)$$
$$\text{where } P(\mathbf{y}|\mathbf{x}; \mathbf{g}) = \prod_{i=1}^{n} \frac{\exp(g(x_{y(i)}))}{\sum_{k=i}^{n} \exp(g(x_{y(k)}))}.$$

Note that we actually define a parameterized exponential probability distribution over all the permutations given the predicted result (by the ranking function), and define the loss function as the negative log likelihood of the ground truth list. The probability distribution turns out to be a Plackett-Luce model (Marden, 1995).

The likelihood loss function has the nice properties as below.

First, the likelihood loss is consistent. The following proposition shows that the likelihood loss is order sensitive. Therefore, according to Theorem 5, it is consistent. Due to the space limitations, we omit the proof.

Proposition 6. *The likelihood loss (9) is order sensitive on* $\Omega \subset R^n$.

Second, the likelihood loss function is sound. For simplicity, suppose that there are two objects to be ranked (similar argument can be made when there are more objects). The two objects receive scores of g_1 and g_2 from a ranking function. Figure 1(a) shows the scores, and the point $\mathbf{g} = (g_1, g_2)$. Suppose that the first object is ranked below the second object in the ground truth. Then the upper left area above line $g_2 = g_1$ corresponds to correct ranking; and the lower right area incorrect ranking. According to the definition of likelihood loss, all the points on the line $g_2 = g_1 + d$ has the same loss. Therefore, we say the likelihood loss only depends on d. Figure 1(b) shows the relation between the loss function and d. We can see the loss function decreases monotonously as d increases. It penalizes negative values of d more heavily than positive ones. This will make the learning algorithm focus more on avoiding incorrect rankings. In this regard, the loss function is a good approximation of the $0-1$ loss.

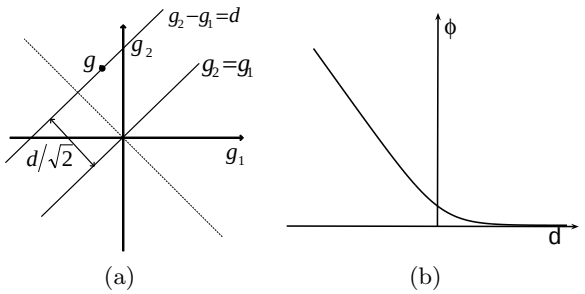

Figure 1. (a) Ranking scores of predicted result; (b) Loss ϕ v.s. d for the likelihood loss.

Third, it is easy to verify that the likelihood loss is continuous, differentiable, and convex (Boyd & Vandenberghe, 2004). Furthermore, the loss can be computed efficiently, with time complexity of linear order to the number of objects.

With the above good properties, a learning algorithm which optimizes the likelihood loss will become powerful for creating a ranking function.

4.3.2. Cosine Loss

The cosine loss is the loss function used in RankCosine (Qin et al., 2007), a listwise method. It is defined on the basis of the cosine similarity between the score vector of the ground truth and that of the predicted result.

$$\phi(\mathbf{g}(\mathbf{x}), \mathbf{y}) = \frac{1}{2}(1 - \frac{\psi_{\mathbf{y}}(\mathbf{x})^T \mathbf{g}(\mathbf{x})}{\|\psi_{\mathbf{y}}(\mathbf{x})\| \|\mathbf{g}(\mathbf{x})\|}). \tag{10}$$

The score vector of the ground truth is produced by a mapping $\psi_{\mathbf{y}}(\cdot) : R^d \to R$, which retains the order in a permutation, i.e, $\psi_{\mathbf{y}}(x_{y(1)}) > \cdots > \psi_{\mathbf{y}}(x_{y(n)})$.

First, we can prove that the cosine loss is consistent, given the following proposition. Due to space limitations, we omit the proof.

Proposition 7. *The cosine loss (10) is order sensitive on* $\Omega \subset R^n$.

Second, the cosine loss is not very sound. Let us again consider the case of ranking two objects. Figure 2(a) shows point $\mathbf{g} = (g_1, g_2)$ representing the scores of the predicted result and point $\mathbf{g}_\psi$ representing the ground truth (which depends on the mapping function ψ). We denote the angle from point $\mathbf{g}$ to line $g_2 = g_1$ as α, and the angle from $\mathbf{g}_\psi$ to line $g_2 = g_1$ as $\alpha_{\mathbf{g}_\psi}$. We investigate the relation between the loss and the angle α. Figure 2(b) shows the cosine loss as a function of α. From this figure, we can see that the cosine loss is not a monotonously decreasing function of α. When $\alpha > \alpha_{\mathbf{g}_\psi}$, it increases quickly, which means that it can heavily penalize correct rankings. Furthermore, the mapping function and thus $\alpha_{\mathbf{g}_\psi}$ can also affect the loss function. Specifically, the curve of the loss function can shift from left to right with different values of $\alpha_{\mathbf{g}_\psi}$. Only when $\alpha_{\mathbf{g}_\psi} = \pi/2$, it becomes a relatively satisfactory representation of loss for the learning problem.

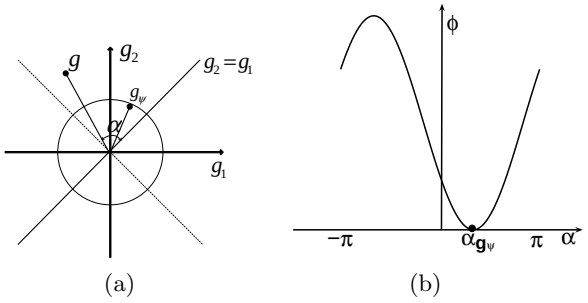

Figure 2. (a) Ranking scores of predicted result and ground truth; (b) Loss ϕ v.s. angle α for the cosine loss.

Third, it is easy to see that the cosine loss is continuous, differentiable, but not convex. It can also be computed in an efficient manner with a time complexity linear to the number of objects.

4.3.3. Cross Entropy Loss

The cross entropy loss is the loss function used in ListNet (Cao et al., 2007), another listwise method. The cross entropy loss function is defined as:

$$\phi(\mathbf{g}(\mathbf{x}), \mathbf{y}) = D(P(\pi|\mathbf{x}; \psi_{\mathbf{y}})||P(\pi|\mathbf{x}; \mathbf{g})) \quad (11)$$

$$\text{where} \quad P(\pi|\mathbf{x}; \psi_{\mathbf{y}}) = \prod_{i=1}^{n} \frac{\exp(\psi_{\mathbf{y}}(x_{\pi(i)}))}{\sum_{k=i}^{n} \exp(\psi_{\mathbf{y}}(x_{\pi(k)}))}$$

$$P(\pi|\mathbf{x}; \mathbf{g}) = \prod_{i=1}^{n} \frac{\exp(g(x_{\pi(i)}))}{\sum_{k=i}^{n} \exp(g(x_{\pi(k)}))}$$

where ψ is a mapping function whose definition is similar to that in RankCosine.

First, we can prove that the cross entropy loss is consistent, given the following proposition. Due to space limitations, we omit the proof.

Proposition 8. *The cross entropy loss (11) is order sensitive on* $\Omega \subset R^n$.

Second, the cross entropy loss is not very sound. Again, we look at the case of ranking two objects. $\mathbf{g} = (g_1, g_2)$ denotes the ranking scores of the predicted result. $\mathbf{g}_\psi$ denotes the ranking scores of the ground truth (depending on the mapping function). Similar to the discussions in the likelihood loss, the cross entropy loss only depends on the quantity d. Figure 3(a) illustrates the relation between $\mathbf{g}$, $\mathbf{g}_\psi$, and d. Figure 3(b) shows the cross entropy loss as a function of d. As can be seen that the loss function achieves its minimum at point $d_{\mathbf{g}_\psi}$, and then increases as d increases. That means it can heavily penalize those correct rankings with higher confidence. Note that the mapping function also affects the penalization. According to mapping functions, the penalization on correct rankings can be even larger than that on incorrect rankings.

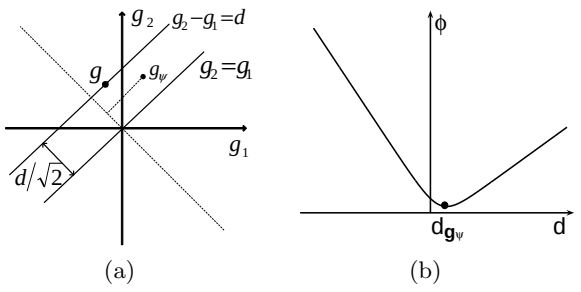

Figure 3. (a) Ranking scores of predicted result and ground truth; (b) Loss ϕ v.s. d for the cross entropy loss.

Third, it is easy to see that the cross entropy loss is continuous and differentiable. It is also convex because the log of a convex function is still convex, and the set of convex function is closed under addition (Boyd & Vandenberghe, 2004). However, it cannot be computed in an efficient manner. The time complexity is of exponential order to the number of objects.

Algorithm 1 ListMLE Algorithm

Input: training data $\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{y}^{(m)})\}$
Parameter: learning rate η, tolerance rate ϵ
Initialize parameter ω
repeat
  for $i = 1$ **to** m **do**
    Input $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ to Neural Network and compute gradient $\triangle\omega$ with current ω
    Update $\omega = \omega - \eta \times \triangle\omega$
  end for
  calculate likelihood loss on the training set
until change of likelihood loss is below ϵ
Output: Neural Network model ω

Table 1 gives a summary of the properties of the loss functions. All the three loss functions as aforementioned are consistent, as well as continuous and differentiable. The likelihood loss is better than the cosine loss in terms of convexity and soundness, and is better than the cross entropy loss in terms of time complexity and soundness.

5. ListMLE

We propose a novel listwise method referred to as ListMLE. In learning of ListMLE, we employ the likelihood loss as the surrogate loss function, since it is proven to have all the nice properties as a surrogate loss. On the training data, we actually maximize the sum of the likelihood function with respect to all the training queries.

$$\sum_{i=1}^{m} \log P(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}; \mathbf{g}). \quad (12)$$

We choose Stochastic Gradient Descent (SGD) as the algorithm for conducting the minimization. As ranking model, we choose linear Neural Network (parameterized by ω). Algorithm 1 shows the learning algorithm based on SGD.

6. Experimental Results

We conducted two experiments to verify the correctness of the theoretical findings. One data set is synthetic data, and the other is the LETOR benchmark data for learning to rank (Liu et al., 2007).

Table 1. Comparison between different surrogate losses.

Loss	Consistency	Soundness	Continuity	Differentiability	Convexity	Complexity
Likelihood	√	√	√	√	√	$O(n)$
Cosine	√	×	√	√	×	$O(n)$
Cross entropy	√	×	√	√	√	$O(n! \cdot n)$

6.1. Experiment on Synthetic Data

We conducted an experiment using a synthetic data set. We created the data as follows. First, we randomly sample a point according to the uniform distribution on the square area $[0, 1] \times [0, 1]$. Then we assign to the point a score using the following rule, $y = x_1 + 10x_2 + \epsilon$ where ϵ denotes a random variable normally distributed with mean of zero and standard deviation of 0.005. In total, we generate 15 points and their scores in this way, and create a permutation on the points based on their scores, which forms an instance of ranking. We repeat the process and make 100 training instances, 100 validation instances, and 100 testing instances. We applied RankCosine, ListNet[4], and ListMLE to the data.

We tried different score mapping functions for RankCosine and ListNet, and used five most representative ones, i.e., $log(15 - r)$, $\sqrt{15 - r}$, $15 - r$, $(15 - r)^2$ and $\exp(15 - r)$, where r denotes the positions of objects. We denote the mapping functions as *log*, *sqrt*, *l*, *q*, and *exp* for simplicity. The experiments were repeated 20 times with different initial values of parameters in the Neural Network model. Table 2 shows the means and standard deviations of the accuracies and Mean Average Precision (MAP)(Baeza-Yates & Ribeiro-Neto, 1999) of the three algorithms. The accuracy measures the proportion of correctly ranked instances and MAP[5] is a commonly used measure in IR.

As shown in the table, ListMLE achieves the best performance among all the algorithms in terms of both accuracy and MAP, owing to good properties of its loss function. The accuracies of RankCosine and ListNet vary according to the mapping functions. Especially, RankCosine achieves an accuracy of only 0.047 when using the mapping function *exp* while 0.917 when using the mapping function *l*. This result indicates that the performances of the cosine loss and the cross entropy loss depend on the mapping functions, while finding a suitable mapping function is not easy. Furthermore, RankCosine has a larger variance than ListMLE and ListNet. The likely explanation is that RankCosine's performance is sensitive to the initial values of parameters due to the non-convexity of its loss function.

[4]The top-1 version of the cross entropy loss was employed as in the original work (Cao et al., 2007).

[5]When calculating MAP, we treated the top-1 items as relevant and the other as irrelevant.

Table 2. The performance of three algorithms on the synthetic data set.

Algorithm	Accuracy	MAP
ListMLE	0.92 ± 0.011	0.999 ± 0.002
ListNet-log	0.905 ± 0.010	0.999 ± 0.002
ListNet-sqrt	0.917 ± 0.009	0.999 ± 0.002
ListNet-l	0.767 ± 0.021	0.995 ± 0.003
ListNet-q	0.868 ± 0.028	0.999 ± 0.002
ListNet-exp	0.832 ± 0.074	0.997 ± 0.004
RankCosine-log	0.180 ± 0.217	0.948 ± 0.034
RankCosine-sqrt	0.080 ± 0.159	0.886 ± 0.056
RankCosine-l	0.917 ± 0.112	0.999 ± 0.002
RankCosine-q	0.102 ± 0.161	0.890 ± 0.060
RankCosine-exp	0.047 ± 0.163	0.746 ± 0.136

6.2. Experiment on OHSUMED Data

We also conducted an experiment on OHSUMED, a benchmark data set for learning to rank provided in LETOR. There are in total 106 queries, and 16,140 query-document pairs upon which relevance judgments are made. The relevance judgments are either definitely relevant, possibly relevant, or not relevant. The data was in the form of feature vector and relevance label. There are in total 25 features. We used the data split provided in LETOR to conduct five-fold cross validation experiments. In evaluation, besides MAP, we adopted another measures commonly used in IR: Normalized Discounted Cumulative Gain (NDCG) (Jarvelin & Kekanainen, 2000).

Note that here the ground truth in the data is given as partial ranking, while the methods need to use total ranking (permutation) in training. To bridge the gap, for RankCosine and ListNet, we adopted the methods proposed in the papers (Cao et al., 2007) (Qin et al., 2007). For ListMLE we randomly selected one perfect permutation for each query from among the possible perfect permutations based on the ground truth.

We applied RankCosine, ListNet, and ListMLE to the data. The results reported below are those averaged over five trials. As shown in Figure 4, ListMLE achieves the best performance among all the algorithms. Especially, on NDCG@1, it has more than

5-point gains over RankCosine which is at the second place. We also conducted the t-test on the improvements of ListMLE over the other two algorithms. The results show that the improvements are statistically significant for NDCG@5, NDCG@7, NDCG@8, NDCG@9, and NDCG@10 (p-value < 0.05).

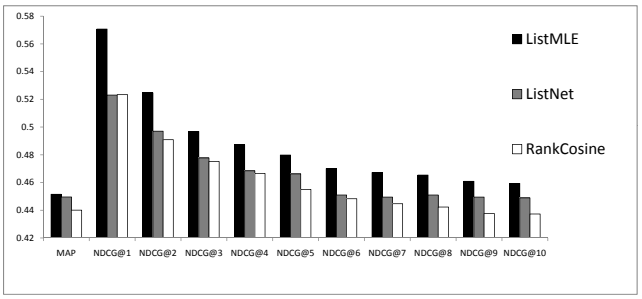

Figure 4. Ranking performance on OHSUMED data.

7. Conclusion

In this paper, we have investigated the theory and algorithms of the listwise approach to learning to rank. We have pointed out that to understand the effectiveness of a learning to rank algorithm, it is necessary to conduct theoretical analysis on its loss function. We propose investigating a loss function from the viewpoints of (a) consistency, (b) soundness, (c) continuity, differentiability, convexity, and (d) efficiency. We have obtained some theoretical results on consistency of ranking. We have conducted analysis on the likelihood loss, cosine loss, and cross entropy loss. The result indicates that the likelihood loss has better properties than the other two losses. We have then developed a new learning algorithm using the likelihood loss, called ListMLE and demonstrated its effectiveness through experiments.

There are several directions which we can further explore. (1) We want to conduct more theoretical analysis on the properties of loss functions, for example, weaker conditions for consistency and the rates of convergence. (2) We plan to study the case where cost-sensitive loss function is used instead of the $0-1$ loss function in defining the expected loss. (3) We plan to investigate other surrogate loss functions with the tools we have developed in this paper.

References

Baeza-Yates, R., & Ribeiro-Neto, B. (Eds.). (1999). *Modern information retrieval*. Addison Wesley.

Bartlett, P. L., Jordan, M. I., & McAuliffe, J. D. (2003). *Convexity, classification, and risk bounds* (Technical Report 638). Statistics Department, University of California, Berkeley.

Boyd, S., & Vandenberghe, L. (Eds.). (2004). *Convex optimization*. Cambridge University.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. *Proceedings of ICML 2005* (pp. 89–96).

Cao, Z., Qin, T., Liu, T. Y., Tsai, M. F., & Li, H. (2007). Learning to rank: From pairwise approach to listwise approach. *Proceedings of the 24th International Conference on Machine Learning* (pp. 129–136). Corvallis, OR.

Cossock, D., & Zhang, T. (2006). Subset ranking using regression. *COLT* (pp. 605–619).

Freund, Y., Iyer, R., Schapire, R. E., & Singer, Y. (1998). An efficient boosting algorithm for combining preferences. *Proceedings of ICML* (pp. 170–178).

Hastie, T., Tibshirani, R., & Friedman, J. H. (Eds.). (2001). *The elements of statistical learning: Data mining, inference and prediction*. Springer.

Herbrich, R., Graepel, T., & Obermayer, K. (1999). Support vector vector learning for ordinal regression. *Proceedings of ICANN* (pp. 97–102).

Jarvelin, K., & Kekanainen, J. (2000). Ir evaluation methods for retrieving highly relevant documents. *Proceedings of SIGIR* (pp. 41–48).

Lin, Y. (2002). Support vector machines and the bayes rule in classification. *Data Mining and Knowledge Discovery*, 259–275.

Liu, T. Y., Qin, T., Xu, J., Xiong, W. Y., & Li, H. (2007). Letor: Benchmark dataset for research on learning to rank for information retrieval. *Proceedings of SIGIR*.

Marden, J. I. (Ed.). (1995). *Analyzing and modeling rank data*. London: Chapman and Hall.

Nallapati, R. (2004). Discriminative models for information retrieval. *Proceedings of SIGIR* (pp. 64–71).

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 and management*.

Xu, J., & Li, H. (2007). Adarank: a boosting algorithm for information retrieval. *Proceedings of SIGIR* (pp. 391–398).

Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007). A support vector method for optimization average precision. *Proceedings of SIGIR* (pp. 271–278).

Zhang, T. (2004). Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research, 5*, 1225–1251.