Research Article

An Improved LambdaMART Algorithm Based on the Matthew Effect

Jinzhong Li1,2 and Guanjun Liu3

1Department of Computer Science and Technology, College of Electronic and Information Engineering, Jinggangshan University, Jì’ān 343009, China
2Network and Data Security Key Laboratory of Sichuan Province, University of Electronic Science and Technology of China, Chengdu, 610054, China
3Department of Computer Science and Technology, College of Electronic and Information Engineering, Tongji University, Shanghai 201804, China

Correspondence should be addressed to Jinzhong Li; 1210510@tongji.edu.cn

Received 9 August 2018; Revised 10 October 2018; Accepted 16 October 2018; Published 6 November 2018

Copyright © 2018 Jinzhong Li and Guanjun Liu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Matthew effect is a desirable phenomenon for a ranking model in search engines and recommendation systems. However, most of the algorithms of learning to rank (LTR) do not pay attention to Matthew effect. LambdaMART is a well-known LTR algorithm that can be further optimized based on Matthew effect. Inspired by Matthew effect, we distinguish queries with different effectiveness and then assign a higher weight to a query with higher effectiveness. We improve the gradient in the LambdaMART algorithm to optimize the queries with high effectiveness, that is, to highlight the Matthew effect of the produced ranking models. In addition, we propose strategies of evaluating a ranking model and dynamically decreasing the learning rate to both strengthen the Matthew effect of ranking models and improve the effectiveness of ranking models. We use Gini coefficient, mean-variance, quantity statistics, and winning number to measure the performances of the ranking models. Experimental results on multiple benchmark datasets show that the ranking models produced by our improved LambdaMART algorithm can exhibit a stronger Matthew effect and achieve higher effectiveness compared to the original one and other state-of-the-art LTR algorithms.

1. Introduction

Ranking is an important component that directly affects the performances of information retrieval systems such as search engines, recommendation systems, and electronic commerce platforms. For instance, the PageRank algorithm [1, 2] of the Google search engine computes the Web page scores based on a graph inferred from the link structure of the Web. This algorithm assigns a Web page with a higher score if the sum of its backlinks is high. In addition, this algorithm also considers the cast votes of pages. A page has an important weight if it casts votes for itself, and the reason is that it can help enhance the importance of other pages [3]. The PageRank algorithm exhibits the Matthew effect [4], which refers to the phenomenon that the rich get richer and the poor get poorer. This is valuable and reasonable in a search service, since people prefer to click on the links to some high-quality pages from other high-quality ones.

In some application scenarios, for examples of the search of hot words for search engines and the recommendation of popular products for recommendation systems, their ranking methods usually expect a Matthew effect. That is, Matthew effect is a desirable phenomenon for their ranking models, which can make their ranking results more effective and can meet the users’ requirements better. Therefore, it has a practical significance to develop a ranking model with a strong Matthew effect in these ranking scenarios. However, the existing LTR methods do not pay attention to the Matthew effect and thus this becomes the motivation of our work.

In this paper, we improve the LambdaMART [5, 6] algorithm via some new strategies in order to strengthen the Matthew effect. Our improvements can not only strengthen...
the Matthew effect of a ranking model but also enhance the effectiveness of the ranking model. To the best of our knowledge, this is the first attempt to improve the algorithm of learning to rank under considering Matthew effect.

The main contributions of this work are summarized as follows:

1. We present a new function of the gradient in the LambdaMART algorithm that can highlight Matthew effect and prove that the function satisfies the consistency property.
2. We propose two strategies to evaluate the ranking models and dynamically decrease the learning rate, respectively. They can not only strengthen the Matthew effect but also improve the effectiveness of the final ranking model.
3. Experiments on multiple benchmark datasets are done to show the advantages of our improved LambdaMART algorithm compared to the original one and other state-of-the-art LTR algorithms.

The remainder of this paper is organized as follows. Section 2 reviews some related work. Section 3 describes the improvements of the LambdaMART algorithm via the idea of Matthew effect and some strategies. Section 4 presents the evaluation measures of the Matthew effect. Our experimental results are illustrated in Section 5. Section 6 concludes this paper.

2. Related Work

An algorithm of learning to rank (LTR) in the field of information retrieval is to use a machine learning technique to train a ranking model for solving ranking problem. A large number of LTR algorithms have been proposed [5–19]. For example, RankNet [7] is a neural-network-based LTR algorithm that optimizes a cross-entropy cost function using gradient descent algorithm to train a neural network model for ranking. LambdaRank [8] improves RankNet by defining the gradients of a given cost function only at the points of interest. It directly optimizes a cost function of effectiveness measures while avoiding the difficulties of working with nondifferentiable measures of information retrieval. LambdaMART [5, 6], denoted as $\lambda$-MART, improves LambdaRank by using multiple additive regression trees (MART) rather than neural network and thus exhibits better effectiveness of a generated model. $\lambda$-MART [5, 6] is the boosted regression tree version of LambdaRank. It models the gradients using the ranked positions of those documents with relation to a given query and utilizes the gradients to compute the optimal weights for combining the weak learners in a boosting model. $\lambda$-MART treats the force of upward or downward movement of each document unequally by the gradient of the document but treats effectiveness of each query equally, which results in not only a relatively lower ratio of queries with low effectiveness (i.e., poor queries) but also a relatively lower ratio of queries with high effectiveness (i.e., rich queries). Therefore, the $\lambda$-MART algorithm cannot highlight the Matthew effect. To solve this problem, we improve $\lambda$-MART based on the Matthew effect. In other words, queries with different effectiveness are distinguished and those with higher effectiveness are given higher weights, and the differences of moving forces between different document-pairs are enhanced.

There are some studies improving the $\lambda$-MART algorithm. Ganjisaaffar et al. [9] combine bagging and boosting for bagged $\lambda$-MART to achieve the high prediction accuracy and low variance. Asadi et al. [10] explore the topology of tree-based ensembles and modify the $\lambda$-MART algorithm to perform the stage-wise pruning for improving efficiency. Ferov et al. [19] propose a modification of the $\lambda$-MART algorithm, which utilizes oblivious trees instead of standard regression trees in the learning process in order to improve the effectiveness. This paper differs from them. We not only improve the gradient but also propose two strategies: one is to evaluate the current ranking models and the other is to dynamically decrease the learning rate. These improvements can both strengthen the Matthew effect and improve the effectiveness of the $\lambda$-MART algorithm.

There are some studies such as RankBoost [11] and AdaRank [12] that are also based on boosting algorithm. In the two LTR approaches, the document-pairs are reweighted by decreasing the weights of those document-pairs that are correctly ranked and increasing the weights of those document-pairs that are incorrectly ranked. Then, the weak rankers are repeatedly constructed by reweighting the training data, and the learning at the next iteration will focus on the training of a weak ranker that can work on the ranking of those poor queries. Our work also differs from RankBoost [11] and AdaRank [12] because we strengthen the differences of the gradients of different document-pairs at each boosting iteration and highlight the importance of rich queries based on the Matthew effect.

Some approaches of LTR treat the queries in the training dataset equally while ignoring the differences of any two queries in the training process of the ranking models. In general, for two different queries, their effectiveness scores are different in a ranking model. Therefore, they should not be treated equally if we expect that a ranking model can make a query with high effectiveness (i.e., rich query) to have higher effectiveness and make a query with low effectiveness (i.e., poor query) to have lower effectiveness. Notice that there are some studies that consider the differences of queries. Geng et al. [20] propose a K-Nearest Neighbor approach for query-dependent ranking which employs different ranking models for different queries to improve the effectiveness of ranking model. Cai et al. [21] propose two methods of query weighting to measure the query importance for ranking model adaptation. Li et al. [22] make a comprehensive examination of different query-level weighting strategies for two unsupervised ranking frameworks. Their aim of considering the differences of queries is to improve the effectiveness of ranking models, while our aim is to improve not only the effectiveness but also the Matthew effect of ranking models.

Most of the existing LTR algorithms focus on improving the effectiveness of the ranking model, while they rarely pursue the Matthew effect of the ranking model. We know that different queries usually have different effectiveness in a ranking model. Therefore, if a ranking model can guarantee that a rich query has higher effectiveness, then the rich query should be given a greater weight so that it is focused...
on optimization in the training process of the ranking models.

3. The Improvements of the \( \lambda \)-MART Algorithm via the Idea of Matthew Effect and Some Strategies

We first recall \( \lambda \)-MART algorithm and then improve its gradient and propose a strategy of evaluating the current ranking model to characterize the Matthew effect. In addition, we propose a strategy of dynamically decreasing the learning rate. Finally, we analyze the computational complexity.

3.1. The Gradient of \( \lambda \)-MART Algorithm. The basic idea of \( \lambda \)-MART [5, 6] is to train an ensemble of weak learners by using MART and approximate Newton step and linearly combining their predictions into a stronger and more effective learner. \( \lambda \)-MART tunes the gradients of the regression trees based on a gradient-based optimization method. The gradient function is calculated according to the selected evaluation metric of effectiveness, and this metric is optimized directly in the training process of a ranking model.

The key of the \( \lambda \)-MART algorithm is the gradient function \( \lambda \) which is defined as a smooth approximation to the gradient of a target cost with respect to the score of a document. The gradient function quantifies the adjusted direction (up or down) and the force of a “to-be-sorted” document in the next iteration. The two documents of any given document-pair have the equal gradient values but opposite moving directions. The gradient of the document in the positive direction pushes it toward the higher rank position of the sorted list, while the gradient of the document in the negative direction pushes it toward the lower rank position of the sorted list.

\( \lambda \)-MART algorithm optimizes the gradient \( \lambda_i \) of each document \( d_i \) for each query \( q \). For a given query \( q \), if the relevance grade \( r \) between \( d_i \) and \( q \) is higher, and the ranked position of \( d_i \) is more closed to the bottom of the ranked list, then the positive value of \( \lambda_i \) indicates a push toward the top of the ranked list, and a bigger value of \( \lambda_i \) represents a stronger force. Similarly, we can understand the opposite case.

\( \lambda \)-MART algorithm integrates the evaluation criteria of information retrieval (such as NDCG [23]) into the gradient computation. The gradient \( \lambda_i \) for each document \( d_i \) is obtained by the summation of all \( \lambda_{ij} \) over all pairs of \( \langle d_i, d_j \rangle \) that \( d_i \) participates in for query \( q \); that is, \( \lambda_i \) is computed as follows:

\[
\lambda_i = \sum_{j \in \{i,j\} \in \mathcal{I}} \lambda_{ij} - \sum_{j \in \{i,j\} \in \mathcal{I}} \lambda_{ij}
\]

(1)

In (1), \( \lambda_{ij} \) denotes the gradient of the document-pair \( \langle d_i, d_j \rangle \) and is calculated as follows:

\[
\lambda_{ij} = -\frac{\beta}{1 + e^{\beta(s_i - s_j)}} \times |\Delta M_{ij}|
\]

(2)

where \( \beta \) is a shape parameter for the sigmoid function, \( s_i \) and \( s_j \) represent the score assigned to \( d_i \) and \( d_j \) by the ranking model, respectively, and \( \Delta M_{ij} \) represents the change on the effectiveness measure \( M \) by swapping the two documents \( d_i \) and \( d_j \) at the rank positions \( i \) and \( j \) accordingly (while keeping the rank positions of all other documents unchanged). The effectiveness evaluation criteria can be measured by any common information retrieval metrics such as NDCG [23] and ERR [24]. \( \Delta M_{ij} \) is calculated as follows:

\[
\Delta M_{ij} = M_q - M_q^*
\]

(3)

where \( M_q \) denotes the effectiveness of the query \( q \) for a ranked list of all documents with respect to \( q \) and \( M_q^* \) denotes the effectiveness of \( q \) after swapping \( d_i \) and \( d_j \) at the rank positions \( i \) and \( j \) for the ranked list. \( \lambda \)-MART algorithm incorporates an effectiveness evaluation criterion into the gradient function in the training process of the ranking models, and thus it optimizes both the loss function and the effectiveness evaluation criterion.

3.2. Improvement of the Gradient Function. In the \( \lambda \)-MART algorithm, we treat all queries unequally in the training process of the ranking models. We discriminate the effectiveness score of each query and assign different weights to different queries. In order to highlight the gradients of rich queries and enhance the effectiveness of rich queries, the weights of rich queries should be given a higher value based on the idea of the Matthew effect. Therefore, we define the weight of a query as the \( (t - 1) \)-th power of the original effectiveness score of the query, where \( t \) is an integer and is bigger than 1. The greater the value of \( t \), the greater the difference between the assigned weights of the rich query and the poor query. Therefore, our method can reflect the importance of rich query in the ranking model, and thus the Matthew effect of the ranking model is highlighted to a certain extent. Thus, the new effectiveness score of a query is defined as the \( t \)-th power of the original effectiveness score of the query, and the original effectiveness \( M \) of the gradient is replaced by \( M^t \) to optimize the rich ranking model as a new objective. Therefore, \( \Delta M_{ij} \) is replaced by \( \Delta M_{ij}^t \) that is defined as follows:

\[
\Delta M_{ij}^t = (M_q^t)^3 - (M_q^*)^3
\]

(4)

\( \Delta M_{ij}^* \) denotes the difference of the \( t \)-th power value of the effectiveness of \( q \) after swapping \( d_i \) and \( d_j \) at the rank positions \( i \) and \( j \) (while fixing the rank positions of all other documents). For example, when \( t = 2 \), we have that \( \Delta M_{ij}^* = (M_q^t)^2 - (M_q^*)^2 \), and when \( t = 3 \), we have that \( \Delta M_{ij}^* = (M_q^t)^3 - (M_q^*)^3 \). When the hyperparameter \( t \) is bigger than 1, the power operation enhances Matthew effect. Because the ranking effectiveness measure \( M \) is changed into \( M^t \), the difference of effectiveness \( M^t \) becomes more prominent with the increase of \( t \), and then the effectiveness of Matthew effect is more obvious in the learning process.

Meanwhile, we incorporate the Gini coefficient into the gradient in order to highlight the difference in effectiveness for different queries in the ranking model. Consequently, \( \lambda_{ij}^{new} \) is defined as follows:

\[
\lambda_{ij}^{new} = -\frac{\beta \times k \times e^{Gini}}{1 + e^{\beta(s_i - s_j)}} \times |\Delta M_{ij}^*|
\]

(5)
where the positive constant $k$ is used to adjust the value of $\lambda_{ij}$ in order to make $\lambda^\text{new}_{ij}$ and $\lambda_{ij}$ in the same order of magnitude. Gini measures the degree of difference in effectiveness among all queries and will be introduced in the next section. The different document-pair $(d_i, d_j)$ for each query is optimized according to the new gradient $\lambda^\text{new}_{ij}$ in the learning process of ranking models, which strengthens the differences of upward or downward ranking force among the document-pairs for different queries in the next iteration and thus enhances the optimization of rich queries and weakens the optimization of poor queries and highlights the effectiveness of Matthew effect.

The gradient function of $\lambda$-MART is changed and we must prove its consistency, which guarantees that our improved $\lambda$-MART algorithm can be used to produce the ranking models. Now, we prove that the new objective $M^T$ satisfies a desirable consistency property proposed in [5, 6]: when swapping the ranked positions of two documents $d_i$ and $d_j$, the optimization objective should be increased. In other words, for any document-pairs, the pairwise swap between the correctly ranked documents $d_i$ and $d_j$ for the same query $q$ must lead to a decrease of $M^T$, and the pairwise swap between the incorrectly ranked documents $d_i$ and $d_j$ for the same query $q$ must lead to an increase of $M^T$.

**Theorem 1.** The new optimization objective function $M^T$ satisfies the consistency property.

**Proof.** According to the previous analysis, we know that $M_q$ is the effectiveness of $q$ for a ranked list of all documents with respect to $q$, $M_q^*$ is the effectiveness of $q$ after swapping $d_i$ and $d_j$, and $M_{ij}^*$ is the changed effectiveness on the new optimization objective $M^T$ by swapping $d_i$ and $d_j$ at the rank positions $i$ and $j$ while leaving all other ranks unchanged in the ranked list. Furthermore, $\Delta M_j^*$ can be expanded according to $n$ variance formula in factorization; namely, $a^n - b^n = (a - b)(a^{n-1} + a^{n-2}b + \cdots + ab^{n-2} + b^{n-1})$. Therefore, (4) becomes

$$
\Delta M^*_{ij} = (M_q - M_q^*) \times \left[(M_q)^{t-1} + (M_q)\times (M_q)^{t-2} \times M_q^* + \cdots + M_q \times (M_q^*)^{t-2} + (M_q^*)^{-1}\right] \tag{6}
$$

Let $P(t) = M_q^{(t-1)} + M_q^{(t-2)} \times M_q^* + \cdots + M_q \times (M_q^*)^{(t-2)} + (M_q^*)^{(t-1)}$. Then (6) is rewritten according to (3) as follows:

$$
\Delta M^*_{ij} = \Delta M_{ij} \times P(t) \tag{7}
$$

To demonstrate that $M^T$ satisfies the consistency property, we only need to show that the change in $\Delta M^*_{ij}$ tends to move in the same direction with the change in $\Delta M_{ij}$ of the original optimization objective $M$ if the positions of $d_i$ and $d_j$ are swapped. This is because the original optimization objective $M$ satisfies the consistency property and can be directly optimized by the $\lambda$-MART algorithm.

We know that no matter which kind of commonly used measure of effectiveness $M_q$ and $M_q^*$ is taken in the information retrieval field, their values are both greater than or equal to 0; that is, $M_q \geq 0$ and $M_q^* \geq 0$. Therefore, $P(t) \geq 0$ when $t > 1$. Hence, the directions of changes of $\Delta M^*_{ij}$ and $\Delta M_{ij}$ are the same according to (7). The bigger value of $M$ indicates that it has better effectiveness in the original $\lambda$-MART algorithm, and the bigger value of $M^T$ indicates that it has also better effectiveness in our improved $\lambda$-MART algorithm. Therefore, the optimization directions of $M$ and $M^T$ are also the same. Therefore, the new optimization objective $M^T$ with the improved gradient satisfies the required consistency property, and it can also be optimized by using the improved $\lambda$-MART algorithm.

### 3.3. A Strategy of Evaluating the Current Ranking Model

In the traditional algorithms of LTR, the strategy of evaluating the current ranking models is directly calculated by the effectiveness measure $M$, such as NDCG [23] and ERR [24], at each learning iteration, which is difficult to reflect the Matthew effect of ranking models. To reflect the Matthew effect, we propose a strategy of evaluating the current ranking models by integrating the Gini coefficient and the new optimization objective $M^T$. The strategy is that we use $k \times e^{Gini} \times M^T$ to compute the scores of the current ranking model on the training and validation datasets and evaluate the total score of the current ranking model to measure its performance in each iteration of training ranking models. After the end of training, an optimal ranking model is chosen by using the training and validation datasets. The ranking model with the maximum value of the sum of $k \times e^{Gini} \times M^T$ on the training and validation datasets is selected as our final one.

### 3.4. A Strategy of Dynamically Decreasing the Learning Rate

How to choose a learning rate is a critical issue. The fixed learning rate may lead to a slow convergence for a gradient decent algorithm [25]. Generally, the value of learning rate (i.e., shrinkage coefficient) has an influence on the speed of convergence for some LTR algorithms with shrinkage coefficient, which directly affects their efficiency and effectiveness to a certain extent. In the $\lambda$-MART algorithm, the learning rate $\eta$ is unchanged during the gradient boosting process; that is, the learning rate is a constant over training iterations. This fixed learning rate is not conducive to the acceleration of convergence when the optimization result has not yet reached the convergence in a given number of iterations. Therefore, the performance of ranking model is decreased at the case of the fixed learning rate if the iteration is not sufficient. In order to overcome the shortcomings, we propose a strategy of dynamically decreasing learning rate to update the weights. The fixed learning rate $\eta$ in the $\lambda$-MART algorithm is replaced by a dynamical learning rate $\eta^*_\text{iter}$:

$$
\eta^*_\text{iter} = \eta_{max} - (\eta_{max} - \eta_{min}) \times \frac{it\text{er}}{nt\text{rees} - 1} \tag{8}
$$

where $\eta_{max}$ and $\eta_{min}$ denote the maximum and minimum learning rates, respectively, $nt\text{rees}$ represents the total
number of trees or boosting iterations, and \( \text{iter} \in [0, n_{\text{trees}} - 1] \) represents the current number of iterations or the \( \text{iter} \)-th tree. \( \eta_{\text{iter}} \) represents the learning rate at iteration \( \text{iter} \)-th. From the definition of \( \eta_{\text{iter}} \), we can know that this strategy starts the training process with the given maximum learning rate \( \eta_{\text{max}} \), and the learning rate is being shrunk gradually, while these trees are being established. When the learning rate decreases to the given minimum value \( \eta_{\min} \), the training process terminates.

\( \eta_{\text{iter}} \) is asymptotically decreased throughout the whole training. It is beneficial to speed up the convergence and improves the effectiveness of the ranking model. Because the produced ranking models are fine in the early stages of the training process, using a relatively high learning rate to train ranking models can speed up the convergence and avoid a local minima. A large value of the shrinkage coefficient can speed up the initial improvement when the algorithm does not converge. Because the produced ranking models are fine in the late stage of the training process, using a relatively low learning rate to refine ranking models can improve a local search and help it converge to a better optimizing point. A small value of the shrinkage coefficient can lead to a better performance when the algorithm is gradually close to convergence.

3.5. Complexity Analysis. In the original \( \lambda \)-MART algorithm, the time complexity of the algorithm is \( O(m \cdot (n_{\text{trees}} + L)) \), where \( m \) represents the number of training samples, \( n_{\text{trees}} \) represents the total number of trees, and \( L \) represents the number of leaves per tree. In our improved \( \lambda \)-MART algorithm, we have improved the gradient and proposed strategies of evaluating a ranking model and dynamically decreasing the learning rate, but these improvements only modify the related computing methods, which do not affect the time complexity of the algorithm; that is, the time complexity of our improved \( \lambda \)-MART algorithm is also \( O(m \cdot (n_{\text{trees}} + L)) \).

In addition, if no performance gain on the validation data is observed in the set number of rounds during the process of learning, then the training process terminates in the original \( \lambda \)-MART algorithm. Our strategy of dynamically decreasing the learning rate may lead to an early stop with a greater probability and then accelerate the convergence of the algorithm. Therefore, the total number of iterations or computational time cost of our improved \( \lambda \)-MART algorithm is reduced in most cases, as shown in Figure 12.

4. Evaluation Measures of Matthew Effect

A ranking model is richer than another model if (1) the former has both more rich queries and more poor queries than the latter, or (2) the distribution of the effectiveness of queries of the former is more discrete than the latter. A richer model is of a stronger Matthew effect. In order to measure the performances of the ranking model yielded by our improved method, we introduce the following utility metrics to characterize the Matthew effect of ranking models from different perspectives. Based on the effectiveness evaluation criteria for information retrieval, we introduce Gini coefficient, mean-variance, and quantity statistics to measure the performances.

4.1. Gini Coefficient. The Gini coefficient [26, 27] is a measure of the statistical dispersion that represents the income distribution of the residents in a nation. It was developed by the Italian statistician and sociologist Corrado Gini in 1912. The Gini coefficient is usually defined mathematically based on the Lorenz curve and is most easily calculated from unordered size data as the “relative mean difference” divided by the mean size \( \mu \). The “relative mean difference” is the mean of the difference between every possible pair of individuals. The Gini coefficient is calculated using the following formula:

\[
G = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} |x_i - x_j|}{2m\mu}
\]

where \( x_i \) denotes the income of individual \( i \), \( |x_i - x_j| \) denotes the absolute value of the difference between \( x_i \) and \( x_j \), \( \mu \) denotes the mean value of all individuals’ incomes, and \( n \) denotes the total number of individuals. The smaller the inequality of income, the smaller the value of the Gini coefficient, and vice versa.

Matthew effect is reflected by using Gini coefficients for the measurement in many economic areas. Gini coefficient can capture the Matthew Effect, which can be used to measure the performance of the ranking model. If we make an analogy with the distribution of the national income in the field of finance, the query in the LTR task resembles the individual in the distribution of the national income, and the effectiveness of the query resembles the income of the individual. Therefore, the Gini coefficient of LTR can be defined as follows:

\[
Gini = \frac{\sum_{i=1}^{q} \sum_{j=1}^{q} |M_{e_i} - M_{e_j}|}{2q \sum_{i=1}^{q} M_{e_i}}
\]

where \( q_i \in Q \) denotes the \( i \)-th query, \( |Q| \) represents the total number of queries in query set \( Q \), and \( M_{e_i} \) represents the effectiveness of \( q_i \). Gini is used to measure the degree of difference in effectiveness among all the queries in a ranking model and reflects the Matthew effect. If the value of Gini obtained by a ranking model is larger, then the ranking model has a stronger Matthew effect.

4.2. Mean-Variance. Mean is the average value of all random variables. In order to observe the effectiveness of a ranking model, the mean \( \mu \) of a ranking model is defined as follows:

\[
\mu = \frac{1}{|Q|} \sum_{q \in Q} M_q
\]

For a ranking model, \( \mu \) measures the average effectiveness (such as NDCG [23] and ERR [24]) of all queries in the query set; that is, it refers to the average effectiveness of the ranking model. The greater \( \mu \), the better the average effectiveness of the ranking model, and vice versa.

Variance is used to measure the degree of deviation between a set of random variables and their mean, and thus
it is an important and common metric for calculating the
discrete trend. The greater the variance, the greater the degree
of deviation, and vice versa.
In a ranking model, some queries are of high effectiveness
but some queries are of low effectiveness. Therefore, in order
to observe the degrees of their deviation, we divide
the variance into the upside semivariance and the downside
semivariance. The upside semivariance $V_{up}$ and the down-
side semivariance $V_{down}$ of a ranking model are defined as
follows:

$$V_{up} = \frac{1}{|Q'|} \sum_{q \in Q'} (M_q - \mu)^2$$

(12)

$$V_{down} = \frac{1}{|Q'|} \sum_{q \in Q'} (M_q - \mu)^2$$

(13)

where $Q'$ and $Q$ denote the set of queries with above-mean
effectiveness and below-mean effectiveness in the query set
$Q$, respectively, and $|Q'|$ and $|Q|$ denote the number of sets
of queries $Q'$ and $Q$, respectively.

For a ranking model, $V_{up}$ measures the discrete degree of
effectiveness of the queries that are over the value of $\mu$, and
$V_{down}$ measures the discrete degree of effectiveness of the
queries that are under the value of $\mu$. The Matthew effect of
a ranking model can be reflected by its $V_{up}$ and $V_{down}$. The
higher the values of both $V_{up}$ and $V_{down}$ of a ranking model,
the stronger the Matthew effect of the ranking model, and vice versa.

4.3. Quantity Statistics. Generally, for most of the effective-
ness metrics in information retrieval, the range of their values
is between 0 and 1, such as $NDCG$ [23] and $ERR$ [24]. To
compute the effectiveness distribution of all queries in a query
set, the range of values of the effectiveness is divided into
4 intervals: $[0, 0.1]$, $(0.1, 0.4]$, $(0.4, 0.7]$, and $(0.7, 1.0]$. We
compute the number of queries in these different intervals
according to the effectiveness values of the queries in a given
query set for different ranking models, and the purpose is
to evaluate the strengths of their Matthew effects. We use
an array $count$ to express the quantity statistics of different
intervals for the effectiveness of queries, and $count$ is defined as
follows:

$$count[i] = \begin{cases} 
\sum_{M_q \in [0,0.1]} & 1 \quad i = 0 \\
\sum_{M_q \in (0.1,0.4]} & 1 \quad i = 1 \\
\sum_{M_q \in (0.4,0.7]} & 1 \quad i = 2 \\
\sum_{M_q \in (0.7,1.0]} & 1 \quad i = 3 
\end{cases}$$

(14)

If the values of $count[0]$ and $count[3]$ of a ranking model
are larger, then the ranking model has a stronger Matthew
effect.

5. Experiments

In order to illustrate the advantages of our improved
$\lambda$-MART algorithm (named as $Matthew-\lambda$-MART), we
implement the algorithm based on the open-source RankLib
library of LTR algorithms developed by Van Dang et al.
(http://sourceforge.net/p/lemur/code/HEAD/tree/RankLib/
trunk/). We conduct extensive experiments to evaluate and
compare the ranking performance of $Matthew-\lambda$-MART
with three well-known LTR algorithms, $\lambda$-MART [5, 6],
RankBoost [11], and AdaRank [12], over three standard
benchmark LTR datasets (including one large scale LTR
dataset: MSLR-WEBS30K (http://research.microsoft.com/en-
us/projects/mslr/download.aspx), with 30000 queries and a
total of 136 features in the Microsoft LTR datasets and two
small LTR datasets, MQ2007 (http://research.microsoft.com/en-
us/um/beijing/projects/letor/letor4dataset.aspx), with
1692 queries and a total of 46 features in the LETOR
4.0 datasets, and WCL2R with 79 queries and a total of 29
features released by Alcântara et al. [28], which includes click-
through features. In order to make the order of magnitude
of the gradient’s value and the learning rate in $Matthew-
\lambda$-MART be the same as $\lambda$-MART’s, we set parameters
for $Matthew-\lambda$-MART as follows: $t = 2$ or 3, $k = 100$,
$k_{\text{max}} = 0.3$, and $k_{\text{min}} = 0.1$, where $k$, $t$, $k_{\text{max}}$, and
$k_{\text{min}}$ are introduced in Section 3. We denote $Matthew-\lambda$-MART
as $Matthew2-\lambda$-MART and $Matthew3-\lambda$-MART when $t = 2$
and $t = 3$, respectively. The values of other parameters of
$Matthew-\lambda$-MART are the same as $\lambda$-MART’s. We use
$NDCG$ [23] as the effectiveness optimization criterion
during the training process for each LTR algorithm and each
dataset. Based on the produced ranking models, we calculate
the effectiveness measures $\mu$ (including $NDCG$ and $ERR$)
and our introduced Matthew effect measures (including $Gini$,
$V_{up}$, $V_{down}$, $count[0]$, and $count[3]$) according to (10),
(12), (13), and (14)). These results are shown in Figures 1–9.
All experimental results are summed on these five folds for
the test dataset. Meanwhile, the comprehensive evaluations
are conducted and the performance of the strategy of
dynamically decreasing the learning rate is also evaluated.
These results are shown in Figures 10–12.

5.1. Experiments on Large Dataset: MSLR-WEBS30K. Figures
1–6 provide the total results of five folds on MSLR-WEBS30K
dataset in terms of $\mu$, $Gini$, $V_{up}$, $V_{down}$, $count[0]$, and $count[3]$ for
different LTR algorithms, respectively.
From mean $\mu$ (including $NDCG@20$, $NDCG@10$,
$NDCG@5$, $ERR@20$, $ERR@10$, and $ERR@5$) in Figure 1, we
can see that the values of $\mu$ obtained by $Matthew2$-$\lambda$-MART
are bigger than $Matthew3$-$\lambda$-MART, $\lambda$-MART, RankBoost,
and AdaRank. These results show that $Matthew2$-$\lambda$-MART
obtains the highest and best effectiveness. There are two
main reasons. One is that $Matthew-\lambda$-MART incorporates
the effectiveness evaluation measures $M'$ into the gradient
and combines the overall scores $k \times e^{Gini} \times M'$ on training
and validation datasets to choose a ranking model as the final
ranking model. The other one is that $Matthew-\lambda$-MART uses the strategy of dynamically decreasing a learning rate to
obtain an optimal tree. The main reason why the effectiveness of Matthew2-$\lambda$-MART outperforms Matthew3-$\lambda$-MART is that the enhancement of the Matthew effect of a ranking model usually reduces its effectiveness of the ranking model, while Matthew3-$\lambda$-MART with $t = 3$ is more able to strengthen the differences of the gradients and the differences of the effectiveness of queries than Matthew2-$\lambda$-MART with $t = 2$. Therefore, Matthew3-$\lambda$-MART strengthens the optimization of Matthew effect more than Matthew2-$\lambda$-MART does.

From Gini coefficient $Gini$ in Figure 2, Matthew2-$\lambda$-MART and Matthew3-$\lambda$-MART both obtain a bigger value than $\lambda$-MART. These results show that the effectiveness across different individual queries in Matthew-$\lambda$-MART has a greater difference and thus indicates that the ranking model trained by Matthew-$\lambda$-MART has a stronger Matthew effect than $\lambda$-MART. Figure 2 also indicates that the values of $Gini$ of Matthew3-$\lambda$-MART are bigger than the one of Matthew2-$\lambda$-MART. The main reason is that the value of $t$ in Matthew3-$\lambda$-MART is bigger than that in Matthew2-$\lambda$-MART. The bigger the value of $t$, the greater the difference
between the assigned weights of the rich query and the poor query. Therefore, the rich query can get more optimization than the poor query in the process of learning a ranking model. Therefore, Matthew3-\(\lambda\)-MART strengthens the optimization of Matthew effect compared to Matthew2-\(\lambda\)-MART, and the Matthew effect of Matthew3-\(\lambda\)-MART is highlighted to a certain extent.

From Figures 3 and 4, the values of variances \(V_{up}\) and \(V_{down}\) generated by Matthew2-\(\lambda\)-MART and Matthew3-\(\lambda\)-MART are bigger than \(\lambda\)-MART, RankBoost, and AdaRank in most of cases (except that the values of \(V_{up}\) generated by Matthew3-\(\lambda\)-MART are smaller than RankBoost and AdaRank in terms of NDCG@20, NDCG@10, and NDCG@5, and the value of \(V_{up}\) generated by Matthew3-\(\lambda\)-MART is smaller than \(\lambda\)-MART in terms of ERR@5). These results demonstrate that the effectiveness across different individual queries in Matthew-\(\lambda\)-MART has also a greater difference generally. Therefore, the ranking model trained by Matthew-\(\lambda\)-MART usually exhibits a stronger Matthew effect than \(\lambda\)-MART, RankBoost, and AdaRank.

From Figures 5 and 6, the values of \(\text{count}[0]\) with regard to rich queries and \(\text{count}[3]\) with regard to poor queries obtained by Matthew2-\(\lambda\)-MART and Matthew3-\(\lambda\)-MART are bigger than \(\lambda\)-MART, RankBoost, and AdaRank in most of cases. These results show that Matthew-\(\lambda\)-MART leads
to a relative polarization, thereby further exhibiting a stronger Matthew effect than \(\lambda\)-MART in general.

Since \textit{Matthew-\(\lambda\)-MART} highlights the corresponding differences of upward or downward ranking force between documents with regard to rich queries with high effectiveness and documents with regard to poor queries with low effectiveness, it strengthens the optimization of rich queries and weakens the optimization of poor queries in the training process of the ranking models. Consequently, more forces are used to optimize the ranked positions of the documents in the rich queries and less for the poor queries. Finally, \textit{Matthew-\(\lambda\)-MART} increases the degree of difference or dispersion in the effectiveness across all the queries. Therefore, the ranking model trained by \textit{Matthew-\(\lambda\)-MART} shows a stronger Matthew effect than \(\lambda\)-MART.

5.2. Experiments on Small Datasets: WCL2R and MQ2007. Wu et.al. [5] have pointed out that smaller trees and shrinkage may be used if the training dataset is smaller. WCL2R with 79 queries and MQ2007 with 1692 queries are small datasets, so we set parameters for \textit{Matthew-\(\lambda\)-MART} as follows: \(ntrees = 200\), \(\eta_{\text{max}} = 0.1\), and \(\eta_{\text{min}} = 0.05\), where \(ntrees\) represents the number of trees. We use NDCG@5 as the effectiveness optimization criterion during the training process on WCL2R.

![Figure 9: count[0] and count[3] of each algorithm on WCL2R and MQ2007 datasets.](image)

![Figure 10: \(WN_p\) of each algorithm on effectiveness and Matthew effect.](image)

![Figure 11: \(\mu\) for different strategies of learning rate in \(\lambda\)-MART.](image)

![Figure 12: The total number of iterations for different strategies of learning rate in \(\lambda\)-MART.](image)
and MQ2007 for these LTR algorithms. Figures 7–9 provide the overall results of five folds on the two datasets in terms of measures \( \mu, \text{Gini}, V_{up}, V_{down}, \text{count}[0], \) and \( \text{count}[3] \) for these LTR algorithms, respectively. These results also show that the Matthew-\( \lambda \)-MART algorithm performs better than others in most cases.

### 5.3. The Comprehensive Evaluation: Winning Number

We use the winning number \( W_{N_p} \) to comprehensively evaluate the effectiveness and Matthew effect of LTR algorithms. \( W_{N_p} \) is introduced by Liu [29], which counts the times a method beats other ones over all datasets. We extend \( W_{N_p} \) to a cross-measure for each LTR algorithm, which counts the times a method beats other ones over all measures and datasets. We redefine \( W_{N_p} \) as follows:

\[
W_{N_p} = \sum_{p=1}^{R} \sum_{k=1}^{S} \sum_{i=1}^{N} I \left( \text{Meas}_j^i(i) > \text{Meas}_k^i(i) \right)
\]

(15)

where \( N \) is the number of the compared methods, \( P \) and \( k \) are the indices of the compared methods, \( R \) is the number of LTR datasets, \( i \) is the index of an LTR dataset, \( S \) is the number of measures, \( j \) is the index of a measure, \( \text{Meas}_j^i(i) \) is the performance of the \( k \)-th method on the \( i \)-th dataset in terms of the \( j \)-th measure, and \( I(\cdot) \) is an indicator function such that

\[
I \left( \text{Meas}_j^i(i) > \text{Meas}_k^i(i) \right) = \begin{cases} 
1 & \text{if} \text{Meas}_j^i(i) > \text{Meas}_k^i(i), \\
0 & \text{otherwise}.
\end{cases}
\]

\( W_{N_p} \) is a comprehensive evaluation of ranking performance. The larger \( W_{N_p} \) is, the better the \( P \)-th method performs.

Figure 10 shows \( W_{N_p} \) for these LTR algorithms on the three datasets in terms of the effectiveness (i.e., \( \mu \), including NDCG@20, NDCG@5, ERR@20, ERR@10, and ERR@5) and Matthew effect (including Gini, \( V_{up}, V_{down}, \text{count}[0], \) and \( \text{count}[3] \)). From this figure, we can observe the following facts:

1. In terms of \( W_{N_p} \) of effectiveness, Matthew2-\( \lambda \)-MART and Matthew3-\( \lambda \)-MART are better than \( \lambda \)-MART, RankBoost, and AdaRank, and \( \lambda \)-MART is better than RankBoost and AdaRank.
2. In terms of \( W_{N_p} \) of Matthew effect, Matthew2-\( \lambda \)-MART and Matthew3-\( \lambda \)-MART are also better than \( \lambda \)-MART, RankBoost, and AdaRank, but \( \lambda \)-MART is worse than RankBoost and AdaRank.

### 5.4. Performance of the Strategy of Dynamically Decreasing the Learning Rate

We conduct experiments on the MSLR-WEBS30K dataset to evaluate the advantage of the strategy of dynamically decreasing the learning rate; that is, the effectiveness and convergence efficiency are indeed enhanced when the dynamically decreasing learning rate replaces a fixed one. The fixed learning rate \( \eta_{\text{max}} = 0.1 \) in the \( \lambda \)-MART algorithm is replaced with the dynamical learning rate \( \eta_{\text{dert}} \) in (8). For the large-scale dataset, in order to keep the consistency of values of parameters of the \( \lambda \)-MART algorithm with the fixed learning rate, we set parameters of the \( \lambda \)-MART algorithm with the dynamically decreasing learning rate as follows: \( \eta_{\text{max}} = 0.3, \eta_{\text{min}} = 0.1 \), and other parameters are unchanged. We use ERR and NDCG as the effectiveness optimization criteria during the training process of the ranking models, respectively. Figure 11 reports the overall results of \( \mu \) including ERR@5, ERR@10, ERR@20, NDCG@5, NDCG@10, and NDCG@20 on the five folds, respectively. Figure 12 reports the total number of iterations accordingly. Obviously, the strategy of dynamically decreasing the learning rate not only improves the effectiveness of the ranking models but also enhances the convergence efficiency of our improved \( \lambda \)-MART.

In summary, experimental results on three benchmark datasets demonstrate that Matthew-\( \lambda \)-MART can achieve a better performance on both Matthew effect and effectiveness compared with the state-of-the-art LTR algorithms. In addition, the experiments prove that our proposed strategy of dynamically decreasing the learning rate is better than the fixed learning rate in the \( \lambda \)-MART algorithm.

### 6. Conclusion

In this paper, we integrate the idea of Matthew effect into the algorithm of LTR and present a new function of the gradient in the \( \lambda \)-MART algorithm which can highlight Matthew effect and prove that the function satisfies the consistency property. In addition, we propose two strategies to evaluate the ranking models and dynamically decrease the learning rate, respectively. They can not only strengthen Matthew effect but also improve the effectiveness of the produced ranking model. Experiments do prove that our improvements work.

Some interesting issues can be explored in the future research.

1. Other learning rates such as optimal learning rate and adaptive learning rate can be considered in LTR.
2. It is obvious that different information has different popularity in different time periods, which leads to the users’ queries always changing their popularity over time. Some queries (hot queries) gain a huge popularity with numerous searches, while some queries (cold queries) are just the opposite. We plan to integrate the hot and cold queries into an LTR algorithm.
3. The multiobjective evolutionary algorithms and reinforcement learning techniques can be incorporated into LTR to develop a multiobjective and online LTR algorithm.

### Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

### Conflicts of Interest

The authors declare that they have no conflicts of interest.
Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants nos. 61762052 and 61572360, in part by the Natural Science Foundation of Jiangxi Province of China under Grant no. 2017BAB202010, in part by the Opening Foundation of Network and Data Security Key Laboratory of Sichuan Province under Grant no. NDSMS201602, and in part by the Doctoral Scientific Research Startup Foundation of Jinggangshan University under Grant no. JZB1804.

References


Submit your manuscripts at
www.hindawi.com