International Journal of Innovative Computing, Information and Control
Volume **14**, Number **4**, August **2018**

ICIC International ©2018 ISSN 1349-4198
pp. 1207–1221

A MINI-BATCH STOCHASTIC GRADIENT METHOD FOR SPARSE LEARNING TO RANK

Fan Cheng[1,2], Dongliang Wang[2], Lei Zhang[1,2], Yansen Su[1,2]
Jianfeng Qiu[1,2,*] and Yi Suo[2]

[1]Key Laboratory of Intelligent Computing and Signal Processing, Ministry of Education
[2]School of Computer
Anhui University
No. 3, Feixi Road, Hefei 230039, P. R. China
[*]Corresponding author: qiujianf0216@163.com

Received November 2017; revised March 2018

Abstract. *Learning to rank has attracted increasing attention in recent years. Compared with the existing works, which mainly focus on the dense model, it is more challenging to design the ranking model with sparsity, especially when data is of large scale. To address the issue, in this paper, we present a mini-batch stochastic gradient method for sparse ranking. The algorithm begins with formulating sparse learning to rank as a mini-batch based convex optimization problem with L1 regularization. Then for the problem that simple adding L1 term does not necessarily induce the sparsity, FTRL (Follow-The-Regularized-Leader) method is adopted for inner optimization, which can obtain good solution with high sparsity. A feature-wise learning rates updating strategy is also suggested, and with this strategy, the performance of the proposed algorithm can be further improved. Empirical studies on LETOR data sets demonstrate the significant advantage of the proposed algorithm in comparison with several state-of-the-art ranking methods.*

Keywords: Learning to rank, Sparse model, FTRL, Mini-batch

1. Introduction.

Learning to rank has attracted the focus of many machine learning researchers in the last decade because of its growing applications in the areas like document retrieval [1, 2], recommendation systems [3, 4], and disease diagnosis [5]. Rank learning, when applied to document retrieval, is a task as follows. In training, a ranking model is constructed with data consisting of queries, their corresponding retrieved documents, and relevance levels given by humans. In testing, given a new query, the documents are sorted by using this trained ranking model. Due to its wide usages, various ranking algorithms have been developed, which can be mainly divided into three categories. The pointwise methods transform ranking into classification or regression by learning the numeric rank value of documents as an absolute quantity [6, 7, 8, 9]. The pairwise methods, consider the pairs of documents as independent variables and learn classification or regression models to correctly order the training pairs [10, 11, 12, 13, 14, 15]. While the listwise methods solve the ranking problem in a straightforward fashion, which learn the models by treating each ranking list of documents for a query as a training instance [16, 17, 18, 19].

Those studies can achieve promising performances and obtain accurate models; however, in many ranking applications, such as computer vision [20] and disease diagnosis [5], the number of features in training examples is large, which may cause great challenge to the existing ranking methods. First, along with the number of available ranking features mount up, ranking models become more complicated and worse interpretable. Secondly,

DOI: 10.24507/ijicic.14.04.1207

as more and more features are incorporated into algorithms, some features may be noisy or redundant, which results in poor generalization performance. Lastly, too many features will make the learning procedure and the prediction procedure have more computational cost. In machine learning area, sparse learning is an effective way to solve the problem. Thus recently, several algorithms have been proposed to tackle the issue of learning sparse models for ranking. For example, Sun et al. were the first to propose sparse learning to rank algorithm, termed RSRank, which reduced ranking to an important weighted pairwise classification with the L1 regularization [21]. Based on the theory of Fenchel duality, Lai et al. developed an effective primal dual approach (FenchelRank) to solve the sparse model for ranking [22]. Following this work, Lai et al. further presented a novel efficient sparse algorithm, which is named SparseRank. Experimental results indicated that SparseRank can obtain accurate model with high sparsity [23]. Different from the works above, which all used L1 regularization, Laporte et al. got sparse ranking model by adopting a non-convex l_p $(0 < p < 1)$ regularization [24].

Those works are worthwhile in the endeavor of achieving sparse ranking models with competitive generalization performance; nevertheless, all of them are based on batch learning, which means their computational costs are becoming higher with the number of data getting larger. To tackle the issue, in this paper, we present the first work on developing a mini-batch stochastic gradient method for sparse ranking, and the main contributions of our work can be summarized as follows.

1) Based on the query level of learning to rank, we transform the sparse ranking into a mini-batch based convex optimization problem with L1 regularization. For the problem that simple adding L1 cannot necessarily lead to sparsity, FTRL technique is adopted as the inner optimizer, which can obtain effective sparse model with comparable results.
2) A feature-wise learning rates updating strategy is also suggested, which exploits the knowledge of historical gradients to perform more informative learning. Specifically, it assigns the frequently occurring features with low learning rates while the rarely occurring features are given high learning rates. With this strategy, the performance of the proposed algorithm can be further improved.
3) We empirically verify our algorithm on the LETOR data sets, and experimental results demonstrate that when compared with other state-of-the-art ranking methods, the algorithm we proposed has superior performance in terms of accuracy, sparsity and computational efficiency.

The remainder of the paper is organized as follows. In Section 2, the related work is presented. Section 3 gives the details of our proposed algorithm and the empirical results on the benchmark data sets are reported in Section 4. Section 5 concludes the paper and discusses the future work.

2. **Realted Work.**

2.1. **Learning to rank.**

The process of learning to rank can be described as follows. Assuming that we have a collection of m queries for training, denoted by $\left\{q_1, q_2, \ldots, q_m\right\} \in Q$. For each query q_k $(1 \leq k \leq m)$, we have a collection of n_k documents $\left\{d_k^1, d_k^2, \ldots, d_k^{n_k}\right\}$, whose relevance to q_k is given by a vector $y_k = \left\{y_k^1, y_k^2, \ldots, y_k^{n_k}\right\}$, and each relevance label $y_k^j \in \{r_1, r_2, \ldots, r_l\}$, with $r_l \succ r_{l-1} \succ \cdots \succ r_1$. The goal is to learn from these examples a ranking function which, given a new query q, can rank the documents associated with the query such that more relevant documents are ranked higher than less relevant ones.

More formally, we denote $x_k^j \equiv \phi\left(q_k, d_k^j\right) \in R^d$ as a query-document feature, which maps the j-th query-document pair to a d-dimensional feature vector. The ranker receives labeled samples of the form $S = \{s_1, s_2, \ldots, s_m\}$, where $s_k = \{\underbrace{(x_k^1, y_k^1)}_{s_k^1}, \ldots, \underbrace{(x_k^{n_k}, y_k^{n_k})}_{s_k^{n_k}}\}$ is the training example associated with the k-th query. The aim of learning to rank is to learn a function $f\colon R^d \to R$ that ranks a document d^i associated with a future query q higher than a document d^j, if $y^i \succ y^j$. Following the common practice in learning to rank, in this paper, we are interested in linear ranking function given by:

$$f\left(s_k^j\right) = w^T \cdot \varphi\left(\phi\left(q_k, d_k^j\right), y_k^j\right) \equiv w^T \cdot \varphi\left(x_k^j, y_k^j\right) \tag{1}$$

where φ represents a mapping function from input to output.

To obtain accurate ranking function, various algorithms have been proposed, which can be grouped into three categories: pointwise methods, pairwise methods, and listwise methods. Among them, the latter two are more popular, especially pairwise methods, since the training data of this category can be easily obtained from the user clickthrough data [25], they have attracted many researchers, and three classic ranking algorithms, such as RankNet [10], RankBoost [12] and Ranking SVM (Ranking Support Vector Machine) [13] all belong to this type.

2.2. **Sparse learning to rank algorithms.** Although there are many works on obtaining models with high accuracy, much less effort has been made on sparse learning to rank, which is very important in the ranking applications, whose number of features is large.

Sun et al. presented the first work on inducing sparsity into the ranking algorithm, which is termed as RSRank [21]. Specifically, the authors reduced ranking to an important weighted pairwise classification with L1 regularization, and to achieve sparsity, truncated gradient descent is then utilized. Experimental results on the LETOR data sets showed that RSRank can obtain accurate model with good sparsity. Following this work, Lai et al. presented a convergence provable primal-dual algorithm, named FenchelRank [22]. The algorithm defined an L1 regularized pairwise ranking loss, and adopted the properties of the Fenchel duality to solve the inner optimization. The theoretical analysis of FenchelRank proved after at most $O(1/\varepsilon)$ iterations, the proposed algorithm can guarantee the obtainment of an epsilon-accurate solution. Empirical evaluation on the benchmark data sets demonstrated FenchelRank can not only provide a better accuracy than several state-of-the-art algorithms, but also outperform RSRank in terms of sparsity. In spite of the promising performance of FenchelRank, however, its inner iteration is complex, and to further improve the computational efficiency, Lai et al. developed a simple yet efficient algorithm (termed SparseRank) to solve the sparse learning to rank problem [23]. It employed the gradient descent method for the optimization, and by taking a small proper Lipschitz constant K as input, SparseRank can reduce the number of iterations greatly. Experimental results demonstrated the competitive performance of SparseRank in terms of both computational cost and quality of solution. Different from the sparse ranking algorithms above, which all adopt L1 convex regularization, Laporte et al. proposed a non-convex l_p $(0 < p < 1)$ regularization for the sparse model in learning to rank (termed as MCP) [24]. Numerical results on the LETOR benchmark data sets justified that MCP can lead to more sparsity in the ranking models while preserving comparable prediction accuracy. It is worth noting that there are also some works on applying the sparse ranking algorithms to different area, such as expert ranking [26] and image retrieval [27]; however, since the focus of this paper is on the sparse algorithm itself, we omit these applications, and interested reader can refer [26, 27].

In spite that the sparse ranking algorithms mentioned before can provide accurate model with much less features, all of them based on the batch learning, which indicates when the data set is of large scale, the computational cost of those algorithms may be expensive. Meanwhile, a fact is that in some real ranking applications, such as image retrieval [26], the numbers of examples and features are both large. Thus, how to develop efficient sparse ranking algorithm under this situation is an important yet challenging issue. To address the issue, in the following, we first relate sparse learning and stochastic learning together in approaching the problem of learning to rank, and propose a mini-batch stochastic gradient method to obtain sparse ranker.

Before we further present the details of the proposed algorithm, we briefly review some related works on the mini-batch stochastic gradient method. In machine learning area, it is well known that stochastic learning is very suitable for the applications with large scale [28, 29, 30]. Compared with the traditional SGD (Stochastic Gradient Descent), which only utilizes one example at each iteration, the mini-batch method aggregates multiple examples during one iteration, and can simultaneously achieve good accuracy and high computational efficiency [31, 32, 33]. Due to the merits above, the mini-batch stochastic gradient method has been widely used in many large scale machine learning problems. For example, Brockmeier et al., Tao et al. and Liu and Takac applied mini-batch method to solving the different classification problems, whose data sizes are very large [34, 35, 36], while Smith et al. and Wang et al. used it to design the efficient algorithms on the regression problems [37, 38]. However, different from the works above, in this paper, we firstly apply the mini-batch method to solving the sparse ranking problem.

3. The Proposed Algorithm: MinBatSRank.

3.1. The framework and objective function.

The algorithm we proposed belongs to pairwise method, which is similar to most of the existing sparse ranking algorithms [21, 22, 23, 24]. Specifically, we transform the ranking problem to a binary classification, and for each document pair $\left(x_k^i, x_k^j\right)$ in a given query q_k with different relevance labels, a binary classification example $x_k^{(i,j)}$ is created. If $y_k^i \succ y_k^j$, then the label of $y_k^{(i,j)}$ is set as $+1$, else $y_k^{(i,j)} = -1$. With the pairwise data above, we adopt the mini-batch framework used in classification. However, we shall note that different from classification, learning to rank has the property of "query level", which means only the examples in the same query can be used in one batch. Based on constraint above, we consider the stream of T pairwise examples to be composed of b batches: $Z_1, Z_2, \ldots, Z_b$, and any pairs in Z_t $(1 \le t \le b)$ belong to the same query. Then the loss function of batch Z_t can be defined as $l(Z_t, w) = \sum_{\left(x_k^{(i,j)}, y_k^{(i,j)}\right) \in Z_t} l\left(x_k^{(i,j)}, y_k^{(i,j)}, w\right)$, and the losses of b batches are defined as $l(Z_1, \ldots, Z_b, w) = \sum_{t=1}^{b} l(Z_t, w)$, where $l(x, y, w)$ is a suitable function which measures the discrepancy between a true label y and a predicted value from using parameter w, where $w \in R^d$ is the learned linear ranker. In this paper, we harness the squared hinge loss below, which is often used in learning to rank [22, 23, 24].

$$l\left(x_k^{(i,j)}, y_k^{(i,j)}, w\right) = \max\left(0, 1 - y_k^{(i,j)} w x_k^{(i,j)}\right)^2 \tag{2}$$

Based on Formula (2), we adopt the technique of online-to-batch conversion [39], and define the t-th penalty function as:

$$L_t(w) = l(Z_1, \ldots, Z_t, w) - l(Z_1, \ldots, Z_{t-1}, w) \tag{3}$$

With Equation (3), the proposed sparse ranking model can be obtained by solving the following optimization problem:

$$\text{OP1} \qquad w_{t+1} = \arg\min_{w \in R^d} \left\{L_t(w) + \lambda ||w||_1\right\} \tag{4}$$

where $\lambda > 0$ is a constant that controls the trade-off between training loss and regularization term. It should be noted that for OP1, directly using traditional stochastic optimization technique often fails to produce sparse solutions, since it belongs to black-box method, and has limited capability of exploiting problem structure in solving L1 regularized learning problem [40].

3.2. **The optimization method.** To overcome the above problem, in this paper, we adopt the FTRL (Follow-The-Regularized-Leader) method proposed by Mcmahan [41], which can exploit the regularization structure, and obtain sparse solutions effectively. The idea underlying FTRL is that it regards the objective function of Formula (4) as two parts. The first part is $\lambda||w||_1$, which is L1 regularization. The second part is $L_t(w)$. During the inner optimization, FTRL only takes the derivative of the second part, while keeping the regularization unchanged. By doing so, it can achieve both good sparsity and desirable performance. It should be mentioned that in the original work, to stabilize the FTRL algorithm, the author added a third component: a strong convexity Bregman divergence function, which forces the next iteration w_{t+1} to lie close to the first t rounds output weights, and the main update of FTRL is written as:

$$w_{t+1} = \arg\min_{w \in R^d} \left\{ \sum_{\tau=1}^{t} g_\tau \cdot w + \lambda \|w\|_1 + \sum_{\tau=1}^{t} \sigma_\tau \cdot B_\varphi(w, w_\tau) \right\} \tag{5}$$

where $g_\tau = \partial L_\tau(w)$ denotes an arbitrary (sub) gradient of $L_\tau(w)$. σ_τ and $B_\varphi(w, w_\tau)$ are the learning rate parameter and the Bregman Divergence function, respectively. In this paper, for $B_\varphi(w, w_\tau)$, we use the same setting as in [41], where $B_\varphi(w, w_\tau) = \frac{1}{2}||w - w_\tau||^2$. Then with Formula (5), we can solve OP1 effectively, and the whole algorithm is described in Algorithm 1.

Algorithm 1 FTRL Algorithm for Solving OP1

```
Input: training examples: $Z_1, Z_2, \ldots, Z_b$, the regularization parameter $\lambda$, the generalized
    learning rate $\sigma_\tau$, the maximum number of iterations $T$
Initialize: ranking model $w_0 \in R^d$
1: for $t = 1, \ldots, T$ do
2:
     $$w_{t+1} = \arg\min_{w \in R^d} \left\{ \sum_{\tau=1}^{t} g_\tau \cdot w + \lambda \|w\|_1 + \frac{1}{2} \sum_{\tau=1}^{t} \sigma_\tau \cdot ||w - w_\tau||^2 \right\} \tag{6}$$
3: end for
Output: return $w_T$ as the final ranking model
```

With Algorithm 1, we can obtain a sparse ranking model effectively. Note that as a stochastic gradient based method, the learning rate parameter σ_τ has great influence on the final model, and in the original FTRL work, σ_τ is suggested to be set as $\sigma_\tau = \frac{1}{\eta_\tau} - \frac{1}{\eta_{\tau-1}}$ $\left(\eta_\tau = \frac{1}{\sqrt{\tau}}\right)$. However, by taking a closer look at σ_τ, we can find this setting can be further improved, since during the inner optimization, σ_τ is equally applied to all features (coordinates), and it is clearly not the optimal behavior. Therefore, in the following, inspired by Adaptive Gradient Updating (AGU) proposed by Duchi et al. [42], we provide a feature-wise learning rates updating strategy, which employs the second order gradient optimization technique, and can yield better performance for the final ranking model.

The intuition of this updating strategy is very natural, which considers the rare occurring features as more informative and discriminative than those frequently occurring features. Thus, these informative rare occurring features should be updated with higher learning rates by incorporating the geometrical property of the data observed in earlier stages. Besides, by using the previously observed gradients, the updating process can mitigate the possible effects of noise data. Thus based on this intuition, a feature-wise learning rates updating strategy is presented, and the detail of it is demonstrated in Algorithm 2.

Algorithm 2 Feature-wise Learning Rates Updating Strategy

```
Input: $\gamma > 0$, $\rho \geq 0$
Variables: $\eta_\tau \in R^d$, $g_\tau \in R^d$, $D_\tau \in R^d$ and $H_\tau \in R^d$
Initialize: $\eta_0 = 0$, $g_0 = 0$, $D_0 = 0$, $H_0 = 0$
1: for $\tau = 1, \ldots, T$ do
2:     Receive (sub)gradient $g_\tau = \partial L_\tau(w)$
3:     for $j = 1, \ldots, d$ do
4:         Update $g_{\tau,j} = g_{\tau,j} + (g_{\tau-1,j})^2$
5:         Set $H_{\tau,j} = \sqrt{\rho + g_{\tau,j}}$
6:         $\eta_{\tau,j} = H_\tau^{-1} \cdot \gamma$
7:         $\sigma_{\tau,j} = \dfrac{1}{\eta_{\tau,j}} - \dfrac{1}{\eta_{\tau-1,j}}$
8:     end for
9: end for
Output: $\sigma_\tau$
```

In Algorithm 2, γ is an initial learning rate, and ρ is a controlling parameter, which ensures the early learning rate not to be too high.

According to Algorithm 2, the solution of Equation (6) is presented in Theorem 3.1.

Theorem 3.1. *The solution of Equation (6) is $w_{t+1} = [w_{t+1,1}, \ldots, w_{t+1,d}]$, and for $j \in \{1, \ldots, d\}$*

$$w_{t+1,j} = \begin{cases} 0 & \text{if } |v_{t,j}| \leq \lambda \\ -\left(H_t^{-1} \cdot \gamma\right)(v_{t,j} - \lambda) & \text{if } v_{t,j} > \lambda \\ -\left(H_t^{-1} \cdot \gamma\right)(v_{t,j} + \lambda) & \text{if } v_{t,j} < -\lambda \end{cases} \tag{7}$$

where $v_{t,j} = \sum_{\tau=1}^{t} (g_{\tau,j} - \sigma_{\tau,j} \cdot w_{\tau,j})$.

Proof: We can re-write Equation (6) as:

$$\begin{aligned} w_{t+1} = \arg\min_{w \in R^d} \Bigg\{ & \left(\sum_{\tau=1}^{t} g_\tau - \sum_{\tau=1}^{t} \sigma_\tau \cdot w_\tau \right) \cdot w + \lambda \|w\|_1 \\ & + \frac{1}{2} \sum_{\tau=1}^{t} \sigma_\tau \cdot \|w\|_2^2 + \frac{1}{2} \sum_{\tau=1}^{t} \sigma_\tau \cdot \|w_\tau\|_2^2 \Bigg\} \end{aligned} \tag{8}$$

which can be further simplified as:

$$w_{t+1} = \arg\min_{w \in R^d} \left\{ v_t \cdot w + \lambda \|w\|_1 + \frac{1}{2} \sum_{\tau=1}^{t} \sigma_\tau \cdot \|w\|_2^2 \right\} \tag{9}$$

where $v_{t,j} = \sum_{\tau=1}^{t} (g_{\tau,j} - \sigma_{\tau,j} \cdot w_{\tau,j})$.

For a solution vector $w_{t+1} = [w_{t+1,1}, \ldots, w_{t+1,d}] \in R^d$, we can compute each $w_{t+1,j}$ independently, where $j \in \{1, \ldots, d\}$. Thus, we begin the analysis by rewriting Formula

(9) as:

$$w_{t+1,j} = \arg\min_{w_j \in R^d} \left\{ v_{t,j} \cdot w_j + \lambda \left| w_j \right| + \frac{1}{2} \sum_{\tau=1}^{t} \sigma_\tau \cdot (w_j)^2 \right\} \tag{10}$$

We take the derivative of Equation (10) with respect to w to zero and denote $\mathrm{sgn}(w)$ as the sub-gradient of $|w|_1$, and then have

$$v_{t,j} + \lambda \cdot \mathrm{sgn}(w_j) + \sum_{\tau=1}^{t} \sigma_\tau \cdot w_j = 0 \tag{11}$$

$$\mathrm{sgn}(w) = \begin{cases} \{w \in R | -1 \le w \le 1\} & \text{if } w = 0 \\ 1 & \text{if } w > 0 \\ -1 & \text{if } w < 0 \end{cases} \tag{12}$$

In the following, we will further discuss three cases for finding the solution of Equation (11).

(1) If $|v_{t,j}| \le \lambda$, $w_j = 0$ satisfies the equality of Equation (12), when let $\mathrm{sgn}(w_j) = -\frac{v_{t,j}}{\lambda} \in [-1,\ 1]$.

(2) If $v_{t,j} > \lambda$, then $w_j < 0$. If not, assume that $w_j > 0$, we have $\mathrm{sgn}(w_j) = 1$. Substituting $\mathrm{sgn}(w_j) = 1$ into Equation (11), we obtain $v_{t,j} + \lambda + \sum_{\tau=1}^{t} \sigma_\tau \cdot w_j > v_{t,j} + \lambda \ge 0$. This is a contradiction. Again assume that $w_j = 0$, according to Equation (11), we have $\mathrm{sgn}(w_j) = -\frac{v_{t,j}}{\lambda} < -1$, and this is a contradiction with $\mathrm{sgn}(w_j) \ge -1$. So we must have $w_j < 0$. According to Algorithm 2, the solution is $w_j = -\left(H_t^{-1} \cdot \gamma\right)(v_{t,j} - \lambda)$.

(3) Likewise, when $v_{t,j} < -\lambda$, we must have $w_j > 0$. The solution is $w_j = -\left(H_t^{-1} \cdot \gamma\right)$ $(v_{t,j} + \lambda)$.

Combining the three cases, we obtain

$$w_{t+1,j} = \begin{cases} 0 & \text{if } |v_{t,j}| \le \lambda \\ -\left(H_t^{-1} \cdot \gamma\right)(v_{t,j} - \lambda) & \text{if } v_{t,j} > \lambda \\ -\left(H_t^{-1} \cdot \gamma\right)(v_{t,j} + \lambda) & \text{if } v_{t,j} < -\lambda \end{cases}$$

This concludes the proof.

With Algorithms 1 and 2, we can solve the OP1 efficiently. We name the proposed mini-batch sparse ranking method as MinBatSRank, and in the next section, we will evaluate MinBatSRank with other state-of-the-art methods on the benchmark data sets.

4. **Experiments.**

In this section, we conduct a series of experiments based on LETOR set to validate the performance of the proposed MinBatSRank by comparing it with several representative baseline ranking algorithms. More specifically, we first present the experimental setting (including data sets, comparison algorithms and evaluation metrics), and then report the comparison results between the proposed algorithm and other baselines (including dense and sparse ranking algorithms). Finally, we discuss the effectiveness of feature-wise learning rates strategy in the proposed algorithm.

4.1. **Experimental setting.**

4.1.1. *Data sets.* The experimental sets are chosen from LETOR3.0 [43], which are considered as the benchmark data sets in learning to rank. In LETOR3.0, there are seven data sets collected for four different search tasks of document retrieval: Topic Distillation (TD2003, TD2004), Home Page finding (HP2003, HP2004), Named Page finding (NP2003, NP2004) and medical document retrieval (OHSUMED). In this work, to make a fair comparison, we select one set from each search task. Specifically, we use TD2004,

Table 1. Characteristics of experimental sets

Data set	Queries	Features	Rel.levels	Query-document pairs
TD2004	75	64	2	74146
HP2004	75	64	2	74409
NP2004	75	64	2	73834
OHSUMED	106	45	3	16140

HP2004, NP2004 and OHSUMED as the experimental sets. Among them, the first three are two-level set, and the last one (OHSUMED) is three-level ranking set. The detailed characteristics of those four sets are depicted in Table 1.

For each set above, LETOR3.0 divides it into five folds and each fold contains a training/validation/test set, respectively. In the following experiments, we adopt the same splits as LETOR 3.0 provides, and report the results by averaging on the five folds.

4.1.2. *Comparison algorithms.* In this paper, five representative algorithms are chosen to compare with MinBatSRank. More specifically, we evaluate our algorithm from two aspects. First, MinBatSRank is compared with three dense (L2 based) ranking algorithms, which are ListNet [18], AdaRank-NDCG (Adaptive Rank-NDCG) [17], and RankSVM-Struct [11]. The reason we choose them lies in the fact that they are all well-known baselines in learning to rank. The former two belong to listwise method, and optimize cross-entropy loss and NDCG criteria respectively. The third one RankSVM-Struct is a pairwise approach, which defines pairwise loss function with structural SVM. Then we compare MinBatSRank with two sparse ranking algorithms, i.e., SparseRank [23] and FenchelRank [22], which are recently proposed methods with both high accuracy and sparsity.

For fair comparison, all parameters of the five compared algorithms are set to the recommended values from their work. While for the proposed method, we set $\rho = 1$, and the parameters γ, λ are chosen from $\left\{2^1, \ldots, 2^{-10}\right\}$ and $\left\{2^{15}, \ldots, 2^{-2}\right\}$, respectively. All the experimental results reported in this work are conducted on an PC with an Intel Core i5-4590 CPU 3.30GHz, 4G internal storage and the Windows 7 SP1 64 bit operating system.

4.1.3. *Evaluation measures.* In this paper, we use two popular ranking metrics to evaluate the effectiveness of the proposed method. The first one is NDCG (Normalized Discounted Cumulative Gain) [44], which is the most common measure in learning to rank. It works for the case with multi-level relevance judgments, and for a query q, DCG score at position k is formally defined as:

$$DCG@k = \sum_{j=1}^{k} \frac{2^{r(j)} - 1}{\log_2(1+j)} \tag{13}$$

where $r(j)$ is the relevance label of the j-th document in the sorted list. Then Normalized DCG score at position k in the ranking list of documents can be calculated by the equation as follows:

$$NDCG@k = Z \cdot \sum_{j=1}^{k} \frac{2^{r(j)} - 1}{\log_2(1+j)} \tag{14}$$

where Z is the normalization constant so that the value of NDCG ranges from 0 to 1. In the rest of this paper, we use N@k as the abbreviation of NDCG@k (which means the NDCG value at position k).

Another evaluation metric is MAP (Mean Average Precision) [45], which deals with binary relevance judgments: relevant and irrelevant. First, we shall introduce the definition of Precision at k, which denotes the proportion of relevant documents in the first k positions:

$$Pre@k = \frac{1}{k}\sum_{j=1}^{k}\zeta_j \tag{15}$$

where ζ_j is an indicator function. If the document at position j is relevant, $\zeta_j = 1$, else ζ_j is 0. Then the average precision of a given query q is defined as the following equation:

$$AP(q) = \frac{\sum\limits_{k=1}^{n} Pre@k \times \zeta_k}{n_{rel}} \tag{16}$$

where n and n_{rel} represent the total numbers of documents and relevant documents associated with query q, respectively. Based on Equations (10) and (11), MAP can be formally defined as:

$$MAP = \frac{1}{|Q|}\sum_{q\in Q} AP(q) \tag{17}$$

where Q is the set of all queries.

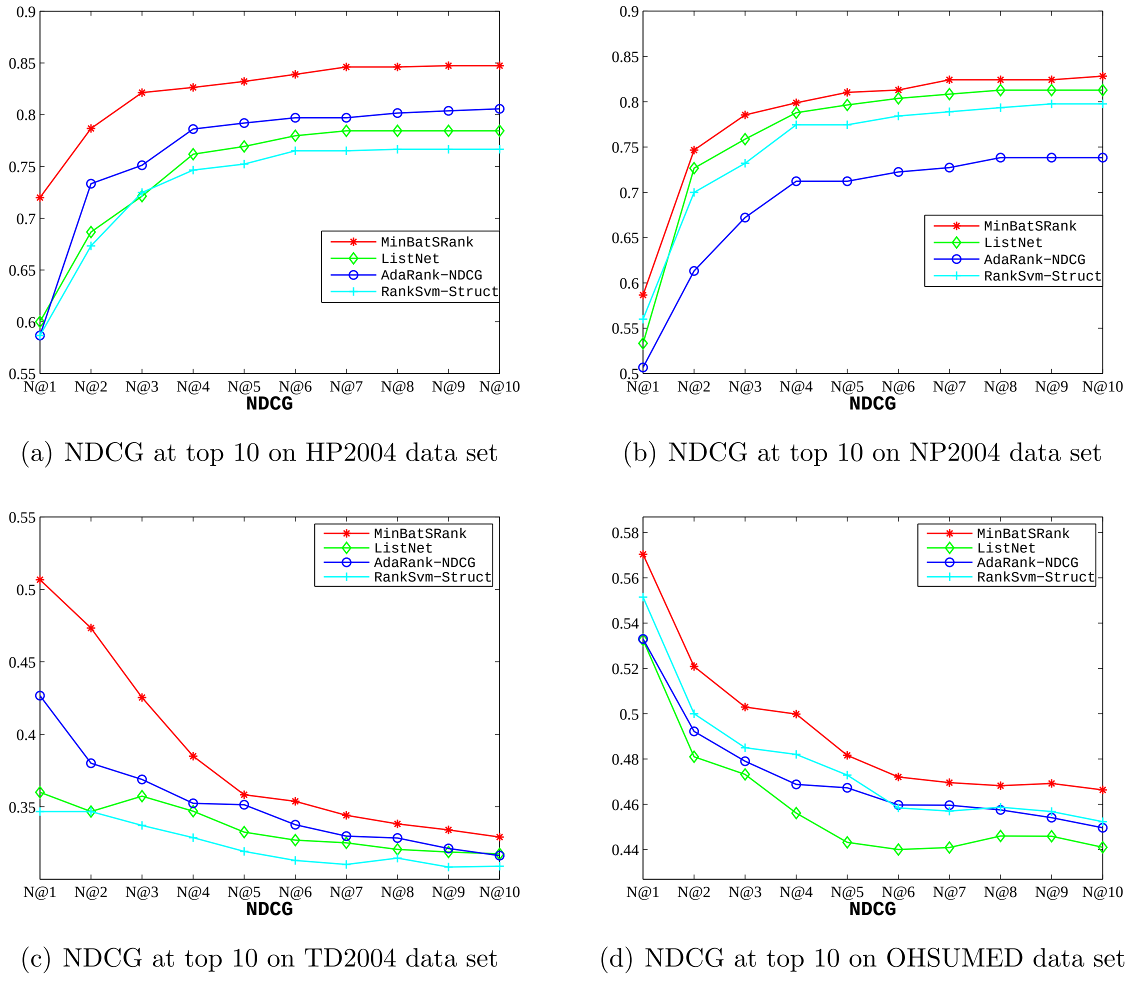
4.2. **Experimental results.**

4.2.1. *Comparison results between MinBatSRank and dense baselines.* In the first set of experiments, we are concerned about how our algorithm performs, when compared with dense (non-sparse) ranking methods. Thus, we evaluate MinBatSRank with ListNet [18], AdaRank-NDCG [17] and RankSVM-Struct [11] on four experimental sets. The results are measured in terms of N@1 to N@10, and Figure 1 presents the performances of different algorithms in detail.

From Figure 1, we can find that three compared algorithms perform variously from one data set to another. For example, ListNet achieves a good performance on NP2004, while performs poorly on OHSUMED. AdaRank-NDCG performs well on TD2004 and HP2004, but yields a poor result on NP2004. RankSVM-Struct can obtain a better performance on OHSUMED, but fails to deliver accurate ranking results on HP2004 and TD2004. However, different from those algorithms, our MinBatSRank algorithm appears to be more stable, and is always the best one on all the data sets. We would like to point out a few statistics. On HP2004 data set, MinBatSRank performs 0.7200 at N@1, a 20% increase in performance, compared to ListNet, the second best algorithm. On NP2004 set, this value for MinBatSRank is 0.5867 that shows a 4.77% increasement, compared with RankSVM-Struct (0.5600), the second best approach. On TD2004 and OHSUMED data sets, the corresponding improvements of the proposed method compared with those of the second best one are 18.73% and 3.42%, respectively.

The empirical results in Figure 1 have demonstrated the superior performance of the proposed MinBatSRank, when compared with dense baselines. In the following, we will further compare our algorithm with other sparse ranking methods.

4.2.2. *Comparison results between MinBatSRank and other sparse baselines.* As the proposed MinBatSRank is a stochastic sparse algorithm, in this section, we compare MinBatSRank with other state-of-the-art sparse ranking methods in terms of accuracy, sparsity and computational efficiency. Specifically, we choose the SparseRank [23] and FenchelRank [22] as the comparison methods, which are well known for their efficiency and good



(a) NDCG at top 10 on HP2004 data set

(b) NDCG at top 10 on NP2004 data set

(c) NDCG at top 10 on TD2004 data set

(d) NDCG at top 10 on OHSUMED data set

FIGURE 1. The performances between MinBatSRank and dense baselines on LETOR data sets

performances. It should be noted that we do not include RSRank [21] and MCP [24], since RSRank has been proven to be outperformed by SparseRank in terms of accuracy and sparsity [22, 23]. While for MCP, in spite of its high sparsity, it is a non-convex algorithm, which only guarantees local optima, to obtain better performance, MCP often needs a proper initialization, and this operation will increase its computational cost greatly.

The experimental results are reported from following aspects. Firstly, we list the accuracy of different algorithms in terms of NDCG and MAP, and the results are shown in Table 2. It should be noticed that since OHSUMED is a 3-level set (0, 1 and 2), to compute the MAP value on OHSUMED, we regard the examples with label 2 as relevant, and other labels as non-relevant examples.

As shown in Table 2, with all the data sets, MinBatSRank performs best on 15 of 20 statistical points, and the second best is on 4 of 20 statistical points. This statistics proves the superior ranking accuracy of MinBatSRank, when compared with other sparse ranking methods.

Secondly, we evaluate the sparsity and training time between our algorithm and SparseRank, FenchelRank. To measure the sparsity, we adopt the sparsity ratio defined in [22], which has the form of $\frac{|w|_0}{m}$, where $|w|_0$ presents the number of nonzero coefficients in model, and m is the number of total dimensions in w. Lower ratio means better sparsity.

Table 2. Ranking accuracy of different sparse algorithms on LETOR data sets

	N@1	N@3	N@5	N@10	MAP
HP2004					
SparseRank	0.6799	0.7687	0.8098	0.8135	0.7435
FenchelRank	0.6800	0.7880	0.8123	0.8239	0.7475
MinBatSRank	**0.7200**	**0.8213**	**0.8321**	**0.8474**	**0.7795**
NP2004					
SparseRank	**0.6133**	0.7734	0.7972	0.8180	0.6837
FenchelRank	0.5600	0.7752	0.7905	0.8148	**0.7144**
MinBatSRank	0.5867	**0.7855**	**0.8103**	**0.8282**	0.7059
TD2004					
SparseRank	0.3867	0.3523	0.3263	0.3151	**0.2362**
FenchelRank	0.3866	0.3763	0.3524	0.3214	0.2241
MinBatSRank	**0.5066**	**0.4253**	**0.3583**	**0.3291**	0.2139
OHSUMED					
SparseRank	0.5671	0.5000	0.4764	0.4563	0.4489
FenchelRank	0.5453	**0.5131**	**0.4879**	0.4608	0.4480
MinBatSRank	**0.5704**	0.5030	0.4816	**0.4663**	**0.4511**

Table 3. Sparsity ratios of different algorithms on LETOR data sets

	HP2004	NP2004	TD2004	OHSUMED
FenchelRank	**0.2031**	**0.2625**	0.4875	0.2844
SparseRank	0.3906	0.6406	0.5875	0.2978
MinBatSRank	0.3469	0.3781	**0.4625**	**0.2311**

Table 4. Training time (Seconds) of different algorithms on LETOR data sets

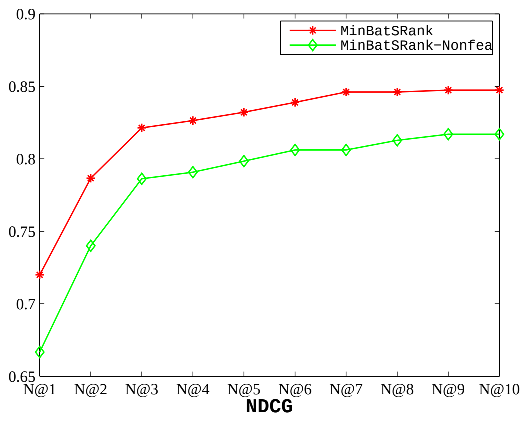
	HP2004	NP2004	TD2004	OHSUMED
FenchelRank	26.97	48.45	1585.03	254.97
SparseRank	15.33	8.61	33.02	18.89
MinBatSRank	**0.68**	**0.08**	**0.53**	**0.04**

Tables 3 and 4 present the sparsity ratios and the training time of three algorithms on the LETOR data sets.

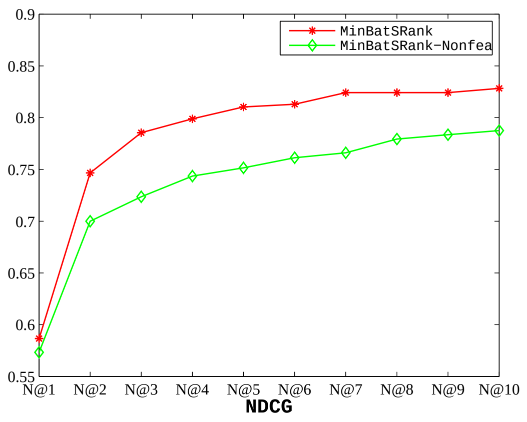
From Table 3, we can find that sparsity ratios of MinBatSRank are optimal on TD2004 and OHSUMED sets, and is the second best on HP2004 and NP2004, which indicates the promising performance of SminBatRank, when measured by sparsity. Meanwhile, Table 3 also shows that FenchelRank can also produce good sparse model on the LETOR sets. The main reason is attributed to the fact that during the inner iterations, FenchelRank greedily selects a feature to achieve the largest absolute ranking value, which may result in the final ranker learned by FenchelRank can obtain comparable accuracy with much less features. However, this greedy operation will increase computational cost, and as shown in Table 4, the training time of FenchelRank is much longer than SparseRank and MinBatSRank, especially for our MinBatSRank. Table 4 demonstrates that its computational efficiency outperforms other two algorithms greatly, which confirms that the algorithm we proposed is very suitable for the data set with large scale. Finally, we note that SparseRank also has less computational cost than FenchelRank. Further analysis reveals that in SparseRank, there is a Lipschitz constant K controlling the number of iterations, and in the suggested setting, K is set with a small value, which will decrease the running time.

Combining Tables 2, 3 and 4, we can empirically conclude that MinBatSRank is a promising sparse ranking algorithm on LETOR data sets, in terms of both accuracy, sparsity and computational efficiency.

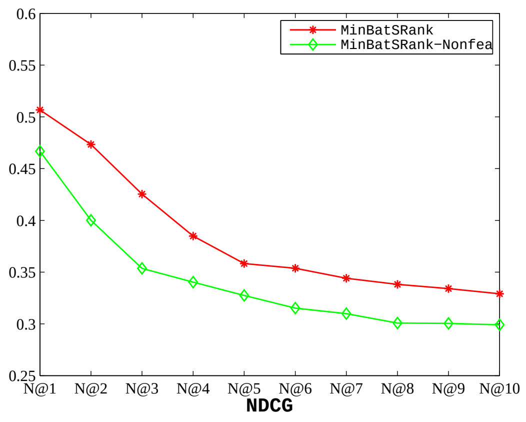
4.2.3. *Effectiveness of feature-wise learning rates updating strategy.* As mentioned before, in the proposed MinBatSRank, a feature-wise learning rates updating strategy is proposed at each iteration, and in the following, we will empirically investigate the influence of this strategy on the performance of MinBatSRank for LETOR data sets. To be specific, we modify our algorithm with the updating strategy used in original FTRL, which sets $\sigma_\tau = \frac{1}{\sqrt{\tau}} - \frac{1}{\sqrt{\tau - 1}}$. This algorithm is termed MinBatSRank-Nonfea, and adopted as the compared baseline. The experimental results are also reported in terms of N@1 to N@10, and Figure 2 below plots the results in detail.



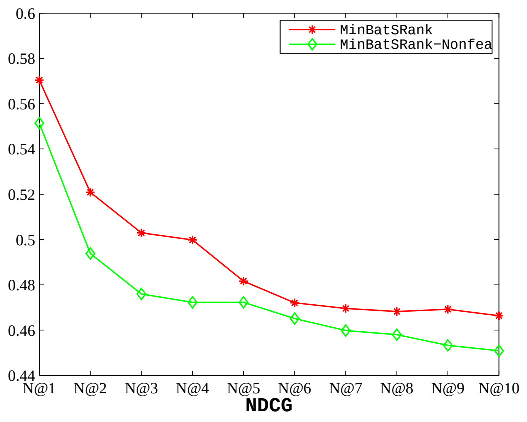
(a) The results on HP2004 data set



(b) The results on NP2004 data set



(c) The results on TD2004 data set



(d) The results on OHSUMED data set

FIGURE 2. The performances between MinBatSRank and MinBatSRank-Nonfea on LETOR data sets

As can be seen from Figure 2, on all the experimental sets, the algorithm we proposed outperforms MinBatSRank-Nonfea significantly, which proves the effectiveness of our feature-wise learning rates updating strategy. This strategy is also theoretically sound, since it takes full advantages of historical gradient information available in learning phase. Based on this previous information, the feature with large gradient has small learning rate, and the one with small gradient has large rate, which can speed up the convergence, and improve the performance of MinBatSRank greatly.

5. Conclusion and Future Work.

In this paper, we have proposed a mini-batch stochastic gradient method named MinBatSRank for sparse ranking. First of all, an objective function composed of mini-batch based pairwise loss and L1-norm regularization penalty is defined. Then to obtain sparse solution, FTRL optimization method is adopted. We also suggested a feature-wise learning rates updating strategy, which can further improve the performance of the proposed MinBatSRank. Experimental results on benchmark collections demonstrated the superior performance of the proposed algorithm in terms of accuracy, sparsity and efficiency.

There still remain some interesting works related with sparse learning to rank that deserve to be further investigated. For example, same with most of the sparse ranking algorithms (such as RSRank, FenchelRank and SparseRank), MinBatSRank is a pairwise algorithm; however, it is well known that in the real world, listwise approaches also have many applications. How to design a sparse stochastic ranking approach based on listwise method is one of our future works. Moreover, as far as we know, the existing sparse ranking algorithms all solve the problem with the technique of single objective optimization. In the future, it is desirable to produce a sparse ranker from the aspect of multi-objective optimization [46].

Acknowledgment. This work is supported by the National Natural Science Foundation of China (Grant Nos. 61502001, 61502004) and the Natural Science Foundation of Anhui Province (Grant No. 1708085MF166), Key Program of Natural Science Project of Educational Commission of Anhui Province (Grant Nos. KJ2015A070, KJ2017A013), the Academic and Technology Leader Imported Project of Anhui University (No. J01006057), Youth Foundation of Anhui University (Grant No. KJQN1119), the Doctor Foundation of Anhui University (Grant No. 01001902), the Foundation for the Key Teacher by Anhui University. This work is also supported by Co-Innovation Center for Information Supply & Assurance Technology, Anhui University.

REFERENCES

[1] A. Grotov and M. De Rijke, Online learning to rank for information retrieval: SIGIR 2016 tutorial, *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.1215-1218, 2016.

[2] H. Li, *Learning to Rank for Information Retrieval and Natural Language Processing*, Morgan & Claypool Publishers, 2011.

[3] R. Zhang, H. Bao, H. Sun, Y. Wang and X. Liu, Recommender systems based on ranking performance optimization, *Frontiers of Computer Science*, vol.10, no.2, pp.270-280, 2016.

[4] S. V. Jose and M. L. Madhavu, Incremental iterative time spent based ranking model for online activitybased friend-group recommendation systems, *International Conference on Computing, Communication and Networking Technologies*, pp.1-6, 2015.

[5] W. Huang, S. Zeng, J. Li and G. Chen, A new image-based immersive tool for dementia diagnosis using pairwise ranking and learning, *Multimedia Tools and Applications*, vol.75, no.9, pp.5359-5376, 2016.

[6] D. Cossock and T. Zhang, Subset ranking using regression, *Conference on Learning Theory*, pp.605-619, 2006.

[7] R. Nallapati, Discriminative models for information retrieval, *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.64-71, 2004.

[8] P. Li, C. J. C. Burges and Q. Wu, Mcrank: Learning to rank using multiple classification and gradient boosting, *International Conference on Neural Information Processing Systems*, pp.897-904, 2007.

[9] K. Crammer and Y. Singer, Pranking with ranking, *Advances in Neural Information Processing Systems*, vol.14, pp.641-647, 2001.

[10] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton and G. Hullender, Learning to rank using gradient descent, *International Conference on Machine Learning*, pp.89-96, 2005.
[11] T. Joachims, Training linear SVMs in linear time, *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.217-226, 2006.
[12] Y. Freund, R. D. Iyer, R. E. Schapire and Y. Singer, An efficient boosting algorithm for combining preferences, *The 15th International Conference on Machine Learning*, pp.170-178, 1998.
[13] T. Joachims, Optimizing search engines using clickthrough data, *The 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.133-142, 2002.
[14] Z. Zheng, K. Chen, G. Sun and H. Zha, A regression framework for learning ranking functions using relative relevance judgments, *Proc. of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.287-294, 2007.
[15] M. F. Tsai, T. Y. Liu, T. Qin, H. H. Chen and W. Y. Ma, FRank: A ranking method with fidelity loss, *Proc. of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Amsterdam, the Netherlands, pp.383-390, 2007.
[16] Y. Yue, T. Finley, F. Radlinski and T. Joachims, A support vector method for optimizing average precision, *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.271-278, 2007.
[17] J. Xu and H. Li, AdaRank: A boosting algorithm for information retrieval, *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.391-398, 2007.
[18] Z. Cao, T. Qin, T. Y. Liu, M. F. Tsai and H. Li, Learning to rank: From pairwise approach to listwise approach, *International Conference on Machine Learning*, pp.129-136, 2007.
[19] F. Xia, T. Y. Liu, J. Wang, W. Zhang and H. Li, Listwise approach to learning to rank: Theory and algorithm, *Proc. of the 25th International Conference on Machine Learning*, Helsinki, Finland, pp.1192-1199, 2008.
[20] S. G. Jeong, *Curvilinear Structure Modeling and Its Applications in Computer Vision*, Ph.D. Thesis, Université Nice Sophia Antipolis, 2016.
[21] Z. Sun, T. Qin, Q. Tao and J. Wang, Robust sparse rank learning for non-smooth ranking measures, *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.259-266, 2009.
[22] H. Lai, Y. Pan, C. Liu, L. Lin and J. Wu, Sparse learning-to-rank via an efficient primal-dual algorithm, *IEEE Trans. Computers*, vol.62, no.6, pp.1221-1233, 2013.
[23] H. Lai, Y. Pan, Y. Tang and N. Liu, Efficient gradient descent algorithm for sparse models with application in learning-to-rank, *Knowledge-Based Systems*, vol.49, no.9, pp.190-198, 2013.
[24] L. Laporte, R. Flamary, S. Canu, S. Déjean and J. Mothe, Nonconvex regularizations for feature selection in ranking with sparse SVM, *IEEE Trans. Neural Networks Learning Systems*, vol.25, no.6, pp.1118-1130, 2015.
[25] I. Sarafis, C. Diou and A. Delopoulos, Building effective SVM concept detectors from clickthrough data for large-scale image retrieval, *International Journal of Multimedia Information Retrieval*, vol.4, no.2, pp.129-142, 2015.
[26] L. Wang, Z. Yu, T. Jin, X. Li and S. Gao, Expert list-wise ranking method based on sparse learning, *Neurocomputing*, vol.217, pp.119-124, 2016.
[27] X. Gao, S. C. H. Hoi, Y. Zhang, J. Wan and J. Li, SOML: Sparse online metric learning with application to image retrieval, *The 28th AAAI Conference on Artificial Intelligence*, pp.1206-1212, 2014.
[28] L. Bottou, Large-scale machine learning with stochastic gradient descent, *Proc. of COMPSTAT'2010*, pp.177-186, 2010.
[29] T. Parnell, C. Duenner, K. Atasu, M. Sifalakis and H. Pozidis, Large-scale stochastic learning using gpus, *Parallel and Distributed Processing Symposium Workshops*, pp.419-428, 2017.
[30] W. Xu, Towards optimal one pass large scale learning with averaged stochastic gradient descent, *Computer Science*, 2011.
[31] A. Cotter, O. Shamir, N. Srebro and K. Sridharan, Better mini-batch algorithms via accelerated gradient methods, *Advances in Neural Information Processing Systems*, pp.1647-1655, 2011.
[32] S. Shalevshwartz and T. Zhang, Accelerated mini-batch stochastic dual coordinate ascent, *Advances in Neural Information Processing Systems*, pp.378-385, 2013.
[33] J. Konecny, J. Liu, P. Richtarik and M. Takac, Mini-batch semi-stochastic gradient descent in the proximal setting, *IEEE Journal of Selected Topics in Signal Processing*, vol.10, no.2, pp.242-255, 2016.

[34] A. J. Brockmeier, J. S. Choi, E. G. Kriminger, J. T. Francis and J. C. Principe, Neural decoding with kernel-based metric learning, *Neural Computation*, vol.26, no.6, pp.1080-1107, 2014.

[35] H. Tao, B. Wu and X. Lin, Budgeted mini-batch parallel gradient descent for support vector machines on spark, *IEEE International Conference on Parallel and Distributed Systems*, pp.945-950, 2015.

[36] J. Liu and M. Takac, Projected semi-stochastic gradient descent method with mini-batch scheme under weak strong convexity assumption, *Modeling and Optimization: Theory and Applications*, pp.95-117, 2016.

[37] V. Smith, S. Forte, M. I. Jordan and M. Jaggi, L1-regularized distributed optimization: A communication-efficient primal-dual framework, *eprint arXiv:1512.04011*, 2015.

[38] C. Wang, X. Yan, M. Smith and K. Kochhar, A unified framework for automatic wound segmentation and analysis with deep convolutional neural networks, *Engineering in Medicine and Biology Society*, pp.2415-2418, 2015.

[39] P. Kar, H. Narasimhan and P. Jain, Online and stochastic gradient methods for non-decomposable loss functions, *Advances in Neural Information Processing Systems*, vol.1, pp.694-702, 2014.

[40] L. Xiao, Dual averaging methods for regularized stochastic learning and online optimization, *Journal of Machine Learning Research*, vol.11, pp.2543-2596, 2010.

[41] H. B. Mcmahan, Follow-the-regularized-leader and mirror descent: Equivalence theorems and L1 regularization, *Jmlr*, vol.15, p.2011, 2011.

[42] J. Duchi, E. Hazan and Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, *Journal of Machine Learning Research*, vol.12, no.7, pp.2121-2159, 2011.

[43] T. Qin, T. Y. Liu, J. Xu and H. Li, LETOR: A benchmark collection for research on learning to rank for information retrieval, *Information Retrieval Journal*, vol.13, no.4, pp.346-374, 2010.

[44] K. Järvelin and J. Kekäläinen, Cumulated gain-based evaluation of IR techniques, *ACM Trans. Information Systems*, vol.20, no.4, pp.422-446, 2002.

[45] R. Baeza-Yates and B. Ribeiro-Neto, Modern information retrieval: Addison Wesley, *Computer Science and Information Technology (CS and IT)*, 1999.

[46] X. Zhang, Y. Tian, R. Cheng and Y. Jin, An efficient approach to nondominated sorting for evolutionary multiobjective optimization, *IEEE Trans. Evolutionary Computation*, vol.19, no.2, pp.201-213, 2015.