# Learning Preferences with Millions of Parameters by Enforcing Sparsity

**Xi Chen** [(1)(2)] **Bing Bai**[(1)] **Yanjun Qi**[(1)] **Qihang Lin**[(2)] **Jaime Carbonell**[(2)]

[(1)]NEC Labs America, Princeton, NJ
{bbai,yanjun}@nec-labs.com

[(2)] Carnegie Mellon University, Pittsburg, PA
{xichen, qihangl, jgc}@cs.cmu.edu

***Abstract*—We study the retrieval task that ranks a set of objects for a given query in the pairwise preference learning framework. Recently researchers found out that raw features (e.g. words for text retrieval) and their pairwise features which describe relationships between two raw features (e.g. word synonymy or polysemy) could greatly improve the retrieval precision. However, most existing methods can not scale up to problems with many raw features (e.g. English vocabulary), due to the prohibitive computational cost on learning and the memory requirement to store a quadratic number of parameters. In this paper, we propose to learn a sparse representation of the pairwise features under the preference learning framework using the L1 regularization. Based on stochastic gradient descent, an online algorithm is devised to enforce the sparsity using a mini-batch shrinkage strategy. On multiple benchmark datasets, we show that our method achieves better performance with fast convergence, and takes much less memory on models with millions of parameters.**

***Keywords*-preference learning; sparse model; online learning; learning to rank; text mining**

## I. Introduction

Learning preferences among a set of objects (e.g. documents) given another object as query is a central task of information retrieval and text mining. One of the most natural frameworks for this task is the *pairwise preference learning*, expressing that one document is preferred over another given the query [1]. Most existing methods [2] learn the preference or relevance function by assigning a real valued score to a feature vector describing a (query, object) pair. This feature vector normally includes a small number of hand-crafted features, such as the BM25 scores for the title or the whole text, instead of the very natural raw features [3]. A drawback of using hand-crafted features is that they are often expensive and specific to datasets, requiring domain knowledge in preprocessing. In contrast, the raw features are easily available, and carry strong semantic information (such as word features in text mining).

In this paper we study a basic model presented in [4], [5] which uses the raw word features under the supervised pairwise preference learning framework and consider feature relationships in the model[1]. To be specific, let $\mathcal{D}$ be the dictionary size, i.e. the size of the query and document feature set[2], given a query $q \in \mathbb{R}^{\mathcal{D}}$ and a document $d \in \mathbb{R}^{\mathcal{D}}$, the relevance score between $q$ and $d$ is modeled as:

$$f(q,d) = q^\top W d = \sum_{i,j} W_{ij}\Phi(q_i, d_j), \qquad (1)$$

where $\Phi(q_i, d_j) = q_i \cdot d_j$ and $W_{ij}$ models the relationship/correlation between $i^{th}$ query feature $q_i$ and $j^{th}$ document feature $d_j$. This is essentially a linear model with pairwise features $\Phi(\cdot,\cdot)$ and the parameter matrix $W \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ is learned from labeled data. Compared to most of the existing models, the capacity of this model is very large because of the $\mathcal{D}^2$ free parameters which can carefully model the relationship between each pair of words. From a semantic point of view, a notable superiority of this model is that it can capture synonymy and polysemy as it looks at all possible cross terms, and can be tuned directly for the task of interest.

Although it is very powerful, the basic model in Eq. (1) suffers from the following weakness which hinders its wide application:

1) Memory issue: Given the large dictionary size $\mathcal{D}$, it requires a huge amount of memory to store the $W$ matrix with a size quadratic in $\mathcal{D}$. When $\mathcal{D} = 10,000$, storing $W$ needs nearly 1Gb of RAM (assuming double); when $\mathcal{D} = 30,000$, it requires 8Gb of RAM.
2) Generalization ability: Given $\mathcal{D}^2$ free parameters (entries of $W$), when the number of training samples is limited, it can easily lead to overfitting. Considering the dictionary with the size $\mathcal{D} = 10,000$, we have $\mathcal{D}^2 = 10^8$ free parameters that need to be estimated which is far too many for small corpora.

To address the above weakness, we propose to constrain $W$ to be a sparse matrix with many zero entries for the pairs of words which are irrelevant for the preference learning task. If $W$ is a highly sparse matrix, then it consumes

[1]For the sake of clarity, we present the model in a text retrieval scenario. We use the term "words" for features, and "query" and "documents" for data instances, depending on their roles.

[2]In our model and algorithm, there is no need to restrict that query $q$ and document $d$ have the same feature size $\mathcal{D}$; however we make this assumption for simplicity of explanation.

much less memory and has only a limited number of free parameters to be estimated. In other words, a sparse $W$ matrix will allow us to greatly scale up the dictionary size to model those non-frequent words which are often essential for the preference ordering. In addition, we can have faster and more scalable learning algorithm since most entries of $W$ are zeros so that those multiplications can be avoided. Another advantage of learning a sparse representation of $W$ is its good interpretability. The zero $W_{ij}$ indicate that $i^{th}$ query feature and $j^{th}$ document feature are not correlated to the specific task. A sparse $W$ matrix will accurately capture correlation information between pairs of words and the learned correlated word pairs could explain the semantic rationale behind the preference ordering.

In order to enforce the sparsity on $W$, inspired by the success of the sparse linear regression model—"lasso" [6], we impose the entry-wise $\ell_1$ regularization on $W$. It has been shown theoretically that under certain conditions, $\ell_1$ regularization can correctly uncover the underlying ground truth sparsity pattern [7].

A practical challenge in using the $\ell_1$ regularized model is to develop an efficient and scalable learning algorithm. Since in many preference learning related applications (e.g. search engine), the model needs to be trained in a timely fashion and new (query, document) pairs may be added to the repository at any time, stochastic gradient descent in the online learning framework is the most desirable learning method for our task [8]. To enforce the $\ell_1$ regularization, based on [9], we propose to perform a mini-batch shrinking step for every $T$ iterations in the stochastic gradient descent which can lead to the sparse solution. Moreover, to reduce the additional bias introduced by the $\ell_1$ regularization, we further propose a *refitting* step which can improve the preference prediction while keeping the learned sparsity pattern.

The key idea of this paper is that: learning on a large number of corpus-independent raw features, or even on combinations of such features, is not impossible. Although the number of involved parameters is intimidatingly large, using sparsity as a powerful tool, the model can be well controlled and efficiently learned. We believe these ideas form a fresh perspective and will benefit many other retrieval related tasks as well.

## II. Basic Model

Let us denote the set of documents in the corpus as $\{d^k\}_{k=1}^K \subset \mathbb{R}^{\mathcal{D}}$ and the query as $q \in \mathbb{R}^{\mathcal{D}}$, where $\mathcal{D}$ is the dictionary size, and the $j^{th}$ dimension of a document/query vector indicates the frequency of occurrence of the $j^{th}$ word, e.g. using the tf-idf weighting and then normalizing to unit length [10].

Given a query $q$ and a document $d$, we wish to learn a scoring function $f(q, d)$ that measures the relevance of $d$ to $q$. In this paper, we assume that $f$ is a linear function which takes the form of Eq. (1). Each entry of $W$ represents a "relationship" between a pair of words.

### A. Margin Rank Loss

Suppose we are given a set of tuples $\mathcal{R}$ (labeled data), where each tuple contains a query $q$, a preferred document $d^+$ and an unpreferred (or lower ranked) document $d^-$. We would like to learn a $W$ such that $q^\top W d^+ > q^\top W d^-$, making the right preference prediction.

For that purpose, given tuple $(q, d^+, d^-)$, we employ the widely adopted margin rank loss [11]:

$$\begin{aligned} L_W(q, d^+, d^-) &\equiv h(q^T W d^+ - q^T W d^-) \\ &= \max(0, 1 - q^T W d^+ + q^T W d^-), \end{aligned} \tag{2}$$

where $h(x) \equiv \max(0, 1 - x)$ is the well-known hinge loss function as adopted in SVM.

Our goal is to learn the $W$ matrix which minimize the loss in Eq. (2) summing over all tuples $(q, d^+, d^-)$ in $\mathcal{R}$:

$$W^* = \arg\min_W \frac{1}{|\mathcal{R}|} \sum_{(q,d^+,d^-)\in\mathcal{R}} L_W(q, d^+, d^-). \tag{3}$$

### B. Stochastic Subgradient Descent

In general, the size of $\mathcal{R}$ is very large and new tuples may be added to $\mathcal{R}$ in a streaming manner, which makes it difficult to directly train the objective in Eq. (3). To overcome this challenge, we adopt stochastic (sub)gradient descent (SGD) in an online learning framework [12],

Specifically, at each iteration, we randomly draw a sample $(q, d^+, d^-)$ from $\mathcal{R}$, compute the subgradient[3] of $L_W(q, d^+, d^-)$ with respect to $W$ as following:

$$\nabla L_W(q, d^+, d^-) = \begin{cases} -q(d^+ - d^-)^\top & \text{if} \quad q^\top W(d^+ - d^-) < 1 \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

and then update the $W$ matrix accordingly. It has been shown that SGD achieves fast learning on large scale datasets [8].

We suggest to initialize $W^0$ to the identity matrix as this initializes the model to the same solution as a cosine similarity score. The strategy introduces a prior expressing that the weight matrix should be close to the identity matrix. We consider term correlations only when it is necessary to increase the score of a relevant document, or conversely, decrease the score of a irrelevant document. As for the learning rate $\eta_t$, we suggest to adopt a decaying learning rate: $\eta_t = \frac{C}{\sqrt{t}}$, where $C$ is a pre-defined constant as the initial learning rate. Intuitively, it should be better than the fixed learning rate since at the beginning, when $W$ is far away from the optimal solution $W^*$, a larger learning rate is desirable since it leads to a significant decrease of the objective value. On the other hand, when $W$ gets close to $W^*$, a smaller rate should be adopted to avoid missing the optimal solution. We will show the advantage of the

[3]Since $L$ is a non-smooth function, it does not have the gradient but only has the subgradient.

decaying learning rate scheme over the fixed one in the experiment section.

## III. Preference Learning with Sparsity

As discussed in the introduction, the model and the learning algorithm in the previous section will lead to a dense $W$ matrix which consumes a large amount of memory and has poor generalization ability for small corpora. To address these problems, we can learn a sparse model with a small number of nonzero entries of $W$. In order to obtain a sparse $W$, inspired by [6], we add an entry-wise $\ell_1$ norm to the $W$ as a regularization term to the loss function. We propose to optimize the following objective function:

$$W^* = \arg\min_W \frac{1}{|\mathcal{R}|} \sum_{(q,d^+,d^-)\in\mathcal{R}} L_W(q,d^+,d^-) + \lambda\|W\|_1, \tag{5}$$

where $\|W\|_1 = \sum_{i,j=1}^{\mathcal{D}} |W_{ij}|$ is the entry-wise $\ell_1$ norm of $W$ and $\lambda$ is the regularization parameter which controls the sparsity level (the number of nonzero entries) of $W$. In general, a larger $\lambda$ leads to a more sparse $W$. On the other hand, a too sparse $W$ will miss some useful relationship information among word pairs (considering diagonal $W$ as an extreme case). Therefore, in practice, we need to tune $\lambda$ to obtain a $W$ with a suitable sparsity level.

### A. Training the Sparse Model

To optimize Eq. (5), we adopt a variant of the general sparse online learning scheme in [9]. In [9], after updating $W^t$ at each iteration in SGD, a shrinkage step is performed by solving the following optimization problem:

$$\widehat{W}^t = \arg\min_W \frac{1}{2}\|W - W^t\|_F^2 + \lambda\eta_t\|W\|_1, \tag{6}$$

and then use $\widehat{W}^t$ as the starting point for the next iteration. In Eq. 6, $\|\cdot\|_F$ denote the matrix Frobenius norm and $\eta_t$ is the decaying learning rate for the $t^{th}$ iteration. According to the next proposition, we know that performing (6) will shrink those $W_{ij}^t$ with an absolute value less than $\lambda\eta_t$ to zero and hence lead to a sparse $W$ matrix.

**Proposition 1.** *The solution $\widehat{W}^t$ to the optimization problem in Eq.* (6) *takes the following form:*

$$\widehat{W}_{ij}^t = \begin{cases} W_{ij}^t + \lambda\eta_t & \text{if} \quad W_{ij}^t < -\lambda\eta_t \\ 0 & \text{if} \quad -\lambda\eta_t \le W_{ij}^t \le \lambda\eta_t \\ W_{ij}^t - \lambda\eta_t & \text{if} \quad W_{ij}^t > \lambda\eta_t \end{cases} \tag{7}$$

Although performing the shrinkage step leads a sparse $W$ solution, it is very expensive for a large dictionary size $\mathcal{D}$. For example, when $\mathcal{D} = 10,000$, we need $\mathcal{D}^2 = 10^8$ operations. Therefore, we suggest to perform the shrinkage step for every $T$ iteration cycles. In general, a smaller $T$ guarantees that the shrinkage step can be done in a timely fashion so that the entries of $W$ will not grow too large to produce inaccurate $\nabla L_W(q,d^+,d^-)$; on the other hand, a smaller $T$ increases the computational cost of the training process. In practice, we suggest to set $T = 100$. The details of the algorithm are presented in Algorithm 1.

Note that when $t$ is a multiple of $T$, we perform the shrinkage step with a cumulative regularization parameter for $\|W\|_1$ from $t-T+1$ to $t$: $\lambda\sum_{k=t-T+1}^{t}\eta_t$. The reason why we adopt cumulative regularization parameter is due to the following simple fact that: $W^T$ obtained by solving a sequence of successive optimization problems:

$$W^t = \arg\min_W \frac{1}{2}\|W - W_{t-1}\|_F^2 + \lambda\eta_t\|W\|_1, \quad t = 1,\dots T$$

is identical to the one by solving the following single optimization problem:

$$W^T = \arg\min_W \frac{1}{2}\|W - W_0\|_F^2 + \lambda\sum_{t=1}^{T}\eta_t\|W\|_1.$$

The above equivalence can be easily proved using Proposition 1 as shown in [9]. Although we take the gradient update so that Algorithm 1 is not exactly identical to the one taking the shrinkage step at every iteration, empirically, the learned sparse $W$ from Algorithm 1 is a good approximation.

**Algorithm 1** Sparse SGD

**Initialization**: $W^0 \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$, $T$, learning rate sequence $\{\eta_t\}$.
**Iterate** for $t = 1, 2, \dots$ until convergence of $W^t$:

1) Randomly draw a tuple $(q, d^+, d^-) \in \mathcal{R}$
2) Compute the subgradient of $L_{W^{t-1}}(q,d^+,d^-)$ with respect to $W$: $\nabla L_{W^{t-1}}(q,d^+,d^-)$
3) Update $W^t = W^{t-1} - \eta_t\nabla L_{W^{t-1}}(q,d^+,d^-)$
4) If ($t$ mod $T$ =0)

$$W^t = \arg\min_W \frac{1}{2}\|W - W^t\|_F^2 + \lambda\sum_{k=t-T+1}^{t}\eta_t\|W\|_1$$

### B. Refitting the Sparse Model

From Eq. (7), we see that $\ell_1$ regularization will not only shrink the weights for uncorrelated word pairs to zero but also reduce the absolute value of the weights for correlated word pairs. This additional bias introduced by $\ell_1$ regularization often harm the prediction performance. In order to reduce this bias, we propose to refit the model without $\ell_1$ regularization, but enforcing the sparsity pattern of $W$ obtained from Algorithm 1.

More precisely, after learning the sparse $\widehat{W}$ from Algorithm 1, let $\Omega$ be the indices of nonzero entries of $\widehat{W}$, i.e. $\Omega = \{(i,j)|\widehat{W}_{ij} \neq 0\}$. Given a matrix $W \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$, let $P_\Omega(W) \in \mathbb{R}^{\mathcal{D}\times\mathcal{D}}$ be the matrix defined as following:

$$P_\Omega(W)_{ij} = \begin{cases} W_{ij} & \text{if} \quad (i,j)\in\Omega \\ 0 & \text{if} \quad (i,j)\notin\Omega \end{cases},$$

where $P_\Omega$ is called the projection operator which projects $W$ onto $\Omega$.

In the refitting step, given $\Omega$, we try to minimize the following objective function:

$$W^* = \arg\min_W \frac{1}{|\mathcal{R}|} \sum_{(q,d^+,d^-)\in\mathcal{R}} L_{P_\Omega(W)}(q, d^+, d^-), \quad (8)$$

We still adopt SGD to minimize Eq. (8), but replace $\nabla L_W(q, d^+, d^-)$ with $\nabla L_{P_\Omega(W)}(q, d^+, d^-)$. Using the chain rule for subgradient, we can show that $\nabla L_{P_\Omega(W)}(q, d^+, d^-)$ takes the following form:

$$\nabla L_{P_\Omega(W)}(q, d^+, d^-) = \begin{cases} -P_\Omega(q(d^+ - d^-)^\top) & \text{if } q^\top P_\Omega(W)(d^+ - d^-) < 1 \\ 0 & \text{otherwise} \end{cases}.$$

In the experiment section, we show that the prediction performance gets improved after the refitting step.

## IV. Experiment

### A. Experiment Setup

Pairwise preference learning discussed in this paper belongs to a more general framework "Learning to rank", which is a key topic in the research of information retrieval [2]. Learning to rank methods are usually evaluated using standard benchmark data like TREC[4] data or LETOR [3]. However, TREC has only a limited number of queries available, which makes the training of a large number of features very difficult. On the other hand, LETOR and most of the other learning to rank datasets (e.g. Microsoft Learning to Rank Datasets[5], Yahoo! Learning to Rank Challenge Datasets[6]) have only few hundred (or even less) features such as BM25 or pagerank scores instead of the actual word features, and are therefore not adequate for evaluating our methods. It would be ideal to test the proposed method on click-through data from web search logs, but such data are not publicly available.

As pointed out by a seminal paper [13], preference learning and multiclass classification can be modeled in a unified framework. It is natural to adopt the multiclass classification (with many different classes) datasets to evaluate the our proposed preference learning models. More precisely, we construct training samples $(q, d^+, d^-) \in \mathcal{R}$ where $q$ and $d^+$ are in the same class while $q$ and $d^-$ belong to different classes. In our experiment, we use several benchmark multiclass classification datasets, including the text datasets 20 Newsgroups[7] (20NG), RCV1[8][15] and digital recognition dataset MNIST[9]. For 20NG and RCV1, we adopt the normalized tf-idf of the 10,000 most frequent words as the document features. For MINST, the normalized grey scale pixel values are used as features. Some statistics of these datasets are shown in Table I.

Table I
THE STATISTICS OF THE EXPERIMENTAL DATASETS

| | 20NG | RCV1 | MNIST |
|---|---|---|---|
| No. of training samples | 11,314 | 15,564 | 60,000 |
| No. of testing samples | 7,532 | 518,571 | 10,000 |
| No. of Classes | 20 | 53 | 10 |
| Dictionary Size $\mathcal{D}$ | 10,000 | 10,000 | 784 |
| No. of Free Parameters | $10^8$ | $10^8$ | $614,656$ |

For the experiments, we use the cosine similarity as the baseline, i.e. $W = I$ and mainly compare the following methods:

1) $W$ is a diagonal matrix.
2) $W$ is unconstrained and trained by SGD with the fixed learning rate $\eta = 0.01$.
3) $W$ is unconstrained and trained by SGD with the decaying learning rates $\eta_t = \frac{C}{\sqrt{t}}$ where $C = 200$.
4) $W$ is sparse and trained by Algorithm 1 with the decaying learning rates $\eta_t = \frac{C}{\sqrt{t}}$ where $C = 200$ and then perform the corresponding refitting step[10].

Note that for the decaying learning rate case, we try a wide range of starting rate $C$ and find that $C = 200$ can provide us the most rapid decrease of the objective value. So $C$ is set to be 200 through out the paper. As for the regularization parameter $\lambda$, when the dictionary size is large, say $\mathcal{D} = 10,000$ for the text data, we choose $\lambda$ which leads to the 5% to 10% density (percentage of nonzero entries) of $W$. When $\mathcal{D}$ is relatively small, say 748 for the MNIST dataset, we set $\lambda$ so that the density of $W$ is roughly 50%. This way of setting $\lambda$ provides us a sparse enough $W$ which has the advantage of the memory savings and being easy to interpret. In the meanwhile, we have enough nonzero entries of $W$ to guarantee a reasonably good preference learning performance on the testing datasets.

We compare the performance of each method by the following metrics:

1) Test error rate: for a testing tuple $(q, d^+, d^-)$, if $q^T W d^+ \leq q^T W d^-$, test error is increased by 1 and normalized by the test sample size.
2) Mean average precision (MAP) [16].

And we also provide the semantic information encoded in the $W$ matrix for the text data.

### B. Results

*1) Learning Curves:* It has been shown that test error rate is monotonic to area under ROC curve (AUC). In Figure 1, we show the curves of the test error rate and

[4]http://trec.nist.gov/

[5]http://research.microsoft.com/en-us/projects/mslr/

[6]http://learningtorankchallenge.yahoo.com

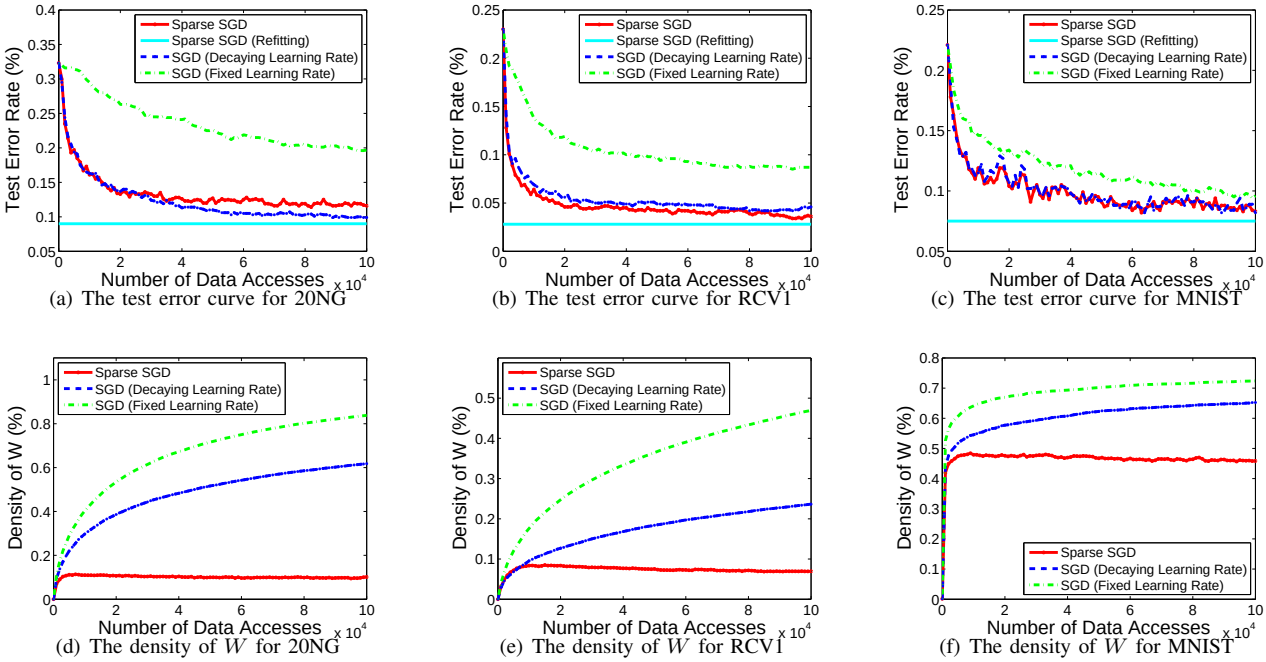
[7]http://people.csail.mit.edu/jrennie/20Newsgroups

[8]We adopt the preprocessing method in [14], remove the multi-labelled instances and result in 53 different classes.

[9]http://yann.lecun.com/exdb/mnist

[10]For the fair comparison, we use exactly the same training samples in the refitting step as in the sparse learning procedure without any additional samples.

Figure 1. The test error rate and the density $W$ for 3 benchmark datasets




(a) The test error curve for 20NG (b) The test error curve for RCV1 (c) The test error curve for MNIST

(d) The density of $W$ for 20NG (e) The density of $W$ for RCV1 (f) The density of $W$ for MNIST

the corresponding density of $W$ vs. the number of training iterations, i.e. the number of accesses of the training data. Each row in Figure 1 corresponds to one dataset.

We have the following observations:

1) For SGD with the fixed learning rate, the test error rate decreases very slowly, and is relatively flat compared to other methods.
2) The schemes with decaying learning rates (including two sparse models "Sparse SGD", and the dense model "SGD (Decaying Learning Rate)") have similarly good convergence rate.
3) Using the same regularization parameter, "Sparse SGD" achieves the most sparse model. For dense models, "SGD (Decaying Learning Rate)" reaches a more sparse model than using naive method "SGD (Fixed Learning Rate)". This is another evidence that decaying learning rates are superior to fixed learning rates.
4) Refitting the sparse models clearly achieves the best test error rate while keeping the low density of the $W$ matrix.

*2) Retrieval Performance and Memory:* In this section, we present mean average precision (MAP), test error rate and the memory space of $W$ of each method after training 100,000 iterations. The results are presented in Table II. We use the abbreviation for each method due to the space limitation of the table. "Identity" stands for $W = I$ and "Diagonal" means that $W$ is a diagonal matrix where only the diagonal entries are learned. 'SGD FLR" means SGD with the fixed learning rate, while "SGD DLR" means SGD with the decaying learning rate. Both of them are trained on the basic model without the $\ell_1$ regularization. "Sparse" and "Sparse-R" are sparse models, without refitting and with refitting, respectively.

The results on MAP are essentially consistent to those on the test error rate. The sparse method with decaying learning rate has similar performance compared to its dense counterpart but takes much less memory. Moreover, sparse models after refitting achieves the best performance

*3) Anecdotal Evidences:* The sparse model can also provide much useful semantic information. For example, we can easily infer the most related word pair between query word and document word from the sparse $W$ matrix. More precisely, each row of $W$ represents the strength of the correlation (either positive or negative) of document words to a specific query word. Given the $i^{th}$ query word, we sort the absolute value of the $i^{th}$ row of $W$ in a descending order and pick the first few nonzero entries. Those selected entries of $W$ represent the most correlated document words to the given $i^{th}$ query word.

In Table III, we present the query words from different categories and the five most correlated document words from the learned sparse $W$ in Eq. (5). We can see that most correlated document words are clearly from the same topics as the query words. It also provide us some interesting semantic information. For example, in 20NG, "fbi" is closely related to "handgun"; "colorado" to "hockey" which indicates that there might be a popular hockey team in Colorado (in fact,

Table II
RETRIEVAL PERFORMANCE AND MEMORY. ITEMS IN BOLD FONTS ARE THE BEST AMONG METHODS TESTED.

(a) 20NG

| | MAP | Error | Mem. (MB) |
|---|---|---|---|
| Identity | 0.185 | 0.323 | 0.2 |
| Diagonal | 0.190 | 0.318 | 0.2 |
| SGD FLR | 0.258 | 0.197 | 1294 |
| SGD DLR | 0.399 | 0.099 | 943.1 |
| Sparse | 0.360 | 0.114 | 154.2 |
| Sparse-R | **0.426** | **0.090** | 154.2 |

(b) RCV1

| | MAP | Error | Mem. (MB) |
|---|---|---|---|
| Identity | 0.380 | 0.230 | 0.2 |
| Diagonal | 0.390 | 0.223 | 0.2 |
| SGD FLR | 0.451 | 0.087 | 717.2 |
| SGD DLR | 0.453 | 0.046 | 360.2 |
| Sparse | 0.463 | 0.036 | 105.4 |
| Sparse-R | **0.501** | **0.029** | 105.4 |

(c) MNIST

| | MAP | Error | Mem. (MB) |
|---|---|---|---|
| Identity | 0.453 | 0.221 | 0.018 |
| Diagonal | 0.460 | 0.318 | 0.018 |
| SGD FLR | 0.610 | 0.096 | 6.796 |
| SGD DLR | 0.654 | 0.082 | 6.121 |
| Sparse | 0.654 | 0.083 | 4.301 |
| Sparse-R | **0.669** | **0.075** | 4.301 |

Table III
THE EXAMPLES OF LEARNED RELATED WORD PAIRS IN 20NG

| Query word | Five most related document words |
|---|---|
| clinton | clinton government health people gay |
| cpu | mac drive scsi card jon |
| graphics | graphics tiff image color polygon |
| handgun | gun weapons handgun militia fbi |
| hockey | hockey game espn colorado team |
| motorcycle | bike brake turbo rpi cylinder |
| religions | god religions bible christian jesus |

it is Colorado Avalanche team); "government" to "clinton" which indicates that Clinton might be a famous politician (The former US president Bill Clinton).

## V. CONCLUSIONS

In contrast to the traditional preference learning or "learning to rank" with a few hundred hand-crafted features, our basic model directly performs learning on actual words and considers their pairwise relationships between query and document. Although the pairwise relationship of words could improve and provide us additional semantic information, the basic model suffers from storage overloads and parameter overfitting. To overcome these drawbacks, we introduce sparsity to the model which is achieved by the $\ell_1$ regularization with an efficient online learning algorithm. We show with multiple benchmark datasets that our method achieves good performance with fast convergence, while remaining sparse during training (small memory consumption), on a model with hundreds of millions of parameters.

For future work, we will explore extended models to learn group structure or even hierarchical structure of the words using group lasso [17] type of regularization. The prior knowledge on the structure of $W$ can also be imposed.

## REFERENCES


[1] J. Fürnkranz and E. Hüllermeier, "Pairwise preference learning and ranking," in *ECML*, 2003.

[2] T.Y.Liu, *Learning to Rank for Information Retrieval*. Now Publishers Inc, 2009.

[3] T. Liu, J. Xu, T. Qin, W. Xiong, and H. Li, "Letor: Benchmark dataset for research on learning to rank for information retrieval," in *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, 2007.

[4] D. Grangier and S. Bengio, "A discriminative kernel-based approach to rank images from text queries," *IEEE Trans. PAMI.*, vol. 30, no. 8, pp. 1371–1384, 2008.

[5] B. Bai, J. Weston, R. Collobert, and D. Grangier, "Supervised semantic indexing," in *ECIR*, 2009.

[6] R.Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society, Series B*, vol. 58, pp. 267–288, 1996.

[7] M. J. Wainwright, "Sharp thresholds for noisy and high-dimensional recovery of sparsity using $\ell_1$-constrained quadratic programming (lasso)," *IEEE Tran on Information Theory*, vol. 55, pp. 2183–2202,, 2009.

[8] L. Bottou and Y. LeCun, "Large-scale on-line learning," in *Advances in Neural Information Processing Systems 15*. MIT Press, 2004.

[9] J. Duchi and Y. Singer, "Efficient learning using forward-backward splitting," in *Advances in Neural Information Processing Systems 23*, 2009.

[10] R. Baeza-Yates, B. Ribeiro-Neto *et al.*, *Modern information retrieval*. Addison-Wesley Harlow, England, 1999.

[11] R. Herbrich, T. Graepel, and K. Obermayer, *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA, 2000.

[12] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *ICML*, 2003.

[13] F.Aiolli and A. Sperduti, "Learning preferences for multiclass problems," in *Advances in Neural Information Processing Systems 18*, 2004.

[14] R. Bekkerman and M. Scholz, "Data weaving: Scaling up the state-of-the-art in data clustering," in *CIKM*, 2008.

[15] D. Lewis, Y.Yang, T. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization," *Journal of Machine Learning Research*, vol. 5, pp. 361–397, 2004.

[16] G. Cormack and T. Lynam, "Statistical precision of information retrieval evaluation," in *SIGIR*, 2006.

[17] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of Royal Statistical Society, Series B*, vol. 68(1), pp. 49–67, 2006.