Data Selection Techniques for Large-scale RankSVM

Ken-Yi Lin, Te-Kang Jan, Hsuan-Tien Lin
Dept of Computer Science & Information Engineering
National Taiwan University, Taipei, Taiwan
{*r97922117, r97922090, htlin*}*@csie.ntu.edu.tw*

***Abstract*—Learning to rank has become a popular research topic in several areas such as information retrieval and machine learning. Pair-wise ranking, which learns all the order preferences between pairs of examples, is a typical method for solving the ranking problem. In pair-wise ranking, RankSVM is a widely-used algorithm and has been successfully applied to the ranking problem in the previous work. However, RankSVM suffers from the critical problem of long training time needed to deal with a huge number of pairs. In this paper, we propose a data selection technique, Pruned RankSVM, that selects the most informative pairs before training. Experimental results show that the performance of Pruned RankSVM is on par with RankSVM while using significantly fewer pairs.**

***Keywords*-Keywords: learning to rank, pair-wise ranking, RankSVM, data selection technique**

I. Introduction

Ranking aims at scoring or ordering a group of items according to their relevance, and thus can be used to filter out key information from a huge amount of data. In this age of information explosion, ranking can be utilized in many applications in a wide range of domains. For instance, a document retrieval system needs to return a ranked list of documents, ordered by relevance with respect to a given query from a user [2, 16]; an expert finding system needs to return a few relevant experts with respect to some given tasks [1, 17]; and a collaborative filtering system needs to return some relevant items with respect to users' rating history [14, 9]. Thus, ranking is attracting a lot of research attention related to machine learning and information retrieval.

There are three typical categories of ranking methods [12]: point-wise [5], pair-wise [8, 10], and list-wise [3]. Each category tackles the ranking problem from a different angle. In terms of computation, for N items to be ranked, list-wise ranking globally considers all $O(N!)$ possible orderings, which is often difficult to tackle because of exponential dependence on N. Pair-wise ranking only checks $O(N^2)$ possible pair-wise comparisons on the N items locally, which leads to only a quadratic dependence on N. Point-wise ranking further tries to simply score each of the N examples individually, which results in a linear dependence on N. Arguably, pair-wise ranking captures the essence of ranking—the comparisons— better than point-wise ranking, and thus is often preferred in many ranking applications. Nevertheless, its quadratic dependence on N can inhibit the use of pair-wise ranking on large-scale datasets.

RankSVM is a popular pair-wise ranking approach [8, 10]. In practice, the performance of RankSVM is promising, but it suffers from the quadratic number of pairs generated with respect to the number of examples. In this paper, we propose a data selection technique that reduces the time RankSVM needs for training. The technique pre-selects important pairs before feeding them to regular SVM training. We demonstrate that the those pre-selected pairs correspond to potential support vectors for RankSVM and thus maintains the critical ranking information. After applying the data selection technique, the resulting algorithm, called Pruned RankSVM, uses fewer pairs than the original RankSVM, while maintaining a similar level of performance in practice. This property makes Pruned RankSVM preferable to RankSVM in practice.

The remainder of this paper is organized as follows. In Section II, we briefly describe the ranking problems and given an overview of RankSVM. In Section III, we introduce our proposed data selection technique that improves on the training speed of RankSVM and present the results of experiments conducted using it. In Section IV, we further improve the technique to make it effective for both datasets with low noise and datasets with high noise. Finally, we conclude this paper in Section V.

II. Pair-wise Ranking

In this section, we first describe the ranking problem and introduce the pair-wise ranking approach. We then discuss the well-known RankSVM algorithm.

A. The Ranking Problem

Let $\mathcal{X} \subseteq \mathbb{R}^d$ be the examples space and $\mathcal{Y} \subseteq \mathbb{R}$ be the preference labels space. Consider two examples $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ with $y_i, y_j \in \mathcal{Y}$ denoting their preference labels. We can determine $\mathbf{x}_i \succ \mathbf{x}_j$ if $y_i > y_j$, which means the preference for $\mathbf{x}_i$ is higher than that for $\mathbf{x}_j$. Suppose there is a set $\mathcal{F}$ of ranking functions that can be used to score each example, the ranking problem aims at finding a ranking function that preserves the order of the preferences instead of exactly predicting the preference labels. That is, a perfect ranker $f_p \in \mathcal{F}$ would ideally satisfy

$$\mathbf{x}_i \succ \mathbf{x}_j \quad \Leftrightarrow \quad f_p(\mathbf{x}_i) > f_p(\mathbf{x}_j) \;\; \forall i, j.$$

For any $f \in \mathcal{F}$, $f(\mathbf{x})$ is considered the ranked score of $\mathbf{x}$ in prediction. Assume there is an unknown distribution $\mathcal{D}$ that generates examples associated with their preference labels, $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$. Consider a training set $\mathcal{D}_t = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N}$ that is drawn i.i.d from $\mathcal{D}$. In the ranking problem, a commonly used loss function is

$$e(f(\mathbf{x}_1), f(\mathbf{x}_2), y_1, y_2) = [\text{sgn}(f(\mathbf{x}_1) - f(\mathbf{x}_2)) \neq \text{sgn}(y_1 - y_2)],$$

where the Boolean test $[\cdot]$ is one if the condition is true and zero otherwise; $\text{sgn}(v)$ is one if $v \geq 0$ and -1 if $v < 0$. Then, we define the test ranking error as

$$L(f) = \underset{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \sim \mathcal{D}}{\mathcal{E}} [e(f(\mathbf{x}_1), f(\mathbf{x}_2), y_1, y_2)].$$

The goal of the ranking problem is to learn a ranking function $f \in \mathcal{F}$ with $\mathcal{D}_t$ such that $L(f)$ with respect to $\mathcal{D}$ is as small as possible.

B. The Pair-wise Ranking Approach

The key idea underlying pair-wise ranking is learning of a ranking function for each of the $\binom{N}{2}$ pairwise preferences between two examples. Most formally, for each possible pair of examples $(\mathbf{x}_i, \mathbf{x}_j)$, we learn a ranking function f that predicts for any given example whether $\mathbf{x}_i \succ \mathbf{x}_j$ or $\mathbf{x}_j \succ \mathbf{x}_i$. This function is trained by pairs for every two examples, and the label of the pair is either positive (preferred) or negative (non-preferred). Existing pair-wise methods include RankSVM [8, 10] and RankBoost [7].

C. RankSVM

RankSVM is a famous pair-wise ranking algorithm that feeds the pairs to a regular binary classification SVM solver. The details are described as follows. Let $f \in \mathcal{F}$ be a linear ranking function in a transformed space,

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

where $\mathbf{w}$ is a weight vector and ϕ is a function that maps $\mathbf{x}$ to a higher dimensional space. With function ϕ, RankSVM can use a linear hyperplane $\mathbf{w}$ in the higher dimensional space to do the ranking. It is hoped that f preserves the partial order preference. Let $\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)$ represent a pair (i, j) in pair-wise ranking. We can classify this pair (i, j) into two categories: preferred $(\mathbf{w}^T(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) > 0)$ or non-preferred $(\mathbf{w}^T(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) < 0)$. Consequently, we can use a regular binary classification SVM solver to solve the ranking problem. Assume that l pairs are generated, the pair-wise training set for RankSVM is

$$\mathcal{D}_o = \{(\mathbf{X}_p, z_p)\}_{p=1}^{l}$$

where each p represents a pair (i, j), $\mathbf{X}_p = \phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)$,

$$z_p = \begin{cases} +1 & \text{if } y_i > y_j \\ -1 & \text{if } y_i < y_j \end{cases}.$$

Note that for each pair (i, j), if we can rank $(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j), 1)$ correctly, then we can also rank $(\phi(\mathbf{x}_j) - \phi(\mathbf{x}_i), -1)$ correctly. For convenience, we only consider positive pairs in RankSVM. RankSVM solves the primal problem as follows:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{p=1}^{l} \xi_p \\ \text{subject to} \quad & \mathbf{w}^T(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) \geq 1 - \xi_p, \\ & \xi_p \geq 0, \ p = 1, \ldots, l \end{aligned} \tag{1}$$

where C is a parameter that adjusts the trade-off between the margin and the pair-wise violations and ξ_p is the slack variable associated with margin violation for pair difference $(\mathbf{x}_i, \mathbf{x}_j)$.

III. Closest RankSVM

In this section, we first present a data selection technique that identifies the potential support vectors in RankSVM then propose Closest RankSVM which considers these potential support vectors to be the most critical ones and uses only these pairs to train. Because Pruned RankSVM uses the ordering relationships to train the ranking function, it loses the numerical information associated with preference label y. Hence, we seek to include the numerical information to see whether it can improve the performance.

A. Informative Pairs in RankSVM

Eq.(1) indicates that RankSVM treats all the pairs equally. Nevertheless, it is intuitive that some pairs might be more important in the training process. In RankSVM, the ranking function is represented by only the support vectors. That is, the ranking function would still be the same if we removed those non-support vectors before optimization. This concept motivates us to search the support vectors in RankSVM. In Section II-C we introduced the primal problem of (soft-margin) RankSVM, which allows the pair-wise violations by using $\sum_{p=1}^{l} \xi_p$. A variant that does not allow any violations is called the hard-margin RankSVM, which solves the following primal problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & \mathbf{w}^T(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) \geq 1. \end{aligned}$$

Let us first consider hard-margin RankSVM. Assume that the dataset is perfectly rankable. That is, hard-margin RankSVM has a feasible solution. Consequently, the support vectors must satisfy the equation $\mathbf{w}^T(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) = 1 \Leftrightarrow \mathbf{w}^T\phi(\mathbf{x}_i) - \mathbf{w}^T\phi(\mathbf{x}_j) = 1$ (score difference $= 1$). Because all the other pairs must satisfy $\mathbf{w}^T(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)) \geq 1 \Leftrightarrow \mathbf{w}^T\phi(\mathbf{x}_i) - \mathbf{w}^T\phi(\mathbf{x}_j) \geq 1$ (score difference ≥ 1), we can conclude that the support vectors in RankSVM are the score-closest pairs when using $\mathbf{w}$ to rank. Our goal is to find informative pairs and use only those pairs for training. Obviously, support vectors can be considered the most informative pairs (support vectors form the ranking function). However, we cannot determine which pair is the support vector in RankSVM because $\mathbf{w}$ is unknown before training. To find support vectors (closest pairs), we analyze the ranking problem below.

Assume that we are given a training ranked list $\mathbf{x}_1, \ldots, \mathbf{x}_N$, where $\mathbf{x}_1 \succ \ldots \succ \mathbf{x}_N$, and we define level r pairs as the pairs that contain examples with order difference r. For instance, level 1 pairs: $(\phi(\mathbf{x}_1) - \phi(\mathbf{x}_2), 1), \ldots, (\phi(\mathbf{x}_{N-1}) - \phi(\mathbf{x}_N), 1)$, and level 2 pairs: $(\phi(\mathbf{x}_1) - \phi(\mathbf{x}_3), 1), \ldots, (\phi(\mathbf{x}_{N-2}) - \phi(\mathbf{x}_N), 1)$. Suppose the ranked list is perfectly rankable by some $\mathbf{w}$ using a hard-margin RankSVM, then it is not hard to prove that the score-closest pairs must be within level 1.

Note that the support vectors are the score-closest pairs when using $\mathbf{w}$ to rank in a hard-margin RankSVM. Thus, only level 1 pairs could be the support vectors. Therefore, we define level 1 pairs as the most informative pairs in RankSVM and call them 1-closest pairs. Since we have defined the informative pairs, we may be able to reduce the problem size before training. However, we do not know for sure whether 1-closest pairs would still be important for soft-margin RankSVM, especially for the noisy data that cannot be perfectly ranked.

Consequently, we design an experiment by examining the support vector distribution of RankSVM. We use LIBSVM [4] with the RBF[1] kernel and LIBLINEAR [6] as the solvers for the experiments. Our experimental dataset is bodyfat, from the StatLib Datasets Archive [13].

1) Support Vector Distribution: The dual problem of RankSVM is

$$\begin{aligned}
\min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{Q} \boldsymbol{\alpha} - \mathbf{e}^T \boldsymbol{\alpha} \\
\text{subject to} \quad & 0 \le \alpha_p \le C,\ p = 1, \ldots, l \\
\text{where} \quad & \boldsymbol{Q}:\ l \text{ by } l \text{ matrix} \\
& Q_{mg} = \mathbf{X}_m{}^T \mathbf{X}_g,\ \mathbf{e} = [1, \ldots, 1]^T \\
& \mathbf{X}_m = \phi(\mathbf{x}_i) - \phi(\mathbf{x}_j),\ \mathbf{X}_g = \phi(\mathbf{x}_a) - \phi(\mathbf{x}_b) \\
& (i, j) \text{ and } (a, b) \text{ are two pairs in pair-wise} \\
& \text{ranking.} \\
\text{At optimum,} \quad & \mathbf{w} = \sum_{p=1}^{l} \alpha_p \mathbf{X}_p.
\end{aligned}$$

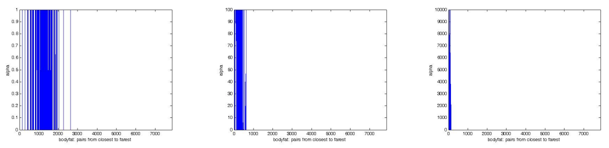
In RankSVM, the corresponding α_p of the non-support vector is zero. In other words, if we remove those non-support vector pairs, $\mathbf{w}$ will still be the same. We would subsequently get the same ranked score for every example. We designed our experiment as follows:

1) Train a ranking function using total pairs ($\mathcal{D}_o$).

[1]RBF: $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0$

2) Plot α_p with p ordered from order-closest to farthest.

The results for the RBF kernel ($C = 1, \gamma = 1/\text{number of features}$) are shown in Figure 1(a) (the results for the linear kernel are similar). In the figures, the horizontal axis represents the pair index from order-closest to farthest and the vertical axis represents the value of α_p. The bodyfat dataset generated 138 1-closest pairs and 7862 total pairs. In Figure 1, it can clearly be seen that the support vectors ($\alpha_p \neq 0$) appear only in the left part of the figures and that they move to the left with larger C. Because these non-support vectors ($\alpha_p = 0$) do not affect the ranking function, we can use much fewer pairs to achieve similar performance by discarding them.



(a) C=1 (b) C=100 (c) C=10000

Figure 1. Support vector distribution on various C with respect to order on the bodyfat dataset

B. Closest RankSVM

In Section III-A, we defined the 1-closest pairs as the most informative pairs by assuming that the examples can be perfectly ranked in hard-margin RankSVM. Unfortunately, in Figure 1, it appears that the 1-closest pairs (the first 138 pairs in the figures) are not enough to train a ranking function. Note that the support vectors move to the left with larger C (become fewer). Training with a larger C signifies that we are more concerned about the pair-wise violations than the margin in RankSVM. Specifically, benefits may be derived from correctly ranking the closest pairs might have the benefit. Take the ranked list in Section III-A for instance. If the 1-closest pairs can be ranked correctly by some $\mathbf{w}$, then

$$\begin{aligned}
& \mathbf{w}^T\phi(\mathbf{x}_1) > \mathbf{w}^T\phi(\mathbf{x}_2), \ldots, \mathbf{w}^T\phi(\mathbf{x}_{N-1}) > \mathbf{w}^T\phi(\mathbf{x}_N) \\
\Longrightarrow \quad & \mathbf{w}^T\phi(\mathbf{x}_1) > \ldots > \mathbf{w}^T\phi(\mathbf{x}_{N-1}) > \mathbf{w}^T\phi(\mathbf{x}_N). \qquad (2)
\end{aligned}$$

It can be easily derived that the total pairs will also be ranked correctly. Thus, we propose Closest RankSVM in Algorithm 1.

```
Algorithm 1 Closest RankSVM
  1) Sort the examples according to the preference labels.
  2) Generate the pairs from order-closest to farthest.
  3) Use closest $S$ pairs to train the ranking function.
```

Closest RankSVM can be thought of as a technique that selects the order-closest pairs for training. The objective of Closest RankSVM is to rank the closest S pairs, whereas that of RankSVM is to rank the total pairs. Our intuition is that ranking the order-closest pairs correctly would imply the correctness of the farthest pairs, in which case we would save a lot of time because we would be using much fewer pairs in training. Next, we present our experimental results, which are divided into two parts : results of experiments conducted on artificial data and results of experiments conducted on the real-world data.

1) Experiment on Artificial Data: We first conducted our experiment using artificial data. Each feature of $\mathbf{x}$ and the perfect ranker $\mathbf{w}'$ were randomly generated within $[-1, 1]$. The preference label

$$y = \mathbf{w}'^T \mathbf{x} + \text{“Gaussian noise”}$$

where the variances of the Gaussian noise are 0, 0.05, 0.1, 0.5, and 1 with centers at 0, respectively. For each experiment, we generated 200 training examples, 200 testing examples, and 150 validation examples. We trained the ranking function with linear kernel, in which C was searched within $\{2^{-5}, 2^{-3}, \ldots, 2^{13}, 2^{15}\}$ using the validation set. Twenty rounds were repeated for the experiments. Our evaluation criterion was the pair-wise accuracy. Let concordant pairs denote the pairs that are ranked correctly, that is, $\mathbf{w}^T\phi(\mathbf{x}_i) > \mathbf{w}^T\phi(\mathbf{x}_j)$ if $\mathbf{x}_i \succ \mathbf{x}_j$; and discordant pairs denote the pairs being mis-ranked, that is, $\mathbf{w}^T\phi(\mathbf{x}_i) < \mathbf{w}^T\phi(\mathbf{x}_j)$ if $\mathbf{x}_i \succ \mathbf{x}_j$. Then,

$$\text{pair-wise accuracy} = \frac{n_c}{n_c + n_d}$$

where n_c is the number of concordant pairs and n_d is the number of discordant pairs. This is very similar to Kendall's tau (τ) coefficient [11], which is typically used to measure the association between two ranked lists.

$$\tau = \frac{n_c - n_d}{n_c + n_d} = 2 \times \text{pair-wise accuracy} - 1$$

For the benchmark algorithm, as mentioned in Section I, we used regression, an efficient method in point-wise ranking, to directly predict the preference labels. We chose Support Vector Regression (SVR) [15] as our regression algorithm for fair comparison. To demonstrate its effectiveness, we used only 1-closest pairs for Closest RankSVM. SVR uses $2N$ Lagrange multiplier in its dual for training, which is size of $O(N)$. In order to verify the efficacy of Closest RankSVM, we compared it with SVR and RankSVM (using total pairs $\mathcal{D}_o$) in the experiment. Further, to show the importance of closest pairs, we also compared it with "random pairs," which is used to train a ranking function using randomly selected pairs from $\mathcal{D}_o$.

Low noise : In Table I, it can be seen that the pair-wise accuracy between total pairs and 1-closest pairs are about the same. However, we can train much faster using 1-closest pairs than using total pairs. For a low noise dataset, it is also easy to rank the 1-closest pairs well. Thus, in accordance with Eq.(2), we can also rank the total pairs well. Closest RankSVM also outperforms random pairs. That is, 1-closest pairs are more important than random pairs in a low noise dataset. Note that SVR is also good, but it solves a size-$2N$ (N is 200 in the experiments) problem. Closest RankSVM uses only $(N-1)$ 1-closest pairs.

Dataset	1-closest (199 pairs)	Random (199 pairs)	Total pairs (19900 pairs)	SVR (400 Lagrange multiplier)
Noiseless	**99.94% ± 0.01%**	98.30% ± 0.17%	**99.94% ± 0.01%**	99.29% ± 0.04%
0.05	**98.45% ± 0.05%**	97.24% ± 0.22%	**98.48% ± 0.06%**	98.40% ± 0.06%
0.1	96.73% ± 0.18%	95.72% ± 0.21%	**96.91% ± 0.17%**	**96.93% ± 0.16%**

Table I
CLOSEST RANKSVM ON ARTIFICIAL DATA : LOW NOISE

High noise : Correctly ranking the 1-closest pairs is much more difficult in the high noise datasets. In Table II, it can be seen that Closest RankSVM performs slightly worse than random pairs and SVR. If we cannot rank the 1-closest pairs correctly, we lose some order information. In other words, for the high noise data, using only 1-closest pairs might not be enough. The problem is that 1-closest pairs may carry misleading information about the ranking, which motivated us to carry out the improvements on Closest RankSVM discussed in the next section.

Dataset	1-closest (199 pairs)	Random (199 pairs)	Total pairs (19900 pairs)	SVR (400 Lagrange multiplier)
0.5	84.01% ± 0.63%	84.78% ± 0.49%	**85.95% ± 0.47%**	**85.82% ± 0.48%**
1	71.33% ± 1.29%	73.64% ± 0.81%	**74.79% ± 0.73%**	**74.66% ± 0.77%**

Table II
CLOSEST RANKSVM ON ARTIFICIAL DATA : HIGH NOISE

IV. PRUNED RANKSVM WITH MIXED PAIRS

In this section, we first propose an improved data selection technique, Pruned RankSVM, and examine its performance. Next, to demonstrate that Pruned RankSVM can train a ranking function more efficiently, we compare it with Closest RankSVM in Section IV-B.

A. Pruned RankSVM

In Section III, we showed that Closest RankSVM adds a large number of pairs for the high noise datasets. In an effort to design a better data selection technique for high noise data, we analyzed the ranking problem of various noise levels (from low to high noise). We ran the total RankSVM, and identified the pairs that are actually used during the optimization process for the bodyfat dataset with noise=0, noise=0.1 and noise=1. The results obtained are shown in Figure 2. C was set to 1000 in these experiments.

As can be seen for the higher noise data, there are more support vectors and the 1-closest pairs (the green parts in the figures) became less important. In the high noise datasets, random pairs are more representative of the total pairs than the 1-closest pairs. If we do not want to lose most of the total order preferences information, then using random pairs is the better choice. Note that the left parts (order-closest pairs) are still more important in the figures. Therefore, we propose Pruned RankSVM, which uses S closest pairs

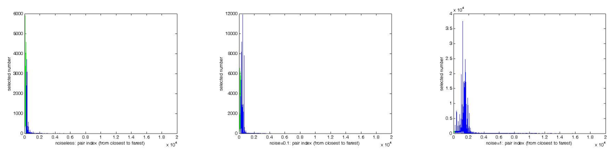

(a) noiseless (b) noise=0.1 (c) noise=1

Figure 2. Optimization on various noise datasets

plus S random ones in Algorithm 2. Next, we test the performance of Pruned RankSVM.

Algorithm 2 Pruned RankSVM

```
1) Sort the examples according to the preference labels.
2) Generate the pairs from order-closest to farthest.
3) Use closest
```

1) Pruned RankSVM on Artificial Data: We use the closest S + random S pairs for Pruned RankSVM. The experimental obtained using results on artificial data are shown in Table III. In order to demonstrate the benefit of this mixed data selection technique, we compared "closest S" with "closest S + random S". It can clearly be seen that adding random S to closest S pairs can significantly improve the performance for the high noise datasets, especially for the one with the highest noise one (noise=1). Further, it can be seen that Pruned RankSVM performs well on both low and high noise datasets. Note that SVR slightly outperforms Pruned RankSVM on the high noise datasets; however, we can still improve the performance of Pruned RankSVM by adding more pairs. We show the improvement obtained on the real-world datasets in the next section.

Dataset	Closest S (199 pairs)	Closest S + random S (398 pairs)	Total pairs (19900 pairs)	SVR (400 Lagrange multiplier)
noiseless	$\mathbf{99.94\% \pm 0.01\%}$	$\mathbf{99.94\% \pm 0.01\%}$	$\mathbf{99.94\% \pm 0.01\%}$	99.29% ± 0.04%
0.05	$\mathbf{98.45\% \pm 0.05\%}$	$\mathbf{98.44\% \pm 0.05\%}$	$\mathbf{98.48\% \pm 0.06\%}$	98.40% ± 0.06%
0.1	96.73% ± 0.18%	96.74% ± 0.17%	$\mathbf{96.91\% \pm 0.17\%}$	$\mathbf{96.93\% \pm 0.16\%}$
0.5	84.01% ± 0.63%	85.31% ± 0.54%	$\mathbf{85.95\% \pm 0.47\%}$	$\mathbf{85.82\% \pm 0.48\%}$
1	71.33% ± 1.29%	73.79% ± 0.82%	$\mathbf{74.79\% \pm 0.73\%}$	$\mathbf{74.66\% \pm 0.77\%}$

Table III
PRUNED RANKSVM ON ARTIFICIAL DATA

2) Pruned RankSVM on Real-world Data: In this section, we discuss the results obtained when RankSVM was applied to real-world datasets in this section. In Tables IV and V, it can be seen that Pruned RankSVM outperforms SVR and the pair-wise accuracy between Pruned RankSVM and RankSVM (using total pairs $\mathcal{D}_o$) are overall comparable on most of the datasets. However, Pruned RankSVM uses much fewer pairs than RankSVM (see Table VI). We also show that we can improve the performance by using more pairs ("closest $2S$ + random $2S$" outperforms "closest S + random S"). For large-scale datasets, it is difficult to train a ranking function using total pairs such as "mg" and "space_ga." Our goal in the data selection technique is to find more informative pairs and use only those pairs for training. The experimental results show that the data selection technique can use the critical partial pairs to achieve a good performance for large-scale datasets.

Dataset	Closest S + random S	Closest $2S$ + random $2S$	Total pairs	SVR
mg	77.45% ± 0.07%	77.44% ± 0.16%	77.56% ± 0.16%	$\mathbf{77.73\% \pm 0.04\%}$
space_ga	78.22% ± 0.13%	$\mathbf{78.71\% \pm 0.12\%}$	-	78.28% ± 0.13%
bodyfat	$\mathbf{99.25\% \pm 0.16\%}$	98.83% ± 0.09%	99.13% ± 0.03%	96.37% ± 0.12%
housing	86.43% ± 0.10%	86.57% ± 0.26%	$\mathbf{86.90\% \pm 0.03\%}$	85.74% ± 0.04%
mpg	90.07% ± 0.25%	90.34% ± 0.20%	$\mathbf{90.71\% \pm 0.09\%}$	90.40% ± 0.02%

('-' means that there were too many pairs, so we could not finish the training process)

Table IV
PRUNED RANKSVM: LINEAR KERNEL

Dataset	Closest S + random S	Closest $2S$ + random $2S$	Total pairs	SVR
mg	80.80% ± 0.20%	$\mathbf{81.18\% \pm 0.19\%}$	-	80.32% ± 1.11%
space_ga	81.65% ± 0.13%	$\mathbf{82.12\% \pm 0.12\%}$	-	79.13% ± 0.24%
bodyfat	96.92% ± 0.13%	$\mathbf{98.21\% \pm 0.12\%}$	$\mathbf{97.87\% \pm 0.61\%}$	96.01% ± 0.15%
housing	87.35% ± 0.27%	88.16% ± 0.33%	$\mathbf{89.19\% \pm 0.32\%}$	87.25% ± 0.27%
mpg	90.03% ± 0.30%	$\mathbf{90.73\% \pm 0.28\%}$	$\mathbf{90.99\% \pm 0.28\%}$	$\mathbf{90.93\% \pm 0.11\%}$

('-' means that there were too many pairs, so we could not finish the training process)

Table V
PRUNED RANKSVM: RBF KERNEL

Dataset	Closest S + random S	Closest $2S$ + random $2S$	Total pairs	SVR
mg	1388.2 ± 0.45	2776.4 ± 0.91	239083 ± 0.22	1384
space_ga	3107.5 ± 0.41	6216.4 ± 1.08	1205130 ± 0.20	3106
bodyfat	270.1 ± 1.41	540.2 ± 2.82	7864.9 ± 0.64	252
housing	815.4 ± 6.60	1630.8 ± 13.2	31721.5 ± 2.66	506
mpg	1012.2 ± 22.1	2024.4 ± 44.2	18767 ± 7.42	392

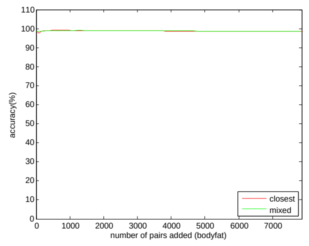
Table VI
SIZE OF THE OPTIMIZATION PROBLEM

B. Comparison between Closest and Pruned RankSVM

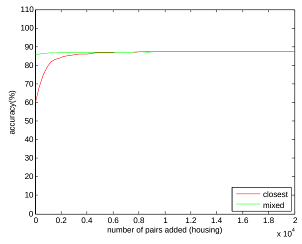
In order to show that Pruned RankSVM can train a ranking function more efficiently than Closest RankSVM, we conducted the experiment using the real-word datasets, bodyfat, housing, and mixmpg. Figures 3(a), 3(b) and 3(c) show that both Closest RankSVM and Pruned RankSVM can train a good ranking function by adding pairs. However, Pruned RankSVM adds a relatively smaller number of pairs before achieving satisfactory performance.

V. CONCLUSION

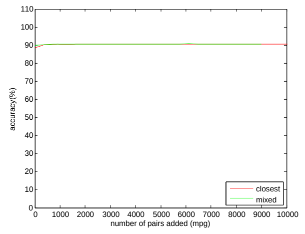
In this paper, we first proposed a data selection technique Closest RankSVM, that discovers the most informative pairs for pair-wise ranking. We then presented the experimental results showing that only a small part of the total order preferences is important—signifying that we can reduce the problem size before training. Consequently, we further improved our technique by analyzing the datasets from low to high noise and then subsequently proposed Pruned RankSVM. Then, by means of experiments conducted, we showed that Pruned RankSVM can train a ranking function on par with RankSVM while using significantly fewer pairs. We therefore make the following recommendations:



(a) bodyfat



(b) housing



(c) mpg

Figure 3. Comparison between mixed and closest in different datasets

1) When enough resources are available for RankSVM, using RankSVM can train an excellent ranking function.
2) When enough resources are available for Pruned RankSVM but not enough for RankSVM, then using Pruned RankSVM can train a ranking function on par with RankSVM and saves a lot of time.
3) When not enough resources are available for RankSVM and Pruned RankSVM, then we suggest using regression because of its efficiency.

References

[1] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *Proceedings of the 29th Annual International ACM SIGIR Conference*, pages 43–50. ACM Press, 2006.

[2] James P. Callan. Passage-level evidence in document retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310. Springer-Verlag New York, Inc., 1994.

[3] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th Annual International Conference on Machine Learning*, pages 129–136. ACM Press, 2007.

[4] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] David Cossock and Tong Zhang. Subset ranking using regression. In *Proceedings of the 19th Annual Conference on Learning Theory*, pages 605–619. Springer, 2006.

[6] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[7] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[8] Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, 2000.

[9] J. Herlocker, J. Konstan, A Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 230–237. ACM Press, 1999.

[10] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th Annual International ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM Press, 2002.

[11] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–93, 1938.

[12] Tie-Yan Liu. Learning to rank for information retrieval. In Fabio Crestani, Stéphane Marchand-Maillet, Hsin-Hsi Chen, Efthimis N. Efthimiadis, and Jacques Savoy, editors, *SIGIR*, page 904. ACM, 2010.

[13] K. Pace and R. Johnson. StatLib-Datasets Archive, 2010.

[14] Denis Parra and Peter Brusilovsky. Collaborative filtering for social tagging systems: An experiment with CiteULike. In *Proceedings of the 3rd ACM Conference on Recommender Systems*, pages 237–240. ACM Press, 2009.

[15] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[16] Tian Weixin and Zhu Fuxi. Learning to rank using semantic features in document retrieval. In *Proceedings of the 2009 (1st) WRI Global Congress on Intelligent Systems*, pages 500–504. IEEE Computer Society, 2009.

[17] Dawit Yimam. Expert finding systems for organizations: Domain analysis and the demoir approach. In *ECSCW 99 Beyond Knowledge Management: Management Expertise Workshop*, pages 276–283. MIT Press, 2000.