Relevance Vector Ranking for Information Retrieval

[1] Fengxia Wang, [2] Huixia Jin, [3] Xiao Chang
[1, *3]*Department of Computer Science and Technology*
Xi'an Jiaotong University,710049, China
wangfengxia@gmail.com，*changxiao66@gmail.com*
[2] *Department of Physics and Telecom Engineering*
Hunan City University,Yiyang, 413008, China
jinhuixia1980@163.com
doi:10.4156/jcit.vol5. issue9.12

Abstract

In recent years, learning ranking function for information retrieval has drawn the attentions of the researchers from information retrieval and machine learning community. In existing approaches of learning to rank, the sparse prediction model only can be learned by support vector learning approach. However, the number of support vectors grows steeply with the size of the training data set. In this paper, we propose a sparse Bayesian kernel approach to learn ranking function. By this approach accurate prediction models can be derived, which typically utilize fewer basis functions than the comparable SVM-based approaches while offering a number of additional advantages. Experimental results on document retrieval data set show that the generalization performance of this approach competitive with two state-of-the-art approaches and the prediction model learned by it is typically sparse.

Keywords*: Learning to Rank, Relevance Vector Learning, Sparse Model*

1. Introduction

Ranking is the central problem for information retrieval. The retrieval results can be rated by giving the grades to the results on the relevant to user's query. The similarity between user's query and document is used to rank the documents. The technique is proposed to eliminate the affixes and their effects on recognizing similar Persian documents [1]. Content-Time-based Ranking algorithm combines keywords, update time and content time of Web page into the ranking procedure [2]. In practice, the instances are ranked by mapped them to the score with ranking function. The task of learning to rank is to find a model on samples data, which can help to predict the order of new instances.

Supported vector machine (SVM) methodology was introduced to develop ranking algorithms[3] [4]. A preference learning algorithm based on regularized least squares is proposed in [5]. RankBoost [6] works by combining many "weak" rankings of the given instances. RankNet [7] use a probabilistic cost function on instances pairs and model the underlying ranking function using a two layeres neural network. In [8], proposed a loss function named Fidelity to measure loss of ranking and adopted a generalized additive model, similar to the boosting approach, to learn a ranking function. A Bayesian framework to preference learning, which based on Gaussian processes is proposed in [9]. Conditional independence tree is introduced to ranking,which is the combination of decision tree and naive Bayes [10]. A Ranking tree algorithm is proposed based on decision tree [11]. Naive Bayesian is utilized to ranking in [12] and the experiments show that naive Bayes has some advantage over C4.5. RankGP [13], a Genetic Programming learning method, was proposed for learning ranking functions. In [14], proposed a generalization bound based on the p-norm objective and boosting-style algorithm for the problem of ranking was given.

In the existing approaches of learning to rank, the sparse prediction model only can be derived by Ranking SVM [3]. Sparse prediction model means the efficient prediction and the less risk to over-fit the training data. Sparse prediction model is desired especially in the real-time system. To the support vector learning approach, however, the number of support vectors grows steeply with the size of the training data.

In this paper, a relevance vector ranking algorithm is proposed to address the problem of sparse ranking model. This work was inspired by the relevance vector machines (RVM) [15] which have received much attention in the machine learning literature as a means of achieving sparse representations in the context of regression and classification. The Bayesian approach is employed to learn prediction model. As a learning result, the posterior distributions of many of the weights in the prediction model will sharply peaked around zero. The sample instances corresponding to the non-zero weights are named as 'Relevance Vector'. This property means efficient prediction and less risk to over-fit training data.

This paper is organized as follows. In Section 2, the problem statement is described. In Section 3, the proposed learning technique is given. The experimental results are shown in Section 4. In Section 5, the conclusion of this paper and future works are given.

2. Problem Statement

One central problem of information retrieval is to determine which documents are relevant and which are not to the user's information need. Given the document set $D=\{d_1,d_2,...,d_N\}$ and a query q, the relevance of a document with the query can be measured by cosine similarity, BM25, etc. According to the score of the function, the documents in D can be rated. In practice, better result can be obtained by combining the relevance measures to rate the instances. The ideal ranking function of combining the relevance measures can be learned with the approach of learning to rank.

In learning to rank problem, the original training data set is i.i.d data set $S=\{(\mathbf{x}_i,\mathrm{y}_i)\}_{i=1}^{N}$, each instance $\mathbf{x}_i$ is associated with a label y_i. The instance $\mathbf{x}_i$ is an n-dimensions observation vector, i.e. $\mathbf{x}_i \in \mathbb{R}^n$. R is a label space, $R=\{R_1,R_2,\ldots,R_k\}$. The object in R can be ranked as $R_k \succ_R \cdots \succ_R R_2 \succ_R R_1$. ($\succ_R$ represents the order among ranks). R_i is the label of rank i. In information retrieval, elements in observation vector can be the relevance measures between document and user's query. The label is the rank scale of relevance between instance and user's query.

The strategy of learning to rank from pairwise data has been employed to design the algorithms. Following this strategy, a new pairwise data set $\boldsymbol{\pi}$ should be generated by combining the instances in the original training data set S. The element π_i is a couple that can be written as (d_i,r_i). The element d_i in π_i denotes an instances pair, which can be written as (x,x'). The element r_i in π_i indicate the preference relation between two instances in the data pair d_i. The value of r_i is set with the function $\Upsilon(\cdot,\cdot)$ which takes the form as

$$\Upsilon(y,y')=\begin{cases}1, & y \succ y' \\ 0, & y \prec y' \text{ or } y=y'\end{cases} \tag{1}$$

where y and y' are the labels corresponding to the instance x and x'.

The goal of learning to rank is transformed to learning a ranking model from pairwise data set $\boldsymbol{\pi}$.

3. Proposed Method for Learning to Rank

3.1. Kernel-Based Ranking Function Model

Assuming the model space of mapping object to real number is $\mathscr{H}=\{f:X\mapsto\mathbb{R}\}$. Each model f in $\mathscr{H}$ creates an order $\succ_X$ in input space $X\subset\mathbb{R}^n$, according the following rule

$$\mathbf{x}_i \succ_X \mathbf{x}_j \Leftrightarrow f(\mathbf{x}_i) > f(\mathbf{x}_j) \tag{2}$$

which means that there is an unobservable latent function value $f(\mathbf{x}) \in \mathbb{R}$ associated with each instance $\mathbf{x}$, and that the preference relation between any two instances dependents on the latent function values of them.

Assuming the ranking model f in $\mathscr{H}$ is a linear model, which takes the SVM-like form

$$f(\mathbf{x}) = \sum_{i=1}^{M} w_i \cdot \left(\kappa\left(\mathrm{d}_i^{(1)}, \mathbf{x} \right) - \kappa\left(\mathrm{d}_i^{(2)}, \mathbf{x} \right) \right) \tag{3}$$

where M is the number of pairwise instances in data set $\boldsymbol{\pi}$, $\kappa(\cdot,\cdot)$ is the kernel function, $\mathbf{w}$ is a weight vector, $d_i^{(1)}$ is the left element in pair d_i, $d_i^{(2)}$ is the right element in pair d_i.

3.2. Prior

We encode a preference for smoother function with a zero-mean Gaussian prior distribution over $\mathbf{w}$. The automatic relevance determination (ARD) can be made by this prior. The prior distribution is given as the form

$$\begin{aligned} p(\mathbf{w} \mid \boldsymbol{\alpha}) &= \prod_{i=0}^{M} \mathrm{N}\left(w_i \left| 0, \alpha_i^{-1} \right. \right) \\ &= (2\pi)^{-\frac{M}{2}} \left| \mathrm{A} \right|^{\frac{1}{2}} \exp\left(-\frac{1}{2} \mathbf{w}^{\mathrm{T}} \mathrm{A} \mathbf{w} \right) \end{aligned} \tag{4}$$

where $\alpha_i^{-1/2}$ is the variance of the distribution of w_i. α_i is called hyperparameter. $\boldsymbol{\alpha}$ is a vector of M hyperparameters. There is an individual hyperparameter associated independently with every weight. We collect all hyperparameters into a diagonal matrix A=diag$(\alpha_1, \alpha_2, \ldots, \alpha_M)$.

3.3. Likelihood for preference variables

Given $\mathbf{x}$ and $\mathbf{x}'$ are observation vectors of two instances in original input space. First, the observation vectors are mapped to the real numbers by latent function f. Then the preference relation between $\mathbf{x}$ and $\mathbf{x}'$ can be derived by comparing the value of $f(\mathbf{x})$ and $f(\mathbf{x}')$. The rule of discriminating the preference relation of a pair takes the form

$$r = \begin{cases} 1, & f(\mathbf{x}) - f(\mathbf{x}') > 0 \\ 0, & f(\mathbf{x}) - f(\mathbf{x}') \le 0 \end{cases} \tag{5}$$

Then the probabilistic function $p(r_i \mid \mathbf{d}_i, \mathbf{w})$ denotes the conditional distribution that a preference label variable r is drew from, given two input vectors $\mathbf{x}$ and $\mathbf{x}'$. Adopting the Bernoulli distribution for $p(r_i \mid \mathbf{d}_i, \mathbf{w})$, the likelihood of target vector $\mathbf{r}$ on dataset $\boldsymbol{\pi}$ can be written as the form

$$\begin{aligned} p(\mathbf{r} \mid \mathrm{D}, \mathbf{w}) &= \prod_{i=1}^{M} p(r_i \mid \mathbf{d}_i, \mathbf{w}) \\ &= \prod_{i=1}^{M} Sig\left(f\left(\mathbf{d}_i^{(1)} \right) - f\left(\mathbf{d}_i^{(2)} \right) \right)^{r_i} \cdot \left[1 - Sig\left(f\left(\mathbf{d}_i^{(1)} \right) - f\left(\mathbf{d}_i^{(2)} \right) \right) \right]^{1-r_i} \end{aligned} \tag{6}$$

where D is the instances pair set in $\boldsymbol{\pi}$, $\mathbf{r}$ is the label set corresponding to the set D, $Sig\left(f\left(\mathbf{d}_i^{(1)}\right)-f\left(\mathbf{d}_i^{(2)}\right)\right)$ denotes the conditional probabilistic function as the form

$$Sig\left(f\left(\mathbf{d}_i^{(1)}\right)-f\left(\mathbf{d}_i^{(2)}\right)\right)=\frac{1}{1+\exp\left(f\left(\mathbf{d}_i^{(2)}\right)-f\left(\mathbf{d}_i^{(1)}\right)\right)} \tag{7}$$

which is the sigmoid function that is popular used to relate a real number to probability.

3.4. Posterior probability

Given hyperparameter vector $\boldsymbol{\alpha}$, the posterior parameter distribution conditioned on the data is given by combining the likelihood and prior within Bayes' rule. The posterior probability can then be written as the form

$$p(\mathbf{w}\mid\mathbf{r},\boldsymbol{\alpha})=\frac{p(\mathbf{r}\mid \mathrm{D},\mathbf{w})\,p(\mathbf{w}\mid\boldsymbol{\alpha})}{p(\mathbf{r})} \tag{8}$$

where the prior $p(\mathbf{w}\mid\boldsymbol{\alpha})$ is defined as in (4), the likelihood function $p(\mathbf{r}\mid \mathrm{D},\mathbf{w})$ is defined as in (6).

3.5. Learning Algorithm

First, fix the value of prior parameter vector $\boldsymbol{\alpha}$, the weight of latent function model is found by maximize the weight posterior $p\left(\mathbf{w}\mid\mathbf{r},\boldsymbol{\tau}\right)$. Then we applied the Laplace approximation [16] in evidence evaluation. The prior parameter vector $\boldsymbol{\alpha}$ is computed by maximize "evidence for the hyperparameters". The complete learning algorithm process by iterated re-estimated the weight and the prior parameter until some suitable convergence criteria have been satisfied.

For Latent Function Model Parameters Estimate, The MAP estimate on the latent functions is referred to find a $\mathbf{w}_{\mathbf{MAP}}\in\mathscr{H}$ which maximize the weight $\mathbf{w}$ posteriori probability of given prior parameter vector $\boldsymbol{\alpha}$. The Newton's method [17] is adapted to find $\mathbf{w}_{\mathbf{MAP}}$.

For prior parameter vector $\boldsymbol{\alpha}$, a practical efficient update equation is adopted, which is proposed in [16]. This method follows the strategy of maximizing "evidence for the hyperparameters". Derivative of optimization objective equating to zero and rearranging, gives the update form

$$a_i^{\text{new}}=\frac{1-a_i\Sigma_{ii}}{w_i^2} \tag{9}$$

where w_i is the i-th posterior mean weight, Σ_{ii} is the i-th diagonal element of matrix Σ . The matrix Σ is defined as $\Sigma=\left(\frac{\partial^2\mathrm{E}(\mathbf{w})}{\partial^2\mathbf{w}}\right)^{-1}$, where $\mathrm{E}(\mathbf{w})=-\ln p\left(\mathbf{r}\mid \mathrm{D},\mathbf{w}\right)p\left(\mathbf{w}\mid\boldsymbol{\alpha}\right)$.

The learning algorithm proceeds by repeated application of latent function parameters estimate and prior parameters optimization, concurrent updating of model weights $\mathbf{w}_{\mathbf{MAP}}$ and $\boldsymbol{\alpha}^*$, until the suitable convergence criteria has been satisfied. We name this learning algorithm as RVRank, for which details are summarized in Algorithm 1.

4. Experiment and Results

4.1. Experiment Setting

The relevance vector learning to rank algorithm is implemented in Matlab 2006. Because of the sigmoid function is used in likelihood, this algorithm is named as RVRankS.

In experiments, the performance of RVRankS is compared with two proposed state-of-the-art ranking algorithms:

One is supported vector learning to rank, is called RankSVM [4]. A binary exactable tool SVM^{light} [1] is used in the experiment.

Another one is preference learning with Gaussian processes, is named as GPPL [18]. The C language source code of GPPL [2] is downloaded for the experiment. This approach owns the advantages

Algorithm 1. Relevance Vector Learning to Rank Algorithm

Input: Training data set S;
Initial prior parameter vector $\boldsymbol{\alpha}$;
Kernel function parameter δ;
Convergency criteria ε.
Output: Model weight vector $\mathbf{w}^*$;
Prior parameter vector $\boldsymbol{\alpha}^*$;
Begin
Generate Kernel matrix M on data set S;
Repeat
$\mathrm{w}_{\mathrm{MAP}} \leftarrow \arg\min_{\mathrm{w}} \mathrm{E}(\mathrm{w})$;
$a_i^* \leftarrow \frac{1-a_i\Sigma_{ii}}{w_i^2},\ i=1,\ldots,\mathrm{length}(\boldsymbol{\alpha})$;
$\alpha' \leftarrow$ Maximal element in $(\boldsymbol{\alpha}^* - \boldsymbol{\alpha})$;
Until α' less than convergency criteria ε
End

of giving the probabilistic prediction of the results and learning all of the parameters in learning process. But the learned prediction model by it is not sparse.

In the experiments, Gaussian kernel function is used in all three algorithms. The Gaussian kernel function takes the form as

$$\kappa(\mathbf{x},\mathbf{x}') = \exp\left(-\left\|\mathbf{x}-\mathbf{x}'\right\|^2 \Big/ 2\delta^2\right)$$

where δ is the parameter that should be set.

The parameter δ in Gaussian kernel function is automatically searched by GPPL.

For RankSVM, the 5-fold cross validation on training set was used to determine the optimal values of error/margin trade-off parameter and the parameter in kernel function. The search is done on grid linearly space by 0.2 in the region of $\left\{(\log_{10} C, \log_{10}\delta)\,\middle|\, -3 \le \log_{10} C \le 3, -3 \le \log_{10}\delta \le 3\right\}$.

For RVRankS, the 5-fold cross validation on training set was used to determine the optimal values of the parameter δ in Gaussian kernel function. The search is done on grid linearly space by 0.2 in the region of $\left\{-3 \le \log_{10}\delta \le 3\right\}$.

The normalized discounted cumulative gain (NDCG) [19] is adopted as a performance measure, which takes the form

[1] http://svmlight.joachims.org/
[2] http://www.gatsby.ucl.ac.uk/~chuwei/plgp.htm

$$NDCG@k = N(k) \bullet \sum_{j=1}^{k} \frac{2^{r(j)} - 1}{\log(1+j)}$$

where N(k) is the NDCG at k of ideal ranking list. It is used as a normalization factor of the NDCG at k in the ranking list of prediction results.

4.2. Experiments on Document Retrieval Data

The OHSUMED collection [20] is used in document retrieval research. The relevance judgments of documents in OHSUMED are either 'd' (definitely relevant), 'p' (possibly relevant), or 'n' (not relevant). Rank 'n' has the largest number of documents, followed by 'p' and 'd'.

The OHSUMED has been collected into a Benchmark dataset LETOR [21] for ranking algorithm research. In this data set, each instance is represented as a vector of features, determined by a query and a document. Every vector consists of 25 features. The value of features has been computed.

We built five groups data sets with 2, 4, 6, 8, 10 queries as training set respectively. To the each training set scale, data set is randomly partitioned into training/test splits as the training queries quantity. The partition was repeated ten times independently. The ten training/test date sets were generated for each training set scale.

Three algorithms were run on these data sets. The evaluation results of NDCG@5 and NDCG@10 are given in Figure 1 and Figure 2. It is clear that the precision of first five instances and ten instances predicted by RVRankS is better than other two algorithms.

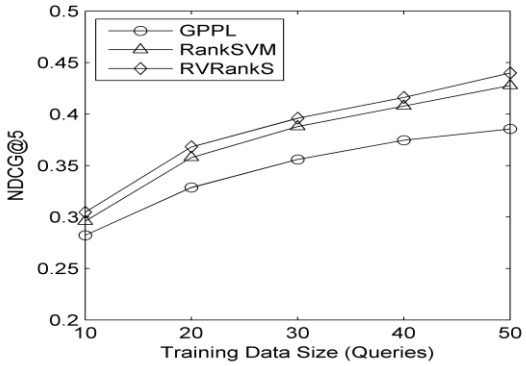

Figure 1. Results of scaling experiment of three algorithms on OHSUMED data set. NDCG@5 vs. the size of training data set.

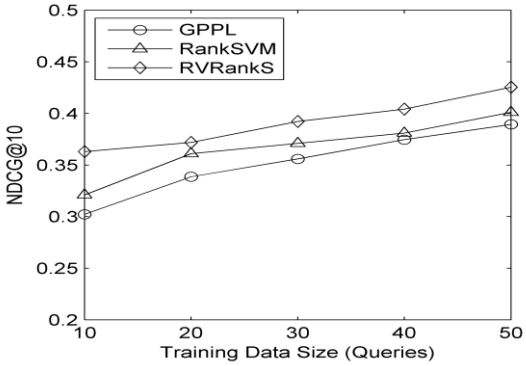

Figure 2. Results of scaling experiment of three algorithms on OHSUMED data set. NDCG@10 vs. the size of training data set.

In Figure 3, it is clear that the number of the training instances pairs utilized in prediction model learned by RVRankS increased slowly with enlarging the size of training data. While the number of the training instances pairs utilized in prediction model learned by RankSVM grows steeply.

5. Conclusion and Future Works

In this paper, the approach of relevance vector learning to rank is given. Experimental results on document retrieval data set show that the generalization of this approach is competitive with two state-

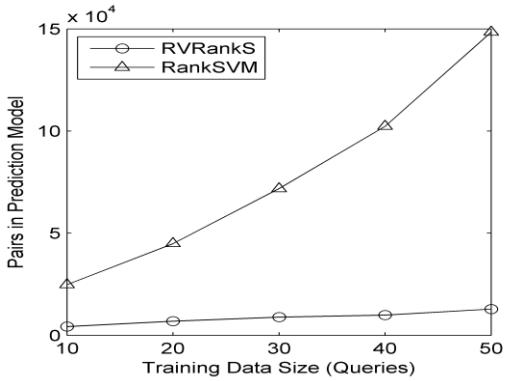

Figure 3. The number of pairs utilized in the prediction model by algorithms running on OHSUMED data set vs. the size of the training data set.

of-the-art algorithms, and that the number of vectors utilized in the prediction model learned by this approach is dramatically less than that of learned by support vector approach.

For future works, it will be interesting to conduct the experiment on more real world data sets, to use new approximation inference techniques to improve the performance of prediction and to develop the faster algorithms to tackle relatively large data sets.

6. References

[1] Kashefi O, Mohseni N, Minaei B, "Optimizing document similarity detection in Persian information retrieval", Journal of Convergence Information Technology, vol.5, no. 2, pp.101-106, 2010.

[2] Jin P, Li X, Chen H, Yue L, "CT-Rank: A Time-aware Ranking Algorithm for Web Search", Journal of Convergence Information Technology, vol.5, no. 6, 2010.

[3] Herbrich R, Graepel T, Obermayer K. "Support vector learning for ordinal regression". In Proceeding of the 9th International Conference on Artificial Neural Networks (ICANN), pp. 97-102, 1999.

[4] Joachims T. "Optimizing Search Engines using Clickthrough Data". In Proceeding of the ACM Conference on Knowledge Discovery and Data Mining, pp. 133-142, 2002.

[5] Pahikkala T, Tsivtsivadze E, Airola A, Boberg J, Salakoski T. "Learning to rank with pairwise regularized least-squares". In Proceeding of the 30th International Conference on Research and Development in Information Retrieval -Workshop on Learning to Rank for Information Retrieval, pp. 27-33, 2007.

[6] Freund Y, Iyer R, Schapire RE, Singer Y. "An efficient boosting algorithm for combining preferences". In Proceeding of the 15th International Conference on Machine Learning, pp. 1532-4435, 1998.

[7] Burges C, Shaked T, Renshaw E, Lazier A, Deeds M, Hamilton N, Hullender G. "Learning to rank using gradient descent". In Proceeding of the 22nd International Conference on Machine Learning, pp. 89-96, 2005.

[8] Tsai M-F, Liu T-Y, Qin T, Chen H-H, Ma W-Y. "FRank: A Ranking Method with Fidelity Loss". In Proceeding of the 30th International Conference on Research and Development in Information Retrieval, pp. 383-390, 2007.
[9] Chu W, Ghahramani Z. "Preference learning with Gaussian processes". In Proceeding pp. 137, 2005.
[10] Su J, Zhang H. "Learning conditional independence tree for ranking". In Proceeding of the 4th Ieee International Conference on Data Mining, pp. 531-534, 2004.
[11] Xia F, Zhang WS, Li FX, Yang YW, "Ranking with decision tree", Knowledge and Information Systems, vol.17, no. 3, pp.381-395, 2008.
[12] Zhang H, Su J, "Naive Bayes for optimal ranking", J EXP THEOR ARTIF IN, vol.20, no. 2, pp.79-93, 2008.
[13] Fan W, Gordon MD, Pathak P, "A generic ranking function discovery framework by genetic programming for information retrieval", INFORM PROCESS MANAG, vol.40, no. 4, pp.587-602, 2004.
[14] Rudin C, "The P-Norm Push: A Simple Convex Ranking Algorithm that Concentrates at the Top of the List", J MACH LEARN RES, vol.10, no. pp.2233-2271, 2009.
[15] Tipping ME, "Sparse Bayesian Learning and the Relevance Vector Machine", J MACH LEARN RES, vol.1, no. 3, pp.211-244, 2001.
[16] MacKay DJC, "A Practical Bayesian Framework for Backpropagation Networks", NEURAL COMPUT, vol.4, no. 3, pp.448-472, 1992.
[17] Nocedal J, Wright SJ. Numerical Optimization. Springer-Verlag New York, New York, 1999.
[18] Chu W, Ghahramani Z. "Preference learning with Gaussian processes". In Proceeding of the 22nd International Conference on Machine Learning, pp. 137-144, 2005.
[19] Kekalainen J, "Binary and graded relevance in IR evaluations - Comparison of the effects on ranking of IR systems", INFORM PROCESS MANAG, vol.41, no. 5, pp.1019-1033, 2005.
[20] Hersh W, Buckley C, Leone TJ, Hickam D. "OHSUMED: an interactive retrieval evaluation and new large test collection for research". In Proceeding of the 17th International Conference on Research and Development in Information Retrieval, pp. 192-201|358, 1994.
[21] Liu TY, Xu J, Qin T, Xiong W, Li H. "Letor: Benchmark dataset for research on learning to rank for information retrieval". In Proceeding of the 30th International Conference on Research and Development in Information Retrieval -Workshop on Learning to Rank for Information Retrieval, 2007.