Learning to Rank: Large Margin Method with Structured Output

Chuannan Huang Jiacheng Liang Shasha Feng

Abstract

We use large margin method with structured output to learn a ranking algorithm from training data. The measurement based on NCDG (Normalized Discounted Cumulative Gain) shows the feasibility of such approach.

1 Introduction

Information retrieval is the process to locate relevant documents on the basis of user inputs such as keywords or example documents, e.g., search engine Google returns a list of webpage given the query. During information retrieval, one important procedure is to rank the relevance of documents (Fig 1). In earlier days, there were engineering methods such as BM25 to reveal the relevance of query-document pair. Recently the trend shifted to learning methods [6]. The Yahoo! Learning to Rank Challenge in 2010 is a community-wide competition to promote the datasets and encourage developing new learning to rank algorithms [1].

In this project we rank the document using a support vector machine (SVM)-style method combined with structure learning, based on Chapelle *et al*'s work [2].

1.1 What we try to learn from training data

For training data, we have a number of features of each query-document pair X_{qi} as input and a relevance score g_{qi} for each query-document pair as ground truth (Fig 2). Given the relevance score,

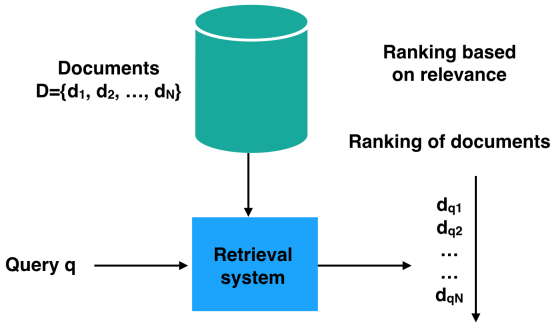

Figure 1: An illustration of information retrieval process and the core of it, ranking problem.

we rank the documents to give a list Y. We list some features here for example [7]. Feature 1 describes the total frequency of query terms appearing in document title. Feature 26 describes BM25 of title and feature 27 describes logarithm of it. Feature 33 describe the ratio between number of documents whose title and abstract contain the query and the total number of documents for a query.

$$\text{Feature 1: } \sum_{q_i \in q \cap d} c(q_i, d) \text{ in title} \tag{1a}$$

$$\text{Feature 25: } BM25(d,q) = \sum_{q_i \in q} \frac{idf(q_i) \cdot c(q_i,d) \cdot (k_1+1)}{c(q_i,d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})} \cdot \frac{(k_3+1)c(q_i,q)}{k_3 + c(q_i,q)} \tag{1b}$$

$$\text{Feature 33: } \sum_{q_i \in q \cap d} \frac{c(q_i,d)}{|d|} \text{ in 'title and abstract'} \tag{1c}$$

In BM25, the *idf* is inverse document frequency, $|d|$ is the document length in words, $c(q_i, d)$ is the query term's frequency in document i, and $avgdl$ is the average document length in the text collection from which documents are drawn. The parameters (k_1, k_3, d) are set as (1.2, 7, 0.75) in OHSUMED dateset.

Essentially we try to learn the relation between those features and the relevance label, so that during test when we are given a set of input features, we will be able to predict the relevance and come up with a ranking result.

2 Background of Methods

2.1 Structured learning

Structured learning deals with complex outputs, e.g., trees, sequences, graphs etc. In ranking problem, the listwise ranking gives a sequence of documents which is a structured output. With the input features X describing the query-document pairs, we try to map the relation from X to ranking result Y, which contains a high dimension hypothesis space and lacks tools to describe such mapping.

In structured learning, usually instead of directly learning of a mapping, we can make both X and Y into the joint feature map (X,Y) and map it to a real domain $\mathbb{R}$ with a function $\mathcal{F}$. The $\mathcal{F}$ is also called the compatibility function, with more compatible X-Y pairs having higher values of $\mathcal{F}$ function. The negative of $\mathcal{F}$ can thus be regarded as a loss function. Structured learning has wide applications in classification with class taxonomies, natural language parsing, sequence alignment, and ranking [8].

2.2 SVM-Style Large Margin Method

Support vector machine is a large margin approach, where we maximize the margin, the closest distance from data points to the separation plane. In this project, 'margin' is defined in a slightly different sense, though the formulations of quadratic programming problem and constraints are very

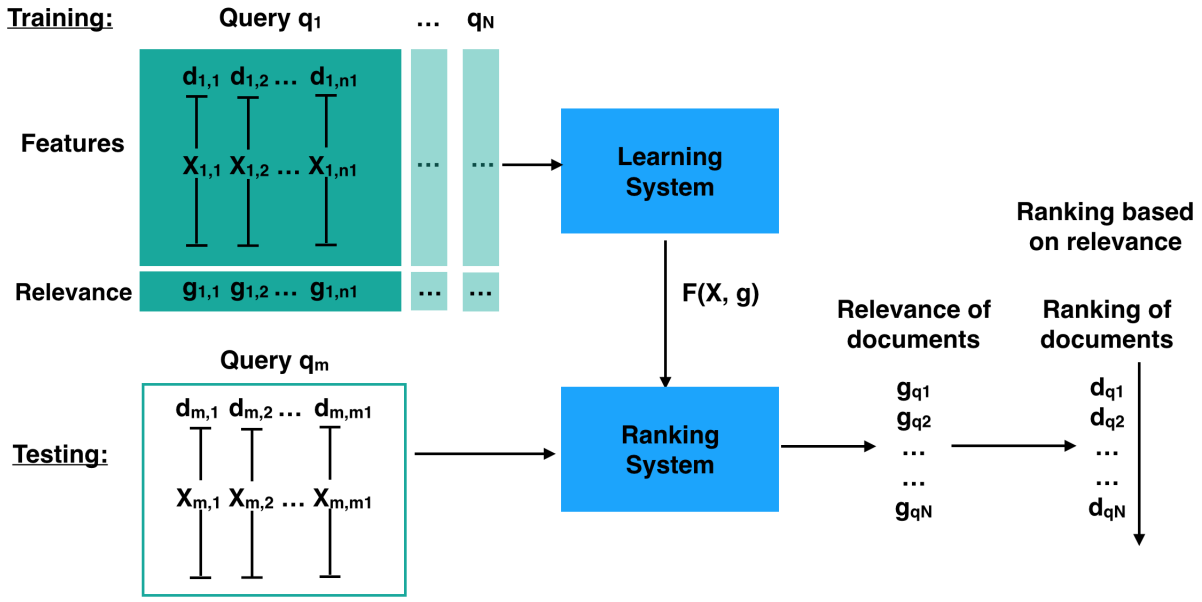

Figure 2: A general learning-to-rank model with training and testing data. The inputs of training data are features describing query-document pair and relevance score g given by human annotators. The learning system tries to learn the relation between feature and relevance score, so for testing data it can predict the relevance score for ranking.

similar to SVM. Here margein is defined by the 'distance' between the best ranking sequence and the closest runner-up. Such distance is measured by Normalized Discounted Cumulative Gain (NDCG):

$$NDCG := \frac{1}{N_q} \sum_{i=1}^{k} D(y_i)\phi(g_{qi}) \tag{2a}$$

$$D(y_i) = \frac{1}{log_2(1+y_i)} \tag{2b}$$

$$\phi(g_{qi}) = 2^{g_{qi}} - 1 \tag{2c}$$

Here D is the discount function, ϕ is an increasing function, and Nq is a normalization constant so that optimal ranking has NDCG score 1. The k is a truncation level so that only the first k documents are important, considering the senarios such as in social network apps only the top few friend recommendation matters most. There are also other ranking measures such as Mean Reciprocal Rank (MRR), Precision@n, and Expected Rank Utility (ERU) [3, 4, 5]. We limit our scope to NDCG in this project, as this measure reduces the computer load and in many cases only the top few documents matter most.

2.3 Datasets

The OHSUMED dataset is a subset of MDELINE, a database on medical literatures. Each record contains following fields: title, abstract, MeSH indexing terms, author, source, and publication type. As part of LETOR package initially published in 2007, the authors from Microsoft engineered the features to provide a standard benchmark dataset for learn-to-rank method [7]. The total 45 features of OHSUMED corpus are listed in Appendix I. Three versions exist in the downloaded dataset, the "null" version uses 'NULL' for missing values, the "min" version uses the minimium value of individual feature of certain query for missing values, and the "norm" version normalizes each feature based on "min" version so the values of each feature is a number between 0 and 1, as some of the features have very large values. In this project we use the "norm" version.

3 Formulation

3.1 Joint Feature of Structured Learning

The joint feature map (X, Y) is described by $\Psi(X_q, y)$. For a given query q with m documents associate with it, the relation of i-th document and query is described by feature vector $\boldsymbol{X}_{qi}$. The i-th document is ranked y_{qi}-th position. The function $A(y_{qi})$ in $\mathcal{F}$ in (4) has the property of $A(y_{qi}) = 0$ for $r > k$, because in NDCG measure we pay attention to only the first k elements of ranking.

$$\Psi(\boldsymbol{X}_q, y_q) = \sum_i \boldsymbol{X}_{qi} \cdot A(y_{qi}) = \sum_i \boldsymbol{X}_{qi} \cdot \max(k + 1 - y_{qi}, 0) \tag{3}$$

The compatible function $\mathcal{F}$ that maps (X, Y) to $\mathbb{R}$ is defined as follows:

$$\mathcal{F}(X_q, Y) = \boldsymbol{w}^T \Psi(\boldsymbol{X}_q, y_q) = \sum_i \boldsymbol{w}^T \boldsymbol{X}_{qi} A(y_{qi}) \tag{4}$$

3.2 Formulation of SVM-style problem

3.2.1 Goal of Learning

In training stage, we aim to find out a set of $\boldsymbol{w}$ that yields a large margin between optimal ranking y_q and the second runner-up y_i. The $\boldsymbol{w}$ here catches the relation between input features and the relevance score. Actually $\boldsymbol{w}^T \boldsymbol{X}_{qi}$ can be viewed as the relevance score for i-th document regarding query q. The $\boldsymbol{w}$ needs to satisfy following condition:

$$\forall q, \forall y \neq y_q, \ \boldsymbol{w}^T \Psi(\boldsymbol{X}_q, y_q) - \boldsymbol{w}^T \Psi(\boldsymbol{X}_q, y) > 0 \tag{5}$$

In testing stage, we try to find the optimal ranking $\widetilde{y}$ by maximizing compatible function $\mathcal{F}$.

$$\widetilde{y} = \arg \max_y \ \mathcal{F}(X_q, Y) = \arg \max_y \sum_i \boldsymbol{w}^T \boldsymbol{X}_{qi} A(y_{qi}) \tag{6}$$

Algorithm 1 Optimization by cutting plan method

$w \leftarrow 0, \xi \leftarrow 0, \S_q \leftarrow \emptyset, i = 1, ..., n_q.$
repeat
  for $q = 1...n_q$ **do**
    $\widetilde{y} \leftarrow \arg\max of(8)$
    if (6) is violated with $y = \widetilde{y}$ **then**
      $S_q \leftarrow S_q \cup \widetilde{y}$
      Optimize () under subset of constraints () $\forall q, \forall y \in S_q$
    end if
  end for
until No S_q has changed.

3.2.2 Primal Form of SVM

Next, we formulate the large margin similar to canonical SVM algorithm as below. We first write the primal form of large margin method with slack variable,

$$\begin{aligned} \min_{w,\,\xi} \quad & \frac{\lambda}{2} w^T w + \sum_q \xi_q \\ \text{s.t. } \forall i, \forall y \in \mathcal{Y} \setminus y_q, \quad & \boldsymbol{w}^T \Psi(\boldsymbol{X}_q, y_q) - \boldsymbol{w}^T \Psi(\boldsymbol{X}_q, y) \geq 1 - \xi_q \end{aligned} \tag{7}$$

Upon slack rescaling and margin rescaling, the above formulation transforms into

$$\begin{aligned} \min_{w,\,\xi} \quad & \frac{\lambda}{2} w^T w + \sum_q \xi_q \\ \text{s.t. } \forall i, \forall y \in \mathcal{Y} \setminus y_q, \quad & \boldsymbol{w}^T \Psi(\boldsymbol{X}_q, y_q) - \boldsymbol{w}^T \Psi(\boldsymbol{X}_q, y) \geq \Delta_q(y) - \xi_q \end{aligned} \tag{8}$$

where $\Delta_q(y)$ is defined as

$$\Delta_q(y) = NDCG_k(y_q, q) - NDCG_k(y, q) = 1 - NDCG_k(y, q) \tag{9}$$

Here we focus on ξ_q value for the time being, the above inequality can also be rewritten as

$$\xi_q \geq \Delta_q(\hat{y}_q) + w^T \Psi(X_q, \hat{y}_q) - w^T \Psi(X_q, y_q) \tag{10}$$

Since we want to minimize the sum of ξ_q yet it needs to be greater than certain value in (10), the optimal ξ_q is being exactly the maxinum value of the right hand side of (x):

$$\xi_q = \max_{y \neq y_q} \Delta_q(y) - w^T \Psi(x_q, y_q) + w^T \Psi(x_q, y). \tag{11}$$

The optimal y is thus as follows after discarding the terms independent of y:

$$\arg\ \max_y \sum_{i=1}^{k} A(y_i) w^T x_{qi} - \sum_{i=1}^{k} \phi(g_{qi}) D(y_i). \tag{12}$$

3.2.3 Dual Form of SVM

To optimize the w, ξ, we solve the quadratic programming problem by extending it into dual form. The Gram matrix K has $K_{ij} = v_i^T v_j$, with $v_i = \Psi(x_q, \widetilde{y}_i) - \Psi(x_q, y_q)$.

$$\max_{\alpha} \; b^T \alpha - \frac{1}{2} \alpha^T K \alpha \tag{13}$$

under constraints

$$\alpha \geq 0 \;\; and \;\; \forall q, \sum_{i \in U_q} \alpha_i \leq C.$$

When we solve this problem, we use the technique similar to expectation maximization (EM) method. In E-step, we estimate the best ranking $\widetilde{y}$, and in M-step, we maximize (13) and get better estimation of w, ξ through α.

Because in (8) there are exponential number of restraints, which would be heavy computation load. In optimization process, we construct a nested sequence of tighter relaxations of original problem using a cutting plane method. For the y we find with the optimizing condition, if the restraint is violated, we add it to the subsets of constraints. So only limited number of constraints are applied and calculated.

4 Implementation

Based on Algorithm 1, we use Kuhn-Munkres algorithm (also called Hungarian method) to find the optimal ranking $\widetilde{y}$ and quadratic programming for optimization problem. In method "kuhn_m(k, df_data)" is imported from scipy.optimize::linear_sum_assignment. The Hungarian method is a combinatorial optimization algorithm that solves the assignment problem in polynomial time and which anticipated later primal-dual methods. As it is designed to solve the assignment problem which is also a classic problem of linear programming. We tried to find some methods that solve optimazation problem and then found a function from spicy library.

Here is the description of func: scipy.optimize.linear_sum_assignment(cost_matrix). The linear sum assignment problem is also known as minimum weight matching in bipartite graphs. A problem instance is described by a matrix C, where each C[i,j] is the cost of matching vertex i of the first partite set (a worker) and vertex j of the second set (a job). The goal is to find a complete assignment of workers to jobs of minimal cost. The method used is the Hungarian algorithm, also known as the Munkres or Kuhn-Munkres algorithm.

The bottleneck of Algorithm 1 is solving the Quadratic Problem(5). We import qpsolvers::solve_qp. This method needs 6 parameters(P, q, G, h, A, b). The standard form for quadratic program it solves is like:

$$\begin{aligned} &\min_{\alpha} \frac{1}{2} x^T P x + q^T x \\ &s.t. : Gx \leq h, \;\; Ax = b \end{aligned} \tag{14}$$

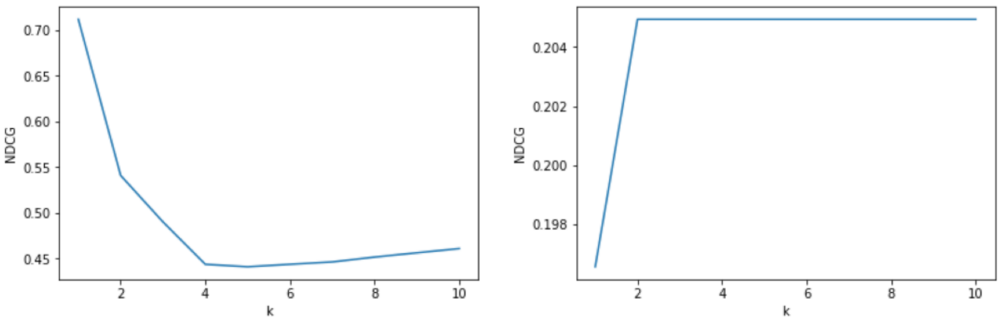

Figure 3: Results of two methods, left: structured learning with large margin method by us, right: RankSVM.

For each k, we get one optimized w for all queries. Then we use this w in test dataset and get NDCG values respect to different k values. One pair of dataset(including train_data and test_data) has ten k values(from 1 to 10) and ten NDCG values, and the final ten NDCG values are the average of all these NDCG value list. In the code, we print the NDCG values list for one dataset.

Third party package used here is "qp_solver", which can be installed by "pip install qpsolvers". Other packages are typical python packages including Numpy, Pandas, Scipy, Matplotlib.

5 Performance

The trend of our result is similar to the paper's. For comparison, we use RankSVM package written in C language downloaded from `http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`. We experimented with different c value for regularization. However, because the default loss function in RankSVM is different from NDCG measure we used here and we were unable to find the handler to modify loss function, the result of RankSVM is different from that presented in the paper. As the authors have pointed out, this structured learning model appears to be unstable in terms of model selection (different k values), which is probably due to the small training set.

6 Conclusion

The large margin method combined with structure learning is able to make relatively good ranking performance. In this project, we mainly reproduce the result of Chapelle *et al*'s work and got better insights into structure learning method.

7 Contributions

Team member contributions are listed as follows:

Chuannan Huang researched NCDG measurement for first checkpoint, wrote the Kuhn algorithm for finding optimal rankingcontributed to "Implementation" sectoin of report, and participated a lot in the discussion of understanding the paper;

Jiacheng Liang wrote data processing, most of Algorithm 1, QP programing and plotted the results, worked out structured output formula details, and contributed to deeper understanding of paper during discussion;

Shasha Feng worked out the major mathematicals of structured learning and large margin, wrote data pre-processing code, wrote the "Introduction", "Background of methods", and "Formulation" sections of report and drafted the first checkpoint report.

References

[1] Chapelle, O., and Chang, Y. Yahoo! Learning to Rank Challenge Overview. In *: Workshop and Conference Proceedings 14* (2011).

[2] Chapelle, O., Le, Q., and Smola, A. Large Margin Optimization of Ranking Measures . In *NIPS* (2007).

[3] Craswell, N. *Mean Reciprocal Rank*. Springer US, Boston, MA, 2009, pp. 1703–1703.

[4] Craswell, N. *Precision at n*. Springer US, Boston, MA, 2009, pp. 2127–2128.

[5] DIECIDUE, E., and WAKKER, P. P. On the intuition of rank-dependent utility. *Journal of Risk and Uncertainty 23*, 3 (2001), 281–298.

[6] Li, H. A Short Introduction to Learning to Rank . In *IEICE TRANS. INF. SYST., VOL.E94D, NO.10* (2011).

[7] Qin, T., Liu, T.-Y., Xu, J., and Li, H. Letor: A benchmark collection for research on learning to rank for information retrieval. *Inf. Retr. 13*, 4 (Aug. 2010), 346–374.

[8] Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res. 6* (Dec. 2005), 1453–1484.