PAPER *Special Section on Information-Based Induction Sciences and Machine Learning*

A Short Introduction to Learning to Rank

Hang LI[†], ***Nonmember***

SUMMARY Learning to rank refers to machine learning techniques for training the model in a ranking task. Learning to rank is useful for many applications in Information Retrieval, Natural Language Processing, and Data Mining. Intensive studies have been conducted on the problem and significant progress has been made [1], [2]. This short paper gives an introduction to learning to rank, and it specifically explains the fundamental problems, existing approaches, and future work of learning to rank. Several learning to rank methods using SVM techniques are described in details.

key words: *Learning to rank, information retrieval, natural language processing, SVM*

1. Ranking Problem

Learning to rank can be employed in a wide variety of applications in Information Retrieval (IR), Natural Language Processing (NLP), and Data Mining (DM). Typical applications are document retrieval, expert search, definition search, collaborative filtering, question answering, keyphrase extraction, document summarization, and machine translation [2]. Without loss of generality, we take document retrieval as example in this article.

Document retrieval is a task as follows (Fig. 1). The system maintains a collection of documents. Given a query, the system retrieves documents containing the query words from the collection, ranks the documents, and returns the top ranked documents. The ranking task is performed by using a ranking model $f(q,d)$ to sort the documents, where q denotes a query and d denotes a document.

Traditionally, the ranking model $f(q,d)$ is created without training. In the BM25 model, for example, it is assumed that $f(q,d)$ is represented by a conditional probability distribution $P(r|q,d)$ where r takes on 1 or 0 as value and denotes being relevant or irreverent, and q and d denote a query and a document respectively. In Language Model for IR (LMIR), $f(q,d)$ is represented as a conditional probability distribution $P(q|d)$. The probability models can be calculated with the words appearing in the query and document, and thus no training is needed (only tuning of a small number of parameters is necessary) [3].

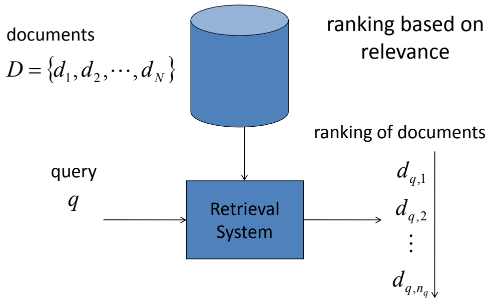

Fig. 1 Document Retrieval

A new trend has recently arisen in document retrieval, particularly in web search, that is, to employ machine learning techniques to automatically construct the ranking model $f(q,d)$. This is motivated by a number of facts. At web search, there are many signals which can represent relevance, for example, the anchor texts and PageRank score of a web page. Incorporating such information into the ranking model and automatically constructing the ranking model using machine learning techniques becomes a natural choice. In web search engines, a large amount of search log data, such as click through data, is accumulated. This makes it possible to derive training data from search log data and automatically create the ranking model. In fact, learning to rank has become one of the key technologies for modern web search.

We describe a number of issues in learning for ranking, including training and testing, data labeling, feature construction, evaluation, and relations with ordinal classification.

1.1 Training and Testing

Learning to rank is a supervised learning task and thus has training and testing phases (see Fig. 2).

The training data consists of queries and documents. Each query is associated with a number of documents. The relevance of the documents with respect to the query is also given. The relevance information can be represented in several ways. Here, we take the most widely used approach and assume that the relevance of a document with respect to a query is represented by

Manuscript received December 31, 2010.
Manuscript revised June 1, 2011.

[†]The author is with Microsoft Research Asia

DOI: 10.1587/transinf.E94.D.1

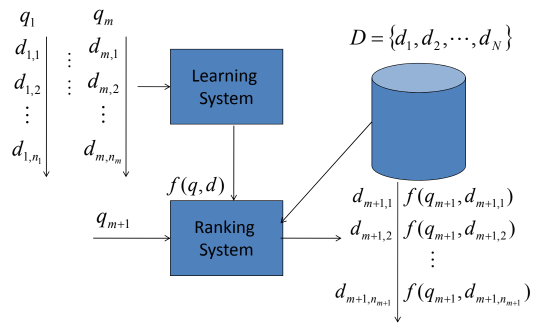
q_1
q_m
$d_{1,1}$
$d_{1,2}$
d_{1,n_1}
$d_{m,1}$
$d_{m,2}$
d_{m,n_m}
Learning System
$f(q,d)$
q_{m+1}
Ranking System
$D=\{d_1,d_2,\cdots,d_N\}$
$d_{m+1,1}$ $f(q_{m+1},d_{m+1,1})$
$d_{m+1,2}$ $f(q_{m+1},d_{m+1,2})$
$d_{m+1,n_{m+1}}$ $f(q_{m+1},d_{m+1,n_{m+1}})$

Fig. 2 Learning to Rank for Document Retrieval

a label, while the labels denote several grades (levels). The higher grade a document has, the more relevant the document is.

Suppose that $\mathcal{Q}$ is the query set and $\mathcal{D}$ is the document set. Suppose that $\mathcal{Y} = \{1, 2, \cdots, l\}$ is the label set, where labels represent grades. There exists a total order between the grades $l \succ l-1 \succ \cdots \succ 1$, where $\succ$ denotes the order relation. Further suppose that $\{q_1, q_2, \cdots, q_m\}$ is the set of queries for training and q_i is the i-th query. $D_i = \{d_{i,1}, d_{i,2}, \cdots, d_{i,n_i}\}$ is the set of documents associated with query q_i and $\mathbf{y}_i = \{y_{i,1}, y_{i,2}, \cdots, y_{i,n_i}\}$ is the set of labels associated with query q_i, where n_i denotes the sizes of D_i and $\mathbf{y}_i$; $d_{i,j}$ denotes the j-th document in D_i; and $y_{i,j} \in Y$ denotes the j-th grade label in $\mathbf{y}_i$, representing the relevance degree of $d_{i,j}$ with respect to q_i. The original training set is denoted as $S = \{(q_i, D_i), \mathbf{y}_i\}_{i=1}^m$.

A feature vector $x_{i,j} = \phi(q_i, d_{i,j})$ is created from each query-document pair $(q_i, d_{i,j}), i = 1, 2, \cdots, m; j = 1, 2, \cdots, n_i$, where ϕ denotes the feature functions. That is to say, features are defined as functions of a query document pair. For example, BM25 and PageRank are typical features [2]. Letting $\mathbf{x}_i = \{x_{i,1}, x_{i,2}, \cdots, x_{i,n_i}\}$, we represent the training data set as $S' = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$. Here $x \in \mathcal{X}$ and $\mathcal{X} \subseteq \Re^d$.

We aim to train a (local) ranking model $f(q,d) = f(x)$ that can assign a score to a given query document pair q and d, or equivalently to a given feature vector x. More generally, we can also consider training a global ranking model $F(q, D) = F(\mathbf{x})$. The local ranking model outputs a single score, while the global ranking model outputs a list of scores.

Let the documents in D_i be identified by the integers $\{1, 2, \cdots, n_i\}$. We define a permutation (ranking list) π_i on D_i as a bijection from $\{1, 2, \cdots, n_i\}$ to itself. We use Π_i to denote the set of all possible permutations on D_i, use $\pi_i(j)$ to denote the rank (or position) of the j-th document (i.e., $d_{i,j}$) in permutation π_i. Ranking is nothing but to select a permutation $\pi_i \in \Pi_i$ for the given query q_i and the associated documents D_i using the scores given by the ranking model $f(q_i, d_i)$.

The test data consists of a new query q_{m+1} and associated documents D_{m+1}. $T = \{(q_{m+1}, D_{m+1})\}$. We create feature vector $\mathbf{x}_{m+1}$, use the trained ranking model to assign scores to the documents D_{m+1}, sort them based on the scores, and give the ranking list of documents as output π_{m+1}.

The training and testing data is similar to, but different from the data in conventional supervised learning such as classification and regression. Query and its associated documents form a group. The groups are i.i.d. data, while the instances within a group are not i.i.d. data. A local ranking model is a function of a query and a document, or equivalently, a function of a feature vector derived from a query and a document.

1.2 Data Labeling

Currently there are two ways to create training data. The first one is by human judgments and the second one is by derivation from search log data. We explain the first approach here. Explanations on the second approach can be found in [2]. In the first approach, a set of queries is randomly selected from the query log of a search system. Suppose that there are multiple search systems. Then the queries are submitted to the search systems and all the top ranked documents are collected. As a result, each query is associated with multiple documents. Human judges are then asked to make relevance judgments on all the query document pairs. Relevance judgments are usually conducted at five levels, for example, perfect, excellent, good, fair, and bad. Human judges make relevance judgments from the viewpoint of average users. For example, if the query is 'Microsoft', and the web page is microsoft.com, then the label is 'perfect'. Furthermore, the Wikipedia page about Microsoft is 'excellent', and so on. Labels representing relevance are then assigned to the query document pairs. Relevance judgment on a query document pair can be performed by multiple judges and then majority voting can be conducted. Benchmark data sets on learning to rank have also been released [4].

1.3 Evaluation

The evaluation on the performance of a ranking model is carried out by comparison between the ranking lists output by the model and the ranking lists given as the ground truth. Several evaluation measures are widely used in IR and other fields. These include NDCG (Normalized Discounted Cumulative Gain), DCG (Discounted Cumulative Gain), MAP (Mean Average Precision), and Kendall's Tau.

Given query q_i and associated documents D_i, suppose that π_i is the ranking list (permutation) on D_i and $\mathbf{y}_i$ is the set of labels (grades) of D_i. DCG [5] measures the goodness of the ranking list with the labels. Specifically, DCG at position k is defined as:

$$DCG(k) = \sum_{j:\pi_i(j) \le k} G(j) D(\pi_i(j)),$$

where $G_i(\cdot)$ is a gain function and $D_i(\cdot)$ is a position discount function, and $\pi_i(j)$ is the position of $d_{i,j}$ in π_i. The summation is taken over the top k positions in the ranking list π_i. DCG represents the cumulative gain of accessing the information from position one to position k with discounts on the positions. NDCG is normalized DCG and NDCG at position k is defined as:

$$NDCG(k) = G^{-1}_{max,i}(k) \sum_{j:\pi_i(j) \leq k} G(j)D(\pi_i(j)),$$

where $G_{max,i}(k)$ is the normalizing factor and is chosen such that a perfect ranking π^*_i's $NDCG$ score at position k is 1. In a perfect ranking, the documents with higher grades are always ranked higher. Note that there can be multiple perfect rankings for a query and associated documents.

The gain function is normally defined as an exponential function of grade. That is to say, the satisfaction of accessing information exponentially increases when the grade of relevance of information increases.

$$G(j) = 2^{y_{i,j}} - 1, \tag{1}$$

where $y_{i,j}$ is the label (grade) of document $d_{i,j}$ in ranking list π_i. The discount function is normally defined as a logarithmic function of position. That is to say, the satisfaction of accessing information logarithmically decreases when the position of access increases.

$$D(\pi_i(j)) = \frac{1}{\log_2(1 + \pi_i(j))}, \tag{2}$$

where $\pi_i(j)$ is the position of document $d_{i,j}$ in ranking list π_i.

Hence, DCG and NDCG at position k become

$$DCG(k) = \sum_{j:\pi_i(j) \leq k} \frac{2^{y_{i,j}} - 1}{\log_2(1 + \pi_i(j))}, \tag{3}$$

$$NDCG(k) = G^{-1}_{max,i}(k) \sum_{j:\pi_i(j) \leq k} \frac{2^{y_{i,j}} - 1}{\log_2(1 + \pi_i(j))}. \tag{4}$$

In evaluation, DCG and NDCG values are further averaged over queries.

Table 1 gives examples of calculating NDCG values of two ranking lists. NDCG (DCG) has the effect of giving high scores to the ranking lists in which relevant documents are ranked high. For perfect rankings, the NDCG value at each position is always one, while for imperfect rankings, the NDCG values are usually less than one.

Table 1 Examples of NDCG Calculation.

Perfect ranking	Formula	Explanation
(3, 3, 2, 2, 1, 1, 1)		grades:3,2,1
(7, 7, 3, 3, 1, 1, 1)	Eq.(1)	gains
(1, 0.63, 0.5, $\cdots$)	Eq.(2)	discounts
(7, 11.41, 12.91, $\cdots$)	Eq.(3)	DCG
(1/7, 1/11.41, 1/12.91,$\cdots$)		normalizers
(1,1,1,$\cdots$)	Eq.(4)	NDCG
Imperfect ranking	**Formula**	**Explanation**
(2, 3, 2, 3, 1, 1, 1)		grades:3,2,1
(3, 7, 3, 7, 1, 1, 1)	Eq.(1)	gains
(1, 0.63, 0.5, $\cdots$)	Eq.(2)	discounts
(3, 7.41, 8.91, $\cdots$)	Eq.(3)	DCG
(1/7, 1/11.41, 1/12.91,$\cdots$)		normalizers
(0.43, 0.65, 0.69, $\cdots$)	Eq.(4)	NDCG

MAP is another measure widely used in IR. In MAP, it is assumed that the grades of relevance are at two levels: 1 and 0. Given query q_i, associated documents D_i, ranking list π_i on D_i, and labels $\mathbf{y}_i$ of D_i, Average Precision for q_i is defined as:

$$AP = \frac{\sum_{j=1}^{n_i} P(j) \cdot y_{i,j}}{\sum_{j=1}^{n_i} y_{i,j}},$$

where $y_{i,j}$ is the label (grade) of $d_{i,j}$ and takes on 1 or 0 as a value, representing being relevant or irrelevant. $P(j)$ for query q_i is defined as:

$$P(j) = \frac{\sum_{k:\pi_i(k) \leq \pi_i(j)} y_{i,k}}{\pi_i(j)},$$

where $\pi_i(j)$ is the position of $d_{i,j}$ in π_i. $P(j)$ represents the precision until the position of $d_{i,j}$ for q_i. Note that labels are either 1 or 0, and thus 'precision' can be defined. Average Precision represents averaged precision over all the positions of documents with label 1 for query q_i.

Average Precision values are further averaged over queries to become Mean Average Precision (MAP).

1.4 Relation with Ordinal Classification

Ordinal classification (also known as ordinal regression) is similar to ranking, but is also different. The input of ordinal classification is a feature vector x and the output is a label y representing a grade, where the grades are classes in a total order. The goal of learning is to construct a model which can assign a grade label y to a given feature vector x. The model mainly consists of a scoring function $f(x)$. The model first assigns a real number to x using $f(x)$ and then determines the grade y of x using a number of thresholds. Specifically, it partitions the real number axis into intervals and aligns each interval to a grade. It takes the grade of the interval that $f(x)$ falls into as the grade of x.

In ranking, one cares more about accurate ordering of objects, while in ordinal classification, one cares more about accurate ordered-categorization of objects. A typical example of ordinal classification is product rating. For example, given the features of a movie, we are to assign a number of stars (ratings) to the movie. In that case, correct assignment of the number of stars is critical. In contrast, in ranking such as document retrieval, given a query, the objective is to correctly sort related documents, although sometimes training data and testing data are labeled at multiple grades as in ordinal classification. The number of documents to be ranked can vary from query to query. There are queries for which more relevant documents are available in the collection, and there are also queries for which only weakly relevant documents are available.

2. Formulation

We formalize learning to rank as a supervised learning task. Suppose that $\mathcal{X}$ is the input space (feature space) consisting of lists of feature vectors, and $\mathcal{Y}$ is the output space consisting of lists of grades. Further suppose that $\mathbf{x}$ is an element of $\mathcal{X}$ representing a list of feature vectors and $\mathbf{y}$ is an element of $\mathcal{Y}$ representing a list of grades. Let $P(X,Y)$ be an unknown joint probability distribution where random variable X takes $\mathbf{x}$ as its value and random variable Y takes $\mathbf{y}$ as its value.

Assume that $F(\cdot)$ is a function mapping from a list of feature vectors $\mathbf{x}$ to a list of scores. The goal of the learning task is to automatically learn a function $\hat{F}(\mathbf{x})$ given training data $(\mathbf{x}_1,\mathbf{y}_1),(\mathbf{x}_2,\mathbf{y}_2),\ldots,(\mathbf{x}_m,\mathbf{y}_m)$. Each training instance is comprised of feature vectors $\mathbf{x}_i$ and the corresponding grades $\mathbf{y}_i$ $(i=1,\cdots,m)$. Here m denotes the number of training instances.

$F(\mathbf{x})$ and $\mathbf{y}$ can be further written as $F(\mathbf{x}) = (f(x_1), f(x_2),\cdots,f(x_n))$ and $\mathbf{y}=(y_1,y_2,\cdots,y_n)$. The feature vectors represent objects to be ranked. Here $f(x)$ denotes the local ranking function and n denotes the number of feature vectors and grades.

A loss function $L(\cdot,\cdot)$ is utilized to evaluate the prediction result of $F(\cdot)$. First, feature vectors $\mathbf{x}$ are ranked according to $F(\mathbf{x})$, then the top n results of the ranking are evaluated using their corresponding grades $\mathbf{y}$. If the feature vectors with higher grades are ranked higher, then the loss will be small. Otherwise, the loss will be large. The loss function is specifically represented as $L(F(\mathbf{x}),\mathbf{y})$. Note that the loss function for ranking is slightly different from the loss functions in other statistical learning tasks, in the sense that it makes use of sorting.

We further define the risk function $R(\cdot)$ as the expected loss function with respect to the joint distribution $P(X,Y)$,

$$R(F) = \int_{\mathcal{X}\times\mathcal{Y}} L(F(\mathbf{x}),\mathbf{y})dP(\mathbf{x},\mathbf{y}).$$

Given training data, we calculate the empirical risk function as follows,

$$\hat{R}(F) = \frac{1}{m}\sum_{i=1}^{m} L(F(\mathbf{x}_i),\mathbf{y}_i).$$

The learning task then becomes the minimization of the empirical risk function, as in other learning tasks. The minimization of the empirical risk function could be difficult due to the nature of the loss function (it is not continuous and it uses sorting). We can consider using a surrogate loss function $L'(F(\mathbf{x}),\mathbf{y})$.

The corresponding empirical risk function is defined as follows.

$$\hat{R}'(F) = \frac{1}{m}\sum_{i=1}^{m} L'(F(\mathbf{x}_i),\mathbf{y}_i).$$

We can also introduce a regularizer to conduct minimization of the regularized empirical risk. In such cases, the learning problem becomes minimization of the (regularized) empirical risk function based on the surrogate loss.

Note that we adopt a machine learning formulation here. In IR, the feature vectors $\mathbf{x}$ are derived from a query and its associated documents. The grades $\mathbf{y}$ represent the relevance degrees of the documents with respect to the query. We make use of a global ranking function $F(\cdot)$. In practice, it can be a local ranking function $f(\cdot)$. The possible number of feature vectors in $\mathbf{x}$ can be very large, even infinite. The evaluation (loss function) is, however, only concerned with n results.

In IR, the true loss functions can be those defined based on NDCG (Normalized Discounted Cumulative Gain) and MAP (Mean Average Precision). For example, we can have

$$L(F(\mathbf{x}),\mathbf{y}) = 1.0 - NDCG.$$

Note that the true loss functions (NDCG loss and MAP loss) makes use of sorting based on $F(\mathbf{x})$.

For the surrogate loss function, there are also different ways to define it, which lead to different approaches to learning to rank. For example, one can define pointwise loss, pairwise loss, and listwise loss functions.

The squared loss used in Subset Regression is a pointwise surrogate loss [6]. We call it pointwise loss, because it is defined on *single objects*.

$$L'(F(\mathbf{x}),\mathbf{y}) = \sum_{i=1}^{n}(f(x_i)-y_i)^2.$$

It is actually an upper bound of $1.0 - NDCG$.

Pairwise losses can be the hinge loss, exponential loss, and logistic loss on *pairs of objects*, which are used in Ranking SVM [7], RankBoost [8], and RankNet [9], respectively. They are also upper bounds of $1.0 - NDCG$ [10].

$$L'(F(\mathbf{x}),\mathbf{y}) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\phi(\mathrm{sign}(y_i-y_j), f(x_i)-f(x_j)),$$

where it is assumed that $L'=0$ when $y_i=y_j$ and ϕ is the hinge loss, exponential loss, or logistic loss function.

Listwise loss functions are defined on *lists of objects*, just like the true loss functions, and thus are more directly related to the true loss functions. Different listwise loss functions are exploited in the listwise methods. For example, the loss function in AdaRank is a listwise loss.

$$L'(F(\mathbf{x}), \mathbf{y}) = \exp(-NDCG),$$

where $NDCG$ is calculated on the basis of $F(\mathbf{x})$ and $\mathbf{y}$. Obviously, it is also an upper bound of $1.0 - NDCG$.

3. Pointwise Approach

In the pointwise approach, the ranking problem (ranking creation) is transformed to classification, regression, or ordinal classification, and existing methods for classification, regression, or ordinal classification are applied. Therefore, the group structure of ranking is ignored in this approach.

The pointwise approach includes Subset Ranking [6], McRank [11], Prank [12], and OC SVM [13]. We take the last one as an example and describe it in detail.

3.1 SVM for Ordinal Classification

The method proposed by Shashua & Levin [13] utilizes a number of parallel hyperplanes as a ranking model. Their method, referred to as OC SVM in this article, learns the parallel hyperplanes by the large margin principle. In one implementation, the method tries to maximize a fixed margin for all the adjacent classes (grades).[†]

Suppose that $\mathcal{X} \subseteq \Re^d$ and $\mathcal{Y} = \{1, 2, \cdots, l\}$ where there exists a total order on $\mathcal{Y}$. $x \in \mathcal{X}$ is an object (feature vector) and $y \in \mathcal{Y}$ is a label representing a grade. Given object x, we aim to predict its label (grade) y. That is to say, this is an ordinal classification problem. We employ a number of linear models (parallel hyperplanes) $\langle w, x\rangle - b_r, (r = 1, \cdots, l-1)$ to make the prediction, where $w \in \Re^d$ is a weight vector and $b_r \in \Re, (r = 1, \cdots, l)$ are biases satisfying $b_1 \leq \cdots \leq b_{l-1} \leq b_l = +\infty$. The models correspond to parallel hyperplanes $\langle w, x\rangle - b_r = 0$ separating grades r and $r+1$, $(r = 1, \cdots, l-1)$. Figure 3 illustrates the model. If x satisfies $\langle w, x\rangle - b_{r-1} \geq 0$ and $\langle w, x\rangle - b_r < 0$, then $y = r$, $(r = 1, \cdots, l)$. We can write it as $\min_{r \in \{1, \cdots, l\}} \{r | \langle w, x\rangle - b_r < 0\}$.

Suppose that the training data is given as follows. For each grade $r = 1, \cdots, l$, there are m_r instances: $x_{r,i}, i = 1, \cdots, m_r$. The learning task is formalized as the following Quadratic Programming (QP) problem.

$$\begin{gathered}
\min_{w,b,\xi} \frac{1}{2}||w||^2 + C\sum_{r=1}^{l-1}\sum_{i=1}^{m_r}(\xi_{r,i} + \xi^*_{r+1,i}) \\
\text{s. t. } \langle w, x_{r,i}\rangle + b_r \geq 1 - \xi_{r,i} \\
\langle w, x_{r+1,i}\rangle + b_r \leq 1 - \xi^*_{r+1,i} \\
\xi_{r,i} \geq 0, \quad \xi^*_{r+1,i} \geq 0 \\
i = 1, \cdots, m_r, \quad r = 1, \cdots, l-1 \\
m = m_1 + \cdots + m_l,
\end{gathered}$$

where $x_{r,i}$ denotes the i-th instance in the r-th grade, $\xi_{r+1,i}$ and $\xi^*_{r+1,i}$ denote the corresponding slack variables, $||\cdot||$ denotes L_2 norm, m denotes the number of training instances, and $C > 0$ is a coefficient. The method tries to separate the instances in the neighboring grades with the same margin.

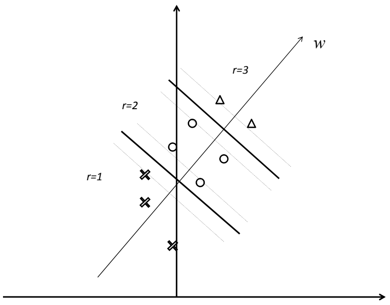

Fig. 3 SVM for Oridinal Classification

4. Pairwise Approach

In the pairwise approach, ranking is transformed into pairwise classification or pairwise regression. In the former case, a classifier for classifying the ranking orders of document pairs is created and is employed in the ranking of documents. In the pairwise approach, the group structure of ranking is also ignored.

The pairwise approach includes Ranking SVM [7], RankBoost [8], RankNet [9], GBRank [14], IR SVM [15], Lambda Rank [16], and LambdaMART [17]. We introduce Ranking SVM and IR SVM in this article.

4.1 Ranking SVM

We can learn a classifier, such as SVM, for *classifying the order of pairs of objects* and utilize the classifier in the ranking task. This is the idea behind the Ranking SVM method proposed by Herbrich et al. [7].

Figure 4 shows an example of the ranking problem. Suppose that there are two groups of objects (documents associated with two queries) in the feature space. Further suppose that there are three grades (levels). For example, objects x_1, x_2, and x_3 in the first group are at three different grades. The weight vector w corresponds to the linear function $f(x) = \langle w, x\rangle$ which can score and rank the objects. Ranking objects with the function is equivalent to projecting the objects into the vector and sorting the objects according to the projections on the vector. If the ranking function is 'good', then there should be an effect that objects at grade 3 are ranked ahead of objects at grade 2, etc. Note that objects belonging to different groups are incomparable.

Figure 5 shows that the ranking problem in Figure 4 can be transformed to Linear SVM classification. The differences between two feature vectors at different

[†] The other method maximizes the sum of all margins.

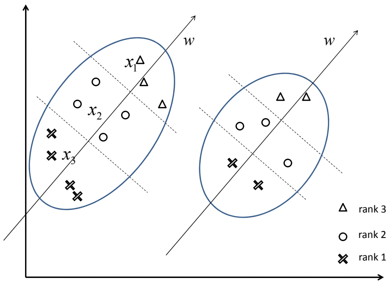

Fig. 4 Example of Ranking Problem

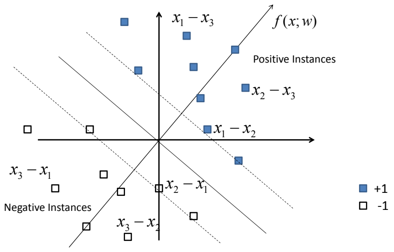

Fig. 5 Transformation to Pairwise Classification

Grade: 3, 2, 1
Documents are represented by their grades
Example 1:
ranking-1: 2 3 2 1 1 1 1
ranking-2: 3 2 1 2 1 1 1
Example 2:
ranking for query-1: 3 2 2 1 1 1 1
ranking for query-2: 3 3 2 2 2 1 1 1 1 1

Fig. 6 Example Ranking Lists

grades in the same group are treated as new feature vectors, e.g., $x_1 - x_2$, $x_1 - x_3$, and $x_2 - x_3$. Furthermore, labels are also assigned to the new feature vectors. For example, $x_1 - x_2$, $x_1 - x_3$, and $x_2 - x_3$ are positive. Note that feature vectors at the same grade or feature vectors from different groups are not utilized to create new feature vectors. One can train a Linear SVM classifier which separates the new feature vectors as shown in Figure 5. Geometrically, the margin in the SVM model represents the closest distance between the projections of object pairs in two grades. Note that the hyperplane of the SVM classifier passes the original and the positive and negative instances form corresponding pairs. For example, $x_1 - x_2$ and $x_2 - x_1$ are positive and negative instances respectively. The weight vector w of the SVM classifier corresponds to the ranking function. In fact, we can discard the negative instances in learning, because they are redundant.

Training data is given as $\{((x_i^{(1)}, x_i^{(2)}), y_i)\}, i = 1, \cdots, m$ where each instance consists of two feature vectors $(x_i^{(1)}, x_i^{(2)})$ and a label $y_i \in \{+1, -1\}$ denoting which feature vector should be ranked ahead.

The learning of Ranking SVM is formalized as the following QP problem.

$$\begin{array}{c} \min_{w,\xi} \frac{1}{2}||w||^2 + C\sum_{i=1}^{m} \xi_i \\ \text{s. t. } y_i\langle w, x_i^{(1)} - x_i^{(2)}\rangle \geq 1 - \xi_i \\ \xi_i \geq 0 \\ i = 1, \ldots, m, \end{array}$$

where $x_i^{(1)}$ and $x_i^{(2)}$ denote the first and second feature vectors in a pair of feature vectors, $|| \cdot ||$ denotes L_2 norm, m denotes the number of training instances, and $C > 0$ is a coefficient.

It is equivalent to the following non-constrained optimization problem, i.e., the minimization of the regularized hinge loss function.

$$\min_{w} \sum_{i=1}^{m} [1 - y_i\langle w, x_i^{(1)} - x_i^{(2)}\rangle]_+ + \lambda||w||^2, \tag{5}$$

where $[x]_+$ denotes function $\max(x, 0)$ and $\lambda = \frac{1}{2C}$.

4.2 IR SVM

IR SVM proposed by Cao et al. [15] is an extension of Ranking SVM for Information Retrieval (IR), whose idea can be applied to other applications as well.

Ranking SVM transforms ranking into pairwise classification, and thus it actually makes use of the 0-1 loss in the learning process. There exists a gap between the loss function and the IR evaluation measures. IR SVM attempts to bridge the gap by modifying 0-1 loss, that is, conducting cost sensitive learning of Ranking SVM.

We first look at the problems caused by straightforward application of Ranking SVM to document retrieval, using examples in Figure 6.

One problem with the direction application of Ranking SVM is that Ranking SVM equally treats document pairs across different grades. Example 1 indicates the problem. There are two rankings for the same query. The documents at positions 1 and 2 are swapped in ranking-1 from the perfect ranking, while the documents at positions 3 and 4 are swapped in ranking-2 from the perfect ranking. There is only one error for each ranking in terms of the 0-1 loss, or difference in order of pairs. They have the same effect on the training of Ranking SVM, which is not desirable. Ranking-2 should be better than ranking-1, from the viewpoint of IR, because the result on its top is better. Note that to have high accuracy on top-ranked documents is crucial for an IR system, which is reflected in the IR evaluation measures.

Another issue with Ranking SVM is that it equally treats document pairs from different queries. In example 2, there are two queries and the numbers of documents associated with them are different. For query-1 there are 2 document pairs between grades 3-2, 4 document pairs between grades 3-1, 8 document pairs between grades 2-1, and in total 14 document pairs. For query-2, there are 31 document pairs. Ranking SVM takes 14 instances (document pairs) from query-1 and 31 instances (document pairs) from query-2 for training. Thus, the impact on the ranking model from query-2 will be larger than the impact from query-1. In other words, the model learned will be biased toward query-2. This is in contrast to the fact that in IR evaluation queries are evenly important. Note that the numbers of documents usually vary from query to query.

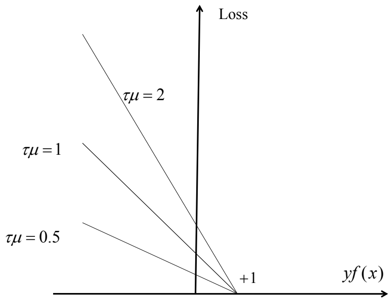

Fig. 7 Modified Hinge Loss Functions

IR SVM addresses the above two problems by changing the 0-1 pairwise classification into a cost sensitive pairwise classification. It does so by modifying the hinge loss function of Ranking SVM. Specifically, it sets different losses for document pairs across different grades and from different queries. To emphasize the importance of correct ranking on the top, the loss function heavily penalizes errors related to the top. To increase the influence of queries with less documents, the loss function heavily penalizes errors from the queries.

Figure 7 plots the shapes of different hinge loss functions with different penalty parameters. The x-axis represents $yf(x)$ and the y-axis represents loss. When $yf(x_i^{(1)} - x_i^{(2)}) \geq 1$, the losses are zero. When $yf(x_i^{(1)} - x_i^{(2)}) < 1$, the losses are represented by linearly decreasing functions with different slopes. If the slope equals -1, then the function is the normal hinge loss function. IR SVM modifies the hinge loss function, specifically modifies the slopes for different grade pairs and different queries. It assigns higher weights to document pairs across important grade pairs and assigns normalization weights to document pairs according to queries.

The learning of IR SVM is equivalent to the following optimization problem. Specifically, the minimization of the modified regularized hinge loss function,

$$\min_w \sum_{i=1}^{m} \tau_{k(i)}\mu_{q(i)}[1 - y_i\langle w, x_i^{(1)} - x_i^{(2)}\rangle]_+ + \lambda||w||^2,$$

where $[x]_+$ denotes $\max(x, 0)$, $\lambda = \frac{1}{2C}$, and $\tau_{k(i)}$ and $\mu_{q(i)}$ are weights. See the loss function of Ranking SVM (5).

Here $\tau_{k(i)}$ represents the weight of instance (document pair) i whose label pair belongs to the k-th type. Xu et al. propose a heuristic method to determine the value of τ_k. The method takes the average drop in NDCG@1 when randomly changing the positions of documents belonging to the grade pair as the value of a grade pair τ_k. Moreover, $\mu_{q(i)}$ represents the weight of instance (document pair) i which is from query q. The value of $\mu_{q(i)}$ is simply determined by $\frac{1}{|n_q|}$, where n_q is the number of document pairs for query q.

The equivalent QP problem is as below.

$$\begin{aligned}&\min_{w,\xi} \tfrac{1}{2}||w||^2 + C_i \textstyle\sum_{i=1}^{m} \xi_i \\ &\text{s. t. } y_i\langle w, x_i^{(1)} - x_i^{(2)}\rangle \geq 1 - \xi_i, \\ &C_i = \tfrac{\tau_{k(i)}\mu_{q(i)}}{2\lambda} \\ &\xi_i \geq 0, \\ &i = 1, \ldots, m.\end{aligned}$$

5. Listwise Approach

The listwise approach addresses the ranking problem in a more straightforward way. Specifically, it takes ranking lists as instances in both learning and prediction. The group structure of ranking is maintained and ranking evaluation measures can be more directly incorporated into the loss functions in learning.

The listwise approach includes ListNet [18], ListMLE [19], AdaRank [20], SVM MAP [21], and Soft Rank [22]. SVM MAP and related methods are explained in this article.

5.1 SVM MAP

The algorithm SVM MAP developed by Yue et al. [21] is designed to directly optimize MAP [2], but it can be easily extended to optimize NDCG. Xu et al. [23] further generalize it to a group of algorithms.

In ranking, for query q_i the ranking model $f(x_{ij})$ assigns a score to each associated document d_{ij} or feature vector x_{ij} where x_{ij} is the feature vector derived from q_i and d_{ij}. The documents $\mathbf{d}_i$ (feature vectors $\mathbf{x}_i$) are then sorted based on their scores and a permutation denoted as π_i is obtained. For simplicity, suppose that the ranking model $f(x_{ij})$ is a linear model:

$$f(x_{ij}) = \langle w, x_{ij}\rangle, \tag{6}$$

where w denotes a weight vector.

Suppose that labels for the feature vectors $\mathbf{x}_i$ are also given as $\mathbf{y}_i$. We consider using a scoring function $S(\mathbf{x}_i, \pi_i)$ to measure the goodness of ranking π_i. $S(\mathbf{x}_i, \pi_i)$ is defined as

$$S(\mathbf{x}_i, \pi_i) = \langle w, \sigma(\mathbf{x}_i, \pi_i) \rangle,$$

where w is still the weight vector and vector $\sigma(\mathbf{x}_i, \pi_i)$ is defined as

$$\sigma(\mathbf{x}_i, \pi_i) = \frac{2}{n_i(n_i - 1)} \sum_{k,l:k<l} [z_{kl}(x_{ik} - x_{il})],$$

where $z_{kl} = +1$, if $\pi_i(k) < \pi_i(l)$ (x_{ik} is ranked ahead of x_{il} in π_i), and $z_{kl} = -1$, otherwise. Recall that n_i is the number of documents associated with query q_i.

For query q_i, we can calculate $S(\mathbf{x}_i, \pi_i)$ for each permutation π_i and select the permutation $\tilde{\pi}_i$ with the largest score:

$$\tilde{\pi}_i = \arg \max_{\pi_i \in \Pi_i} S(\mathbf{x}_i, \pi_i), \tag{7}$$

where Π_i denotes the set of all possible permutations for $\mathbf{x}_i$.

It can be easily shown that the ranking $\tilde{\pi}_i$ selected by Eq.(7) is equivalent to the ranking created by the ranking model $f(x_{ij})$ (when both of them are linear functions). Figure 8 gives an example. It is easy to verify that both $f(x)$ and $S(\mathbf{x}_i, \pi)$ will output ABC as the most preferable ranking (permutation).

Objects: A, B, C
$f_A = \langle w, x_A \rangle, f_B = \langle w, x_B \rangle, f_C = \langle w, x_C \rangle$
Suppose $f_A > f_B > f_C$
For example:
Permutation1: ABC
Permutation2: ACB
$S_{ABC} = \frac{1}{6}\langle w, ((x_A - x_B) + (x_B - x_C) + (x_A - x_C)) \rangle$
$S_{ACB} = \frac{1}{6}\langle w, ((x_A - x_C) + (x_C - x_B) + (x_A - x_B)) \rangle$
$S_{ABC} > S_{ACB}$

Fig. 8 Example of Scoring Function

In learning, we would ideally create a ranking model that can maximize the accuracy in terms of a listwise evaluation measure on training data, or equivalently, minimizes the loss function defined below,

$$L(f) = \sum_{i=1}^{m} (E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)), \tag{8}$$

where π_i is the permutation on feature vector $\mathbf{x}_i$ by ranking model f and $\mathbf{y}_i$ is the corresponding list of grades. $E(\pi_i, \mathbf{y}_i)$ denotes the evaluation result of π_i in terms of an evaluation measure (e.g., NDCG). Usually $E(\pi_i^*, \mathbf{y}_i) = 1$.

We view the problem of learning a ranking model as the following optimization problem in which the following loss function is minimized.

$$\begin{aligned} &\textstyle\sum_{i=1}^{m} \max_{\pi_i^* \in \Pi_i^*; \pi_i \in \Pi_i \setminus \Pi_i^*} (E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)) \\ &\cdot [[S(\mathbf{x}_i, \pi_i^*) \leq S(\mathbf{x}_i, \pi_i)]], \end{aligned} \tag{9}$$

where $[[c]]$ is one if condition c is satisfied, otherwise it is zero. $\pi_i^* \in \Pi_i^* \subseteq \Pi_i$ denotes any of the perfect permutations for q_i.

The loss function measures the loss when the most preferred ranking list by the ranking model is not the perfect ranking list. One can prove that the true loss function such as that in (8) is upper-bounded by the new loss function in (9).

The loss function (9) is still not continuous and differentiable. We can consider using continuous, differentiable, and even convex upper bounds of the loss function (9).

1) The 0-1 function in (9) can be replaced with its upper bounds, for example, hinge functions, yielding

$$\begin{aligned} &\textstyle\sum_{i=1}^{m} \max_{\pi_i^* \in \Pi_i^*, \pi_i \in \Pi_i \setminus \Pi_i^*} (E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)) \cdot \\ &[1 - (S(\mathbf{x}_i, \pi_i^*) - S(\mathbf{x}_i, \pi_i))]_+ \end{aligned}$$

$$\begin{aligned} &\textstyle\sum_{i=1}^{m} \Big[\max_{\pi_i^* \in \Pi_i^*, \pi_i \in \Pi_i \setminus \Pi_i^*} ((E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)) \\ &- (S(\mathbf{x}_i, \pi_i^*) - S(\mathbf{x}_i, \pi_i)))]_+ , \end{aligned}$$

2) The max function can also be replaced with its upper bound, the sum function. This is because $\sum_i x_i \geq \max_i x_i$ if $x_i \geq 0$ holds for all i.

3) Relaxations 1 and 2 can be applied simultaneously.

For example, using the hinge function and taking the true loss as $1.0 - MAP$, we obtain SVM MAP. More precisely, SVM MAP solves the following QP problem:

$$\begin{aligned} &\min_{w;\xi \geq 0} \tfrac{1}{2}||w||^2 + \tfrac{C}{m} \textstyle\sum_{i=1}^{m} \xi_i \\ &s.t. \quad \forall i, \forall \pi_i^* \in \Pi_i^*, \forall \pi_i \in \Pi_i \setminus \Pi_i^* : \\ &S(\mathbf{x}_i, \pi_i^*) - S(\mathbf{x}_i, \pi_i) \geq E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i) - \xi_i, \end{aligned} \tag{10}$$

where C is a coefficient and ξ_i is the maximum loss among all the losses for permutations of query q_i.

Equivalently, SVM MAP minimizes the following regularized hinge loss function

$$\begin{aligned} &\textstyle\sum_{i=1}^{m} \Big[\max_{\pi_i^* \in \Pi_i^*; \pi_i \in \Pi_i \setminus \Pi_i^*} (E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)) \\ &- (S(\mathbf{x}_i, \pi_i^*) - S(\mathbf{x}_i, \pi_i))]_+ + \lambda ||w||^2. \end{aligned} \tag{11}$$

Intuitively, the first term calculates the total maximum loss when selecting the best permutation for each of the queries. Specifically, if the difference between the permutations $S(\mathbf{x}_i, \pi_i^*) - S(\mathbf{x}_i, \pi_i)$ is less than the difference between the corresponding evaluation measures $E(\pi_i^*, \mathbf{y}_i) - E(\pi_i, \mathbf{y}_i)$, then there will be a loss, otherwise not. Next, the maximum loss is selected for each query and they are summed up over all the queries.

Since $c \cdot [[x \leq 0]] < [c - x]_+$ holds for all $c \in \Re^+$ and $x \in \Re$, it is easy to see that the loss in (11) also bounds the true loss function in (8).

6. Ongoing and Future Work

It is still necessary to develop more advanced technologies for learning to rank. There are also many open questions with regard to theory and applications of learning to rank [2], [24]. Current and future research directions include

- training data creation
- semi-supervised learning and active learning
- feature learning
- scalable and efficient training
- domain adaptation and multi-task learning
- ranking by ensemble learning
- global ranking
- ranking of nodes in a graph.

References

[1] T.Y. Liu, "Learning to rank for information retrieval," Foundations and Trends in Information Retrieval, vol.3, no.3, pp.225–331, 2009.

[2] H. Li, "Learning to rank for information retrieval and natural language processing," Synthesis Lectures on Human Language Technologies, 2011, Morgan & Claypool Publishers.

[3] W.B. Croft, D. Metzler, and T. Strohman, Search Engines - Information Retrieval in Practice, Pearson Education, 2009.

[4] T.Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li, "LETOR: Benchmark dataset for research on learning to rank for information retrieval," Proceedings of SIGIR 2007 Workshop on Learning to Rank for Information Retrieval, 2007.

[5] K. Järvelin and J. Kekäläinen, "IR evaluation methods for retrieving highly relevant documents," Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR '00, New York, NY, USA, pp.41–48, ACM, 2000.

[6] D. Cossock and T. Zhang, "Subset ranking using regression.," COLT '06: Proceedings of the 19th Annual Conference on Learning Theory, pp.605–619, 2006.

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

[8] Y. Freund, R.D. Iyer, R.E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences.," Journal of Machine Learning Research, vol.4, pp.933–969, 2003.

[9] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," ICML '05: Proceedings of the 22nd international conference on Machine learning, pp.89–96, 2005.

[10] W. Chen, T.Y. Liu, Y. Lan, Z.M. Ma, and H. Li, "Ranking measures and loss functions in learning to rank," NIPS '09, 2009.

[11] P. Li, C. Burges, and Q. Wu, "McRank: Learning to rank using multiple classification and gradient boosting," in Advances in Neural Information Processing Systems 20, ed. J. Platt, D. Koller, Y. Singer, and S. Roweis, pp.897–904, MIT Press, Cambridge, MA, 2008.

[12] K. Crammer and Y. Singer, "Pranking with ranking.," NIPS, pp.641–647, 2001.

[13] A. Shashua and A. Levin, "Ranking with large margin principle: Two approaches," in Advances in Neural Information Processing Systems 15, ed. S.T. S. Becker and K. Obermayer, MIT Press.

[14] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun, "A general boosting method and its application to learning ranking functions for web search," in Advances in Neural Information Processing Systems 20, ed. J. Platt, D. Koller, Y. Singer, and S. Roweis, pp.1697–1704, MIT Press, Cambridge, MA, 2008.

[15] Y. Cao, J. Xu, T.Y. Liu, H. Li, Y. Huang, and H.W. Hon, "Adapting ranking SVM to document retrieval," SIGIR' 06, pp.186–193, 2006.

[16] C. Burges, R. Ragno, and Q. Le, "Learning to rank with nonsmooth cost functions," in Advances in Neural Information Processing Systems 18, pp.395–402, MIT Press, Cambridge, MA, 2006.

[17] Q. Wu, C.J.C. Burges, K.M. Svore, and J. Gao, "Adapting boosting for information retrieval measures," Inf. Retr., vol.13, no.3, pp.254–270, 2010.

[18] Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," ICML '07: Proceedings of the 24th international conference on Machine learning, pp.129–136, 2007.

[19] F. Xia, T.Y. Liu, J. Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank: theory and algorithm," ICML '08: Proceedings of the 25th international conference on Machine learning, New York, NY, USA, pp.1192–1199, ACM, 2008.

[20] J. Xu and H. Li, "AdaRank: a boosting algorithm for information retrieval," SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, pp.391–398, ACM, 2007.

[21] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," Proceedings of the 30th annual international ACM SIGIR conference, pp.271–278, 2007.

[22] M. Taylor, J. Guiver, S. Robertson, and T. Minka, "SoftRank: optimizing non-smooth rank metrics," WSDM '08: Proceedings of the international conference on Web search and web data mining, New York, NY, USA, pp.77–86, ACM, 2008.

[23] J. Xu, T.Y. Liu, M. Lu, H. Li, and W.Y. Ma, "Directly optimizing evaluation measures in learning to rank," SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, pp.107–114, ACM, 2008.

[24] O. Chapelle, Y. Chang, and T.Y. Liu, "Future directions in learning to rank," Journal of Machine Learning Research - Proceedings Track, vol.14, pp.91–100, 2011.

Hang Li Senior researcher and research manager in Web Search and Mining Group at Microsoft Research Asia. He joined Microsoft Research in June 2001. Prior to that, He worked at the Research Laboratories of NEC Corporation. He obtained a B.S. in Electrical Engineering from Kyoto University in 1988 and a M.S. in Computer Science from Kyoto University in 1990. He earned his Ph.D. in Computer Science from the University of Tokyo in 1998. He is interested in statistical learning, information retrieval, data mining, and natural language processing.