Learning to Rank: From Pairwise Approach to Listwise Approach

Zhe Cao  Tao Qin  Tie-Yan Liu  Ming-Feng Tsai  Hang Li

Microsoft Research Asia, Beijing (2007)

Presented by Christian Kümmerle
December 2, 2014
Learning to Rank: A Listwise Approach

1. Framework: Learning to Rank
2. The Listwise Approach
3. Loss function based on probability model
4. ListNet algorithm
5. Experiments and Conclusion
What is Learning to Rank?
What is Learning to Rank?

Classical IR ranking task: Given a query, rank documents to a list.

- Query-dependent ranking functions:
  Vector space model, BM25, Language model

- Query-independent features of documents: e.g.
  ▶ PageRank
  ▶ URL-depth, e.g. http://sifaka.cs.uiuc.edu/~wang296/Course/IR_Fall/lectures.html has a depth of 4
What is Learning to Rank?

Classical IR ranking task: Given a query, rank documents to a list.

- Query-dependent ranking functions:
  - Vector space model, BM25, Language model
- Query-independent features of documents: e.g.
  - PageRank
  - URL-depth, e.g.
    - http://sifaka.cs.uiuc.edu/~wang296/Course/IR_Fall/lectures.html
    - has a depth of 4
Framework: Learning to Rank

What is Learning to Rank?

Classical IR ranking task: Given a query, rank documents to a list.

- Query-dependent ranking functions:
  Vector space model, BM25, Language model
- Query-independent features of documents: e.g.
  ▶ PageRank
  ▶ URL-depth, e.g.
  http://sifaka.cs.uiuc.edu/~wang296/Course/IR_Fall/lectures.html has a depth of 4

→ How can we combine all these "features" in order to get a better ranking function?
What is Learning to Rank?

Idea: Learn the *best* way to combine the features from given *training data*, consisting of queries and corresponding *labelled* documents.

Supervised learning:

$$X = \{x_1, x_2, ..., x_i\}, \quad y = \{y_1, y_2, ..., y_i\}$$

$$y_i$$: List of judgements of the relevance degree of the documents for $$q_i$$ ← Listwise approach
What is Learning to Rank?

Idea: Learn the best way to combine the features from given training data, consisting of queries and corresponding labelled documents.

Supervised learning:

- Input space
- Output space
- Hypothesis space
- Loss function
What is Learning to Rank?

Idea: Learn the best way to combine the features from given training data, consisting of queries and corresponding labelled documents.

Supervised learning: In the authors’ paper:

- **Input space:** \( X = \{x^{(1)}, x^{(2)}, \ldots \} \), \( x^{(i)} \): List of feature representations of documents for query \( q_i \leftarrow \text{Listwise approach} \)
- **Output space:** \( Y = \{y^{(1)}, y^{(2)}, \ldots \} \), \( y^{(i)} \): List of judgements of the relevance degree of the documents for \( q_i \leftarrow \text{Listwise approach} \)

- **Hypothesis space**
- **Loss function**
What is Learning to Rank?

Idea: Learn the best way to combine the features from given training data, consisting of queries and corresponding labelled documents.

Supervised learning: In the authors’ paper:

- **Input space**: \( X = \{ x^{(1)}, x^{(2)}, \ldots \} \), \( x^{(i)} \): List of feature representations of documents for query \( q_i \) ← Listwise approach
- **Output space**: \( Y = \{ y^{(1)}, y^{(2)}, \ldots \} \), \( y^{(i)} \): List of judgements of the relevance degree of the documents for \( q_i \) ← Listwise approach
- **Hypothesis space** ← **Neural network**
- **Loss function**
What is Learning to Rank?

Idea: Learn the best way to combine the features from given training data, consisting of queries and corresponding labelled documents.

Supervised learning: In the authors’ paper:

- Input space: $X = \{x^{(1)}, x^{(2)}, \ldots\}$, $x^{(i)}$: List of feature representations of documents for query $q_i$ ← Listwise approach
- Output space: $Y = \{y^{(1)}, y^{(2)}, \ldots\}$, $y^{(i)}$: List of judgements of the relevance degree of the documents for $q_i$ ← Listwise approach
- Hypothesis space ← Neural network
- Loss function: Probability model on the space of permutations
The Listwise Approach

- **Queries:** \( Q = \{q^{(1)}, q^{(2)}, \ldots, q^{(m)}\} \) a set of \( m \) queries.
- **List of documents:** For query \( q^{(i)} \), there are \( n_i \) documents:
  \( d^{(i)} = (d_1^{(i)}, d_2^{(i)}, \ldots, d_{n_i}^{(i)}) \).
The Listwise Approach

- **Queries**: $Q = \{ q^{(1)}, q^{(2)}, \ldots, q^{(m)} \}$ a set of $m$ queries.
- **List of documents**: For query $q^{(i)}$, there are $n_i$ documents: $d^{(i)} = (d_1^{(i)}, d_2^{(i)}, \ldots, d_{n_i}^{(i)})$.
- **Feature representation in input space**: $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_{n_i}^{(i)})$ with $x_j^{(i)} = \Psi(q^{(i)}, d_j^{(i)})$, e.g.
  
  \[ x_j^{(i)} = (\text{BM25}(q^{(i)}, d_j^{(i)}), \text{LM}(q^{(i)}, d_j^{(i)}), \text{TFIDF}(q^{(i)}, d_j^{(i)}), \text{PageRank}(d_j^{(i)}), \text{URLdepth}(d_j^{(i)})) \in \mathbb{R}^5 \]
The Listwise Approach

- **Queries:** $Q = \{ q^{(1)}, q^{(2)}, \ldots, q^{(m)} \}$ a set of $m$ queries.
- **List of documents:** For query $q^{(i)}$, there are $n_i$ documents:
  $d^{(i)} = (d_{1}^{(i)}, d_{2}^{(i)}, \ldots, d_{n_i}^{(i)})$.
- **Feature representation in input space:** $x^{(i)} = (x_{1}^{(i)}, x_{2}^{(i)}, \ldots, x_{n_i}^{(i)})$
  with $x_{j}^{(i)} = \Psi(q^{(i)}, d_{j}^{(i)})$, e.g.
  \[ x_{j}^{(i)} = (\text{BM25}(q^{(i)}, d_{j}^{(i)}), \text{LM}(q^{(i)}, d_{j}^{(i)}), \text{TFIDF}(q^{(i)}, d_{j}^{(i)}), \text{PageRank}(d_{j}^{(i)}), \text{URLdepth}(d_{j}^{(i)})) \in \mathbb{R}^5 \]
- **List of judgment scores in output space:** $y^{(i)} = (y_{1}^{(i)}, y_{2}^{(i)}, \ldots, y_{n_i}^{(i)})$
  with implicitly or explicitly given judgement scores $y_{j}^{(i)}$ for all documents corresponding to query $q^{(i)}$.

$\rightarrow$ **Training data set** $\mathcal{T} = \{ (x^{(i)}, y^{(i)}) \}_{i=1}^{m}$
What is a meaningful loss function?

We want: Find a function $f : X \rightarrow Y$ such that the $f(x^{(i)})$ are "not very different" from the $y^{(i)}$. \rightarrow \textbf{Loss function} penalizes too big differences.
What is a meaningful loss function?

We want: Find a function \( f : X \rightarrow Y \) such that the \( f(x^{(i)}) \) are "not very different" from the \( y^{(i)} \). \( \longrightarrow \) Loss function penalizes too big differences.

Idea: Just take NDCG! Perfectly ordered list can be derived from the given judgements \( y^{(i)} \).

Problem: Discontinuity of NDCG with respect to the ranking scores, since NDCG is position based:

Example

<table>
<thead>
<tr>
<th>Training query with ( NDCG = 1 )</th>
<th>Training query with ( NDCG = 0.86 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(x^{(i)}) )</td>
<td>1.2</td>
</tr>
<tr>
<td>( y^{(i)} )</td>
<td>2</td>
</tr>
<tr>
<td>( f(x^{(i)}) )</td>
<td>1.2</td>
</tr>
<tr>
<td>( y^{(i)} )</td>
<td>2</td>
</tr>
</tbody>
</table>
Solution: Define probability distributions $P_{y(i)}$ and $P_{z(i)}$ (for $z^{(i)} := (f(x_1^{(i)}), \ldots, f(x_{n_i}^{(i)}))$) on the set of permutations $\pi$ on $\{1, \ldots, n_i\}$, take the KL divergence as loss function:

\[
L(y^{(i)}, z^{(i)}) := -\sum_{\pi} P_{y(i)}(\pi) \log(P_{z(i)}(\pi)) \propto \text{KL}(P_{y(i)}(\cdot) \parallel P_{z(i)}(\cdot))
\]
Loss function based on probability model on permutations

Solution: Define probability distributions $P_{y(i)}$ and $P_{z(i)}$ (for $z^{(i)} := (f(x^{(i)}_1), \ldots, f(x^{(i)}_{n_i}))$ on the set of permutations $\pi$ on \{1, \ldots, $n_i$\}, take the KL divergence as loss function:

$$L(y^{(i)}, z^{(i)}) := -\sum_{\pi} P_{y(i)}(\pi) \log(P_{z(i)}(\pi)) \propto \text{KL}(P_{y(i)}(\cdot) \ |\ | P_{z(i)}(\cdot))$$

How to define the probability distribution? E.g. for the set of permutations on \{1, 2, 3\}, the scores $(y_1, y_2, y_3)$ and the permutation $\pi := (1, 3, 2)$:

$$P_y(\pi) := \frac{e^{y_1}}{e^{y_1} + e^{y_2} + e^{y_3}} \cdot \frac{e^{y_3}}{e^{y_2} + e^{y_3}} \cdot \frac{e^{y_2}}{e^{y_2}}$$

Definition

If $\pi$ is a permutation on \{1, \ldots, $n$\}, its probability, given the list of scores $y$ of length $n$, is:

$$P_y(\pi) = \prod_{j=1}^{n} \frac{\exp(y_{\pi^{-1}(j)})}{\sum_{l=j}^{n} \exp(y_{\pi^{-1}(l)})}$$
Loss function based on probability model on permutations

**Solution:** Define probability distributions $P_{y(i)}$ and $P_{z(i)}$ (for $z^{(i)} := (f(x_1^{(i)}), \ldots, f(x_{n_i}^{(i)}))$) on the set of permutations $\pi$ on $\{1, \ldots, n_i\}$, take the KL divergence as loss function:

$$L(y^{(i)}, z^{(i)}) := -\sum_{\pi} P_{y(i)}(\pi) \log(P_{z(i)}(\pi)) \propto \text{KL}(P_{y(i)}(\cdot) \parallel P_{z(i)}(\cdot))$$

How to define the probability distribution? E.g. for the set of permutations on $\{1, 2, 3\}$, the scores $(y_1, y_2, y_3)$ and the permutation $\pi := (1, 3, 2)$:

$$P_{y}(\pi) := \frac{e^{y_1}}{e^{y_1} + e^{y_2} + e^{y_3}} \cdot \frac{e^{y_3}}{e^{y_2} + e^{y_3}} \cdot \frac{e^{y_2}}{e^{y_2}}$$

**Definition**

For easier calculation we rather use in the algorithm with $k$ fixed:

$$P_{y}(\pi) = \prod_{j=1}^{k} \frac{\exp(y_{\pi^{-1}(j)})}{\sum_{l=j}^{n} \exp(y_{\pi^{-1}(l)})}$$
The ListNet algorithm

**Advantage:** Loss function is differentiable with respect to the score vectors! We use functions $f_\omega$ from a *Neural Network model* as a hypothesis space.

---> Learning task: $\min_\omega \sum_{i=1}^m L(y^{(i)}, f_\omega(x^{(i)}))$
The ListNet algorithm

**Advantage:** Loss function is differentiable with respect to the score vectors! → We use functions \( f_\omega \) from a *Neural Network model* as a hypothesis space.

→ Learning task: 
\[
\min_\omega \sum_{i=1}^{m} L(y^{(i)}, f_\omega(x^{(i)}))
\]

---

**Algorithm 1** Learning Algorithm of ListNet

**Input:** training data \( \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\} \)

Parameter: number of iterations \( T \) and learning rate \( \eta \)

Initialize parameter \( \omega \)

for \( t = 1 \) to \( T \) do

for \( i = 1 \) to \( m \) do

Input \( x^{(i)} \) of query \( q^{(i)} \) to Neural Network and compute score list \( z^{(i)}(f_\omega) \) with current \( \omega \)

Compute gradient \( \Delta \omega \) using Eq. (5)

Update \( \omega = \omega - \eta \times \Delta \omega \)

end for

end for

Output Neural Network model \( \omega \)
Experiments and Conclusion

- Authors compared ranking accuracy of ListNet with other LtR algorithms on three large scale data sets (TREC, OHSUMED and Csearch; Number of features: 20, 30 and 600)
- Procedure: Divide data into training subset and testing subset, use traditional evaluation metrics (NDCG, MAP) on testing set.

Conclusions:
- ListNet outperforms algorithms based on pairwise approach (RankNet, RankingSVM, RankBoost)
- Drawback: High training complexity ($O(n^k)$ for list length $n$ and parameter $k$)
Authors compared ranking accuracy of ListNet with other LtR algorithms on three large scale data sets (TREC, OHSUMED and Csearch; Number of features: 20, 30 and 600).

Procedure: Divide data into training subset and testing subset, use traditional evaluation metrics (NDCG, MAP) on testing set.

Conclusions:

- ListNet outperforms algorithms based on pairwise approach (RankNet, RankingSVM, RankBoost).
- Drawback: High training complexity ($O(n^k)$ for list length $n$ and parameter $k$).
Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.
Learning to Rank: From Pairwise Approach to Listwise Approach.

Tie-Yan Liu.
Learning to rank for information retrieval.

Thank you for your attention!