Learning to Rank: From Pairwise Approach to Listwise Approach

Seminar Machine Learning
Buu Kieu Lam
Supervisor: Sang-Hyeun Park

Overview

- Motivation
- Definition
  - Ranking
  - Listwise approach
- Probability Model
  - Top one probability
- Learning method: ListNet
  - Learning Algorithm
- Experiments
- Conclusions

Motivation

Example: Document retrieval

- Pairwise approach:
  - Instances: document pairs
  - the problem of learning to rank ≈ classification

    + existing methodologies on classification can be directly applied.

      E.g.: Ranking SVM, RankBoost, RankNet

    + training instances of document pairs can be easily obtained

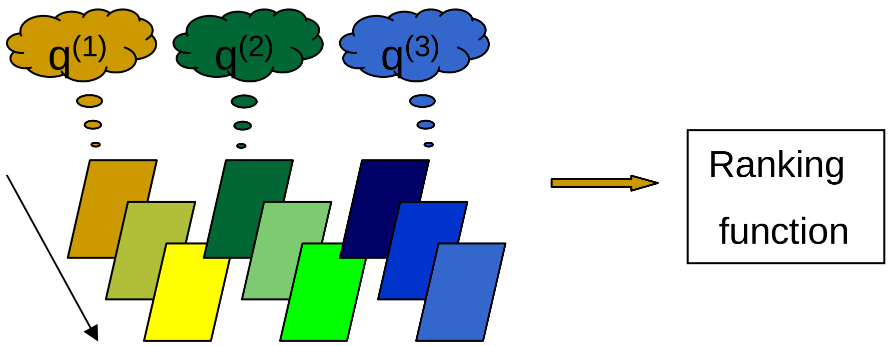
    - minimize errors in classification of document pairs rather than in ranking
    - number of document pairs is very large ➔ training process costly
      - n*(n-1)/2 document pairs
    - the number of generated document pairs varies largely from query to query

    → result in training a model biased toward queries with more document pairs.
- Listwise approach
  - Instances in learning: document lists

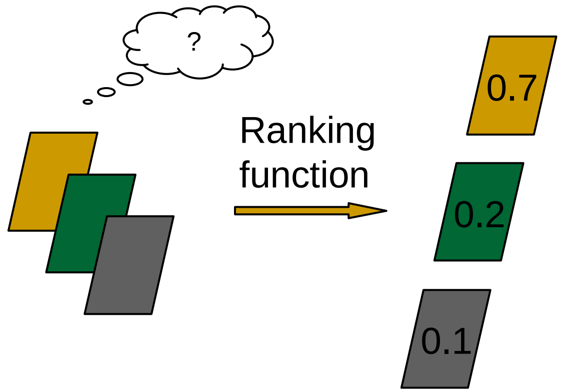
Ranking

Learning to rank: construct a model or a function for ranking objects.

- In learning
  - Given are a number of queries



- In evalutation (i.e. ranking):



- Ranking order represents relative relevance of documents with respect to the query

Listwise approach

- Set of queries $Q = \{q^{(i)}\}, i=1,2,..,m$
  - List of documents $d^{(i)} = \{d^{(i)}_j\}$
    - List of judgments (scores) $y^{(i)} = \{y^{(i)}_j\}$
    - Feature vector $x^{(i)}_j = \psi(q^{(i)},d^{(i)}_j)$ for each query-document pair
- Instance
  - (feature list, judgment list) = $(x^{(i)},y^{(i)})$
  - Training set $\{(x^{(i)},y^{(i)})\}$
- Ranking function f
  - Ranking list: $z^{(i)} = (f(x^{(i)}_j))$
- The objective of learning:
  - L is a listwise loss function

$$\boxed{\min \sum_{i=1}^{m} L(y^{(i)}, z^{(i)})}$$

Listwise approach
abstract

Predicted score list

Correct score list

transform

Probability distributions:

➔Any metric betw. Probability distributions = loss function

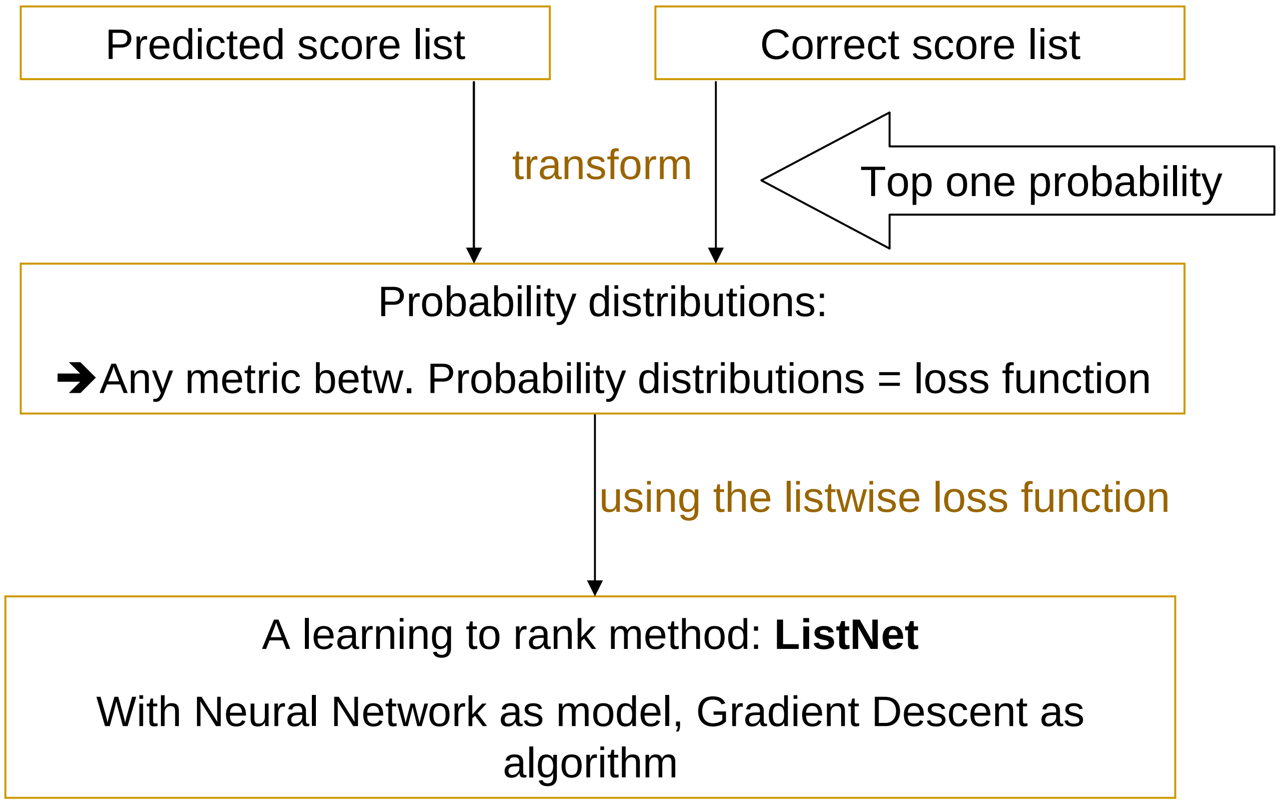
using the listwise loss function

A learning to rank method: **ListNet**

With Neural Network as model, Gradient Descent as algorithm

Listwise approach

abstract



Top One Probability

- The probability of an object *j* being ranked on the top
- <u>Given:</u>
  - scores of all the objects $s = (s_1, s_2, \ldots, s_n)$
  - an increasing and strictly positive function $\Phi(.)$
- <u>Define:</u>

$$P_s(j) = \frac{\phi(s_j)}{\sum_{k=1}^{n} \phi(s_k)} \quad , \quad s_j : \text{score of object } j,\ j = 1, 2, \ldots, n$$

- Given 2 lists of scores: use any metric to represent the distance (listwise loss function) between the two score lists: e.g. Cross Entropy as metric:

$$\boxed{L(y^{(i)}, z^{(i)}) = -\sum_{j=1}^{n} P_{y^{(i)}}(j) \log(P_{z^{(i)}}(j))}$$

ListNet: Learning Algorithm

ranking function based on Neural Network model ω as f_ω

- **Input:** training data $\{(x^{(1)},y^{(1)}), (x^{(2)},y^{(2)}),\ldots,(x^{(m)},y^{(m)})\}$
- Parameter: number of iterations *T* and learning rate *η*
- Initialize parameter ω
- 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 ω
    - Compute gradient Δω

$$\Delta\omega = \frac{\partial L(y^{(i)}, z^{(i)}(f_\omega))}{\partial\omega}$$

    - update ω = ω - *η**Δω
  - end for
- end for
- **Output:** Neural Network model ω

Experiments

Data Collections

	TREC 2003 Web pages from .gov domain	OHSHUMED Documents, queries in medicine	CSearch Data set from a commercial web search engine
Volume	1,053,110 pages 11,164,829 hyperlinks	348,566 documents	
Number of queries	50	106	25,000 *Each query: 1,000 associated documents*
Number of features *Extracted from each query-document pair*	20	30 *(16,140 query-document pairs)*	600 *Query-dependent/ independent features*
Relevance judgments	Relevant or irrelevant	Definitely relevant, possibly relevant, or not relevant	5 levels: 4 (perfect match) → 0 (bad match)
Using of 2 common IR evaluation measures	NDCG & MAP	NDCG & MAP	NDCG

Ranking performance evaluation - measure ranking accuracy: Normalized Discounted Cumulative Gain (NDCG) (for >=2 levels of relevance judgment) & Mean Avarage Precision (MAP) (for relevance judgment with 2 levels)

Experiments
Ranking Accuracy (1)

- TREC & OSHUMED:
  - Divide data set into 5 subsets → 5-fold cross-validation

→ RankNet & ListNet: determine the number of iterations *T*

→ *Ranking SVM: use for parameter tuning*

→ *RankBoost: select the number of weak learners*

Experiments

Ranking Accuracy (2)

- TREC

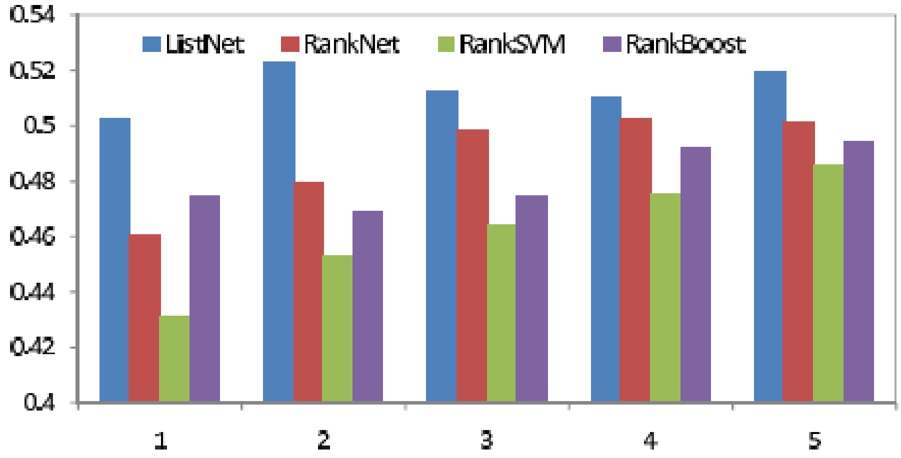


Figure 1. Ranking accuracies in terms of NDCG@n on TREC

Table 1. Ranking accuracies in terms of MAP

ALGORITHMS	LISTNET	RANKBOOST	RANKSVM	RANKNET
TREC	0.216	0.174	0.193	0.197
OHSUMED	0.305	0.297	0.297	0.303

- ListNet outperforms RankNet, RankingSVM and RankBoost.

Experiments

Ranking Accuracy (3)

- OSHUMED

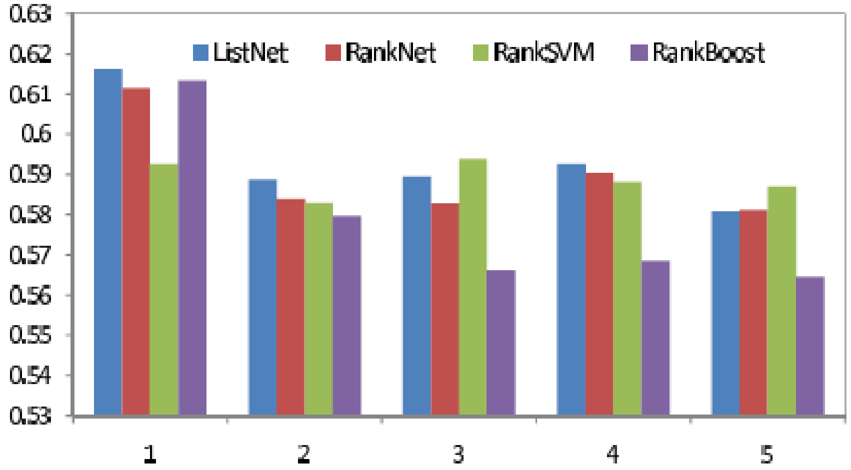


Figure 2. Ranking accuracies in terms of NDCG@n on OHSUMED

Table 1. Ranking accuracies in terms of MAP

ALGORITHMS	LISTNET	RANKBOOST	RANKSVM	RANKNET
TREC	0.216	0.174	0.193	0.197
OHSUMED	0.305	0.297	0.297	0.303

- ListNet outperforms RankNet and RankBoost and better than RankingSVM in terms of MAP and partition in terms of NDCG.

Experiments
Ranking Accuracy (4)

- CSearch:
  - Randomly select

training	validation	testing

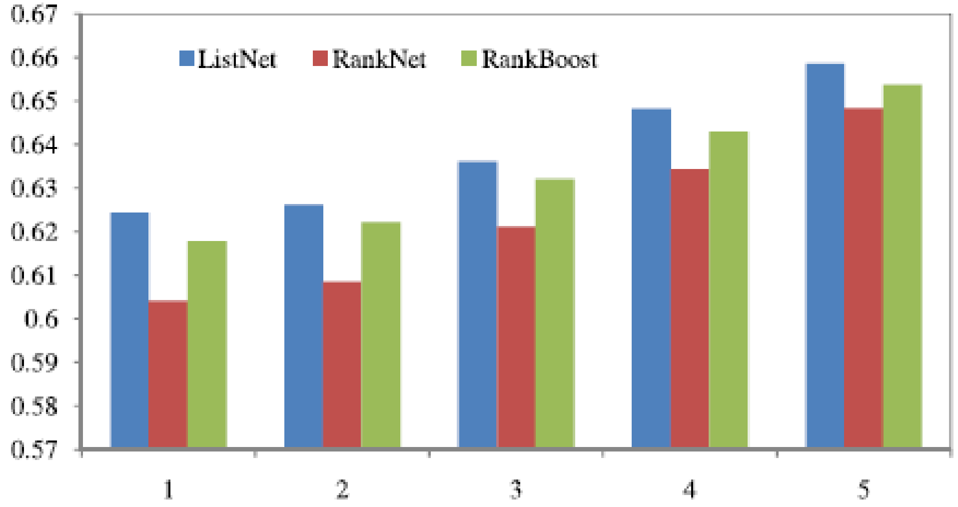


Figure 3. Ranking accuracies in terms of NDCG@n on CSearch

  - ListNet outperforms RankNet and RankBoost
  - Size of training data too large: → impossibly run RankingSVM with the SVMlight tool.

Experiments
Discussion (1)

- Pairwise loss function too loose as an approximation of the performance measures of NDCG and MAP.
- Pairwise loss does not inversely correlate with NDCG

- Listwise loss function can more properly represent the performance measures.
- Listwise loss inversely correlates with NDCG

Experiments

Discussion (2) - evaluation measure NDCG@5 on TREC

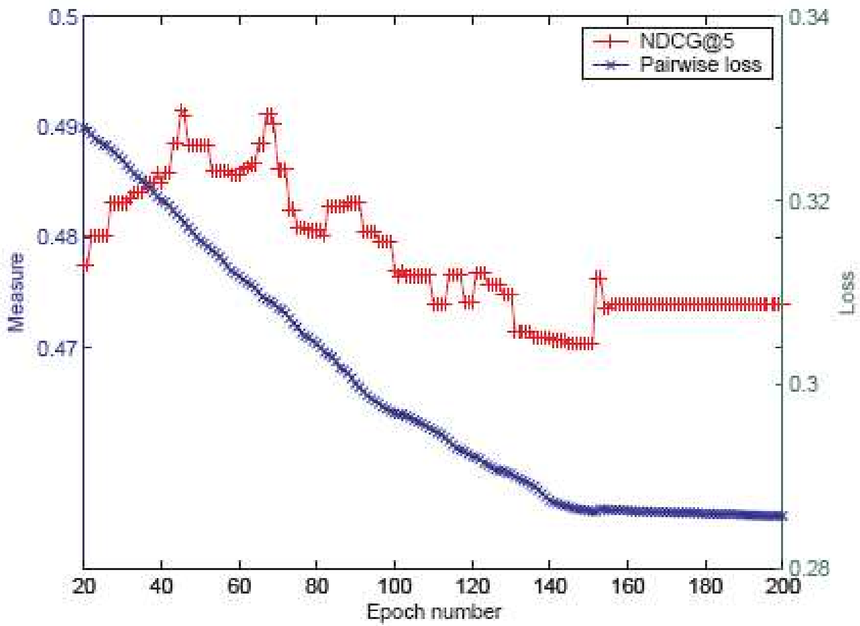


Figure 4. Pairwise loss v.s. NDCG@5 in RankNet

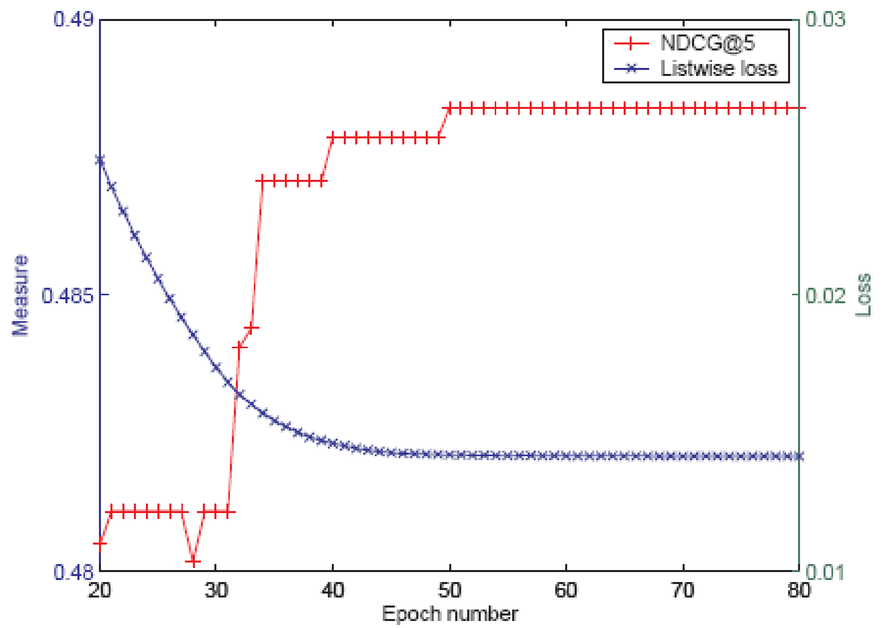


Figure 5. Listwise loss v.s. NDCG@5 in ListNet

- Pairwise loss converges more slowly than listwise loss

➔ RankNet needs more iterations in training than ListNet.

Conclusions

- In learning to rank: listwise approach better.
  - List of objects: instances in learning
  - Listwise loss function:
    - permutation probability and top one probability ➔ ranking scores into probability distribution
    - any metric between probability distributions (e.g. cross entropy) as the listwise loss function
  - Develop a learning method based on the approach
    - Neural Network as model
    - Gradient Descent as algorithm
- Experiment results ➔ proved!
- Future work: explore
  - The performance of other objective function besides cross entropy
  - The performance of other ranking model instead of linear Neural Network model
  - NDCG and MAP performance measures with listwise loss function

Any Questions?