Learning to Rank

Nicholas Ruozzi
University of Texas at Dallas

based on the slides of Tie-Yan Liu & Thorsten Joachims

Course Evaluations

- Take 5-10 minutes and go to

eval.utdallas.edu

Ranking

- In many "information retrieval" applications, the goal is, given a query, to return relevant documents
  - Application: search engines
    - Lots of data
    - Lots of available features: anchor texts, PageRank score, click through data
    - Most search companies use some form of ML to rank query results
  - Other applications: collaborative filtering, key term extraction, sentiment analysis, product ratings

Ranking

- Early document retrieval systems used no learning at all
  - Relevance of a query was determined purely by analyzing the content of each document via some heuristic approach
  - The difficulty: not clear in advance which features/document properties are most relevant & parameter tuning non-trivial
- Different types of ranking systems
  - Pointwise: learn a relevance score for each query/document pair
  - Pairwise: learn the relative ordering of each pair of documents for a given query

Generation of Labeled Data

- Human evaluators
  - Set of queries is randomly selected from the query log
  - Each query is associated with multiple documents
  - Human judges are asked to evaluate relevance typically in five levels, for example, perfect, excellent, good, fair, and bad
- Search log data (e.g., clickthroughs)
  - Use the information about which links users have clicked on for specific queries as a measure of relevance

Clickthroughs

1. **Kernel Machines**
   http://svm.first.gmd.de/
2. Support Vector Machine
   http://jbolivar.freeservers.com/
3. **SVM-Light Support Vector Machine**
   http://ais.gmd.de/ ~ thorsten/svm_light/
4. An Introduction to Support Vector Machines
   http://www.support − vector.net/
5. Support Vector Machine and Kernel Methods References
   http://svm.research.bell − labs.com/SVMrefs.html
6. Archives of SUPPORT-VECTOR-MACHINES@JISCMAIL.AC.UK
   http://www.jiscmail.ac.uk/lists/SUPPORT−VECTOR−MACHINES.html
7. **Lucent Technologies: SVM demo applet**
   http://svm.research.bell − labs.com/SVT/SVMsvt.html
8. Royal Holloway Support Vector Machine
   http://svm.dcs.rhbnc.ac.uk/
9. Support Vector Machine - The Software
   http://www.support − vector.net/software.html
10. Lagrangian Support Vector Machine Home Page
    http://www.cs.wisc.edu/dmi/lsvm

Ranking presented for the query “support vector machine”. Marked in bold are the links the user clicked on. [Joachims 2003]

Learning to Rank

- Supervised learning problem
  - Training includes data queries and document relevance scores:
    - Set of queries $Q = \{q_1, .., q_m\}$
    - Set of documents D
    - Documents relevant to the i^{th} query $D_i = \{d_{i,1}, \dots, d_{n_i}\}$
    - Vector of relevance scores $y_i = (y_{i,1}, \dots, y_{i,n_i})$ for each document relevant to query i
  - Goal: Given a new query q, output a sorted list of (a permutation) of relevant documents

Simple (Pointwise) Relevance

- Training data:
  - Pairs of queries and documents with a corresponding plus or minus one to indicate whether or not the document is relevant to the query
  - Typically, query/document pairs are converted into feature vectors that include features such as PageRank score, number of times the query keyword appears in the document, etc.
  - Use SVMs to learn to predict which documents are relevant to which queries

Mean Average Precision (MAP)

- Precision at position k for query q is the number of fraction of relevant documents in the top k results

$$p_{q,k} = \frac{\#\ relevant\ docs\ in\ top\ k\ results}{k}$$

- Average precision

$$AP(q) = \frac{\sum_k p_{q,k} \cdot I_{k^{th} document\ is\ relevant}}{\#\ of\ relevant\ documents}$$

- $\text{MAP} = \frac{\sum_q AP(q)}{\#\ queries}$

Pointwise Approach

- Training data:
  - Pairs of queries and documents with a corresponding label representing a score
    - If $score(q, d_1) > score(q, d_2)$, then d_1 is more relevant to query q than d_2
  - As before, query-document pairs are converted into feature vectors that include features such as PageRank score, number of times the query keyword appears in the document, etc.
  - Use multiclass SVMs to learn to predict the labels for query-document pairs

Simple (Pointwise) Relevance

- Can also solve this as a regression task
  - Instead of ordered labels, use real numbers
  - Minimize the squared loss between the estimated relevance score and the true relevance score

Drawbacks of Pointwise Ranking

- Ignores the fact that some documents are associated with the same query and some are not
- If the number of documents varies largely for different queries, the overall loss function will be dominated by those queries with a large number of documents
- The position of documents in the ranked list is not factored into the loss function(s)

Pairwise Approach

- Consider pairs of documents at a time
  - We are really interested in the ranking of documents for each query which corresponds to a permutation of the documents
  - Can recover the permutation if we know the relative ordering of pairs of documents for a given query

Ranking SVM

- A ranking matrix r for a query q is a $|D| \times |D|$ matrix whose $(i, j)^{th}$ entry is a 1 if d_i is ranked higher than d_j for the query q and 0 otherwise
- Training data: $(q_1, r_1), \ldots, (q_m, r_m)$
- Objective: given a new q, predict its corresponding r
- This can be approximated as a convex optimization problem similar to the SVM objective…

Kendall's Tau

- Measures the difference between two rankings
- Consider two rankings r and r'
  - The pair $d_i \neq d_j$ is concordant if r and r' agree on the relative ordering of d_i and d_j
  - Else the pair d_i and d_j are said to be discordant

$$\tau(r, r') = \frac{(\#conc.\ pairs) - (\#disc.\ pairs)}{\big((\#conc.\ pairs) + (\#disc.\ pairs)\big)}$$

Ranking SVM

- The Ranking SVM algorithm attempts to minimize the following loss function

$$\frac{1}{m}\sum_{i}^{m} -\tau(r_{f(q_i)}, r_i)$$

where $r_{f(q_i)}$ is the predicted ranking for the query q_i

- When r is thought of as a strict ordering, minimizing this loss is equivalent to minimizing the number of discordant pairs for each query

Ranking SVM

- The idea, express the ordering relations using linear inequalities

$$d_{ki} >_{q_k} d_{kj} \leftrightarrow w^T \phi(q_k, d_{ki}) > w^T \phi(q_k, d_{kj})$$

- Formulating this (approximately) as a max margin optimization problem (with slack):

$$\min_w \frac{1}{2} w^T w + C \sum_{i,j,k} \epsilon_{i,j,k}$$

$$\forall k \; and \; i \neq j \in \{1, \dots, n_k\} \; with \; d_{ki} >_{q_k} d_{kj},$$
$$w^T \phi(q_k, d_{ki}) \geq w^T \phi(q_k, d_{kj}) + 1 - \epsilon_{i,j,k}$$

$$\epsilon_{i,j,k} \geq 0$$

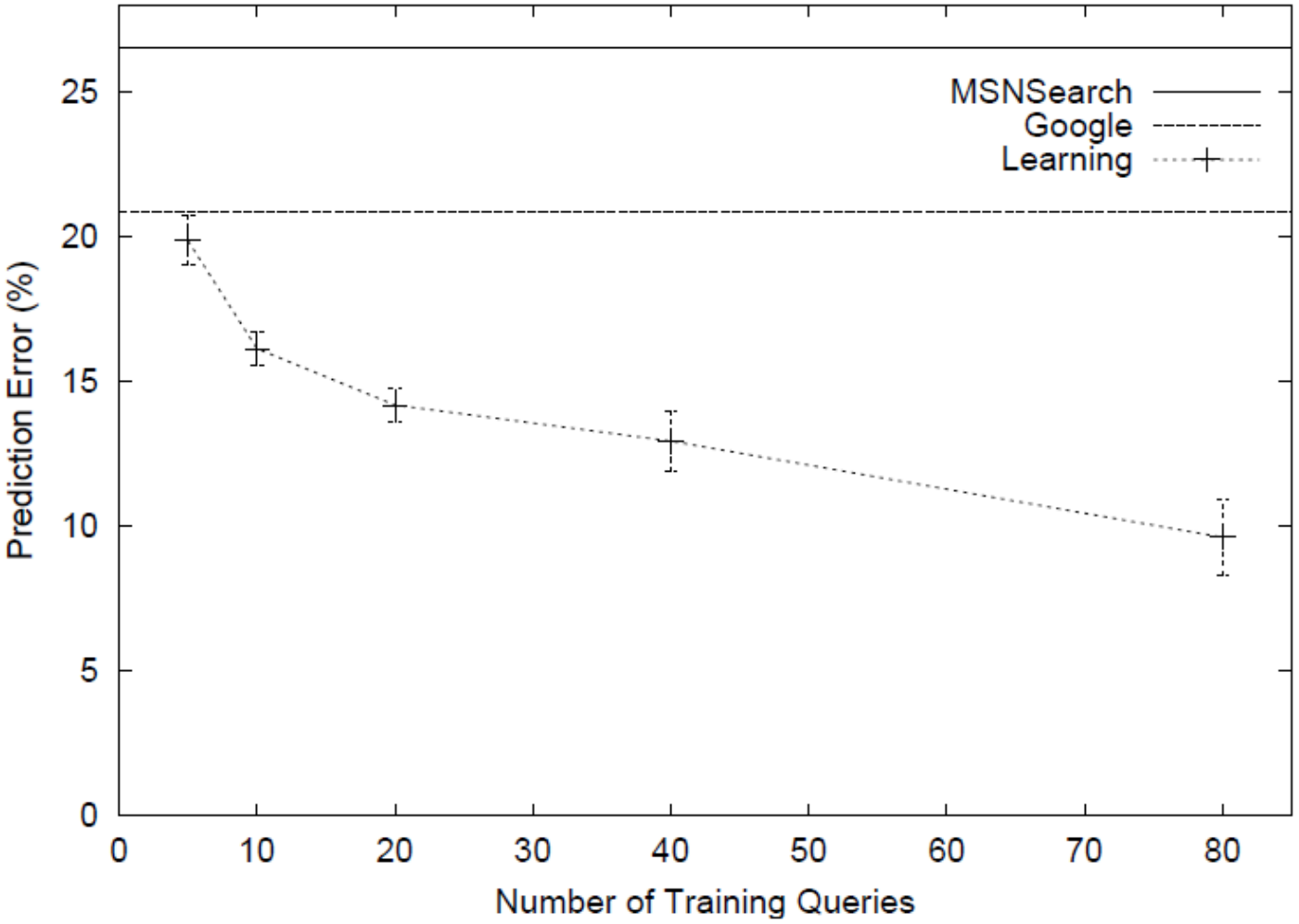
- Can take the dual and apply the kernel trick…

Ranking SVM

Comparison	more clicks on learned	less clicks on learned
Learned vs. Google	29	13
Learned vs. MSNSearch	18	4
Learned vs. Toprank	21	9

[Joachims 2003]

Ranking SVM



[Joachims 2003]

Pairwise Summary

- Predicting relative order is more like the ranking problem than considering query-document pairs
- The number of document pairs per query is quadratic in the number of documents for that query
  - Means that queries with many relevant documents account for most of the loss
  - Heuristic fix: introduce a normalizing constant into the SVM objective per query
    - Seems to work well in practice

Listwise Ranking

- The training data and loss function operate specifically over ordered lists in which each document related to a particular query receives a score

- One then develops a metric to evaluate the quality of a chosen permutation based on these scores

  – This generates a new loss function to minimize (SVM based methods can also be applied here)