

# Introduction to Information Retrieval

CS276: Information Retrieval and Web Search  
Christopher Manning and Pandu Nayak

Lecture 14: Learning to Rank

Introduction to Information Retrieval Sec. 15.4

## Machine learning for IR ranking?

- We've looked at methods for ranking documents in IR
  - Cosine similarity, inverse document frequency, BM25, proximity, pivoted document length normalization, (will look at) Pagerank, ...
- We've looked at methods for classifying documents using supervised machine learning classifiers
  - Rocchio, kNN, SVMs, etc.
- Surely we can also use *machine learning* to rank the documents displayed in search results?
  - Sounds like a good idea
  - A.k.a. "machine-learned relevance" or "learning to rank"

Introduction to Information Retrieval

LinkedIn People Jobs Answers Companies Account & Settings | Help | Sign Out | Language

Explore People Search: Harvard - Vice President at Google - Bill Gates Search Jobs Search Advanced

### Jobs

Jobs Home Advanced Job Search Hiring Solutions

Forward this job to a friend

#### Machine Learned Ranking (MLR)/Search Relevance Researcher and Engineers at Clients of PromptHire, Inc.

Location: San Francisco Bay Area - Burlingame (San Francisco Bay Area)  
URL: <http://prompthire.com/careers/index.php?m=careers&p=showJob&ID=31316>

Type: Full-time  
Experience: Mid-Senior level  
Functions: Research, Engineering  
Industries: Internet  
Posted: November 16, 2008 by Sangeeta Narayan

LinkedIn Exclusive - this job is available only on LinkedIn

**Job Description**

A typical candidate has made a not-trivial contribution into one of the major web search engine.

Specific areas of expertise include

- machine learned ranking - ranking features and training technologies
- query independent search quality - acquisition and analysis of high quality web content
- query and content classification

Posted by Sangeeta Narayan Recruiting Diva (sangeeta@prompthire.com) (Company HR)

32 people have recommended Sangeeta  
Read recommendations

2 Your connections know Sangeeta.  
See who connects you

Apply Now Request Referral

Introduction to Information Retrieval

## Machine learning for IR ranking

- This "good idea" has been actively researched - and actively deployed by major web search engines - in the last 7-10 years
- Why didn't it happen earlier?
  - Modern supervised ML has been around for about 20 years...
  - Naïve Bayes has been around for about 50 years...

Introduction to Information Retrieval

## Machine learning for IR ranking

- There's some truth to the fact that the IR community wasn't very connected to the ML community
- But there were a whole bunch of precursors:
  - Wong, S.K. et al. 1988. Linear structure in information retrieval. *SIGIR 1988*.
  - Fuhr, N. 1992. Probabilistic methods in information retrieval. *Computer Journal*.
  - Gey, F. C. 1994. Inferring probability of relevance using the method of logistic regression. *SIGIR 1994*.
  - Herbrich, R. et al. 2000. Large Margin Rank Boundaries for Ordinal Regression. *Advances in Large Margin Classifiers*.

Introduction to Information Retrieval

## Why weren't early attempts very successful/influential?

- Sometimes an idea just takes time to be appreciated...
- Limited training data**
  - Especially for real world use (as opposed to writing academic papers), it was very hard to gather test collection queries and relevance judgments that are representative of real user needs and judgments on documents returned
    - This has changed, both in academia and industry
- Poor machine learning techniques
- Insufficient customization to IR problem
- Not enough features for ML to show value

Introduction to Information Retrieval

### Why wasn't ML much needed?

- Traditional ranking functions in IR used a very small number of features, e.g.,
  - Term frequency
  - Inverse document frequency
  - Document length
- It was easy possible to tune weighting coefficients by hand
  - And people did
  - You students do it in PA3

Introduction to Information Retrieval

### Why is ML needed now?

- Modern systems – especially on the Web – use a great number of features:
  - Arbitrary useful features – not a single unified model
  - Log frequency of query word in anchor text?
  - Query word in color on page?
  - # of images on page?
  - # of (out) links on page?
  - PageRank of page?
  - URL length?
  - URL contains “~”?
  - Page edit recency?
  - Page loading speed
- The *New York Times* in 2008-06-03 quoted Amit Singhal as saying Google was using over 200 such features (“signals”)

Introduction to Information Retrieval Sec. 15.4.1

### Simple example: Using classification for ad hoc IR

- Collect a training corpus of  $(q, d, r)$  triples
  - Relevance  $r$  is here binary (but may be multiclass, with 3–7 values)
  - Document is represented by a feature vector
    - $\mathbf{x} = (\alpha, \omega)$   $\alpha$  is cosine similarity,  $\omega$  is minimum query window size
    - $\omega$  is the the shortest text span that includes all query words
    - Query term proximity is an **important** new weighting factor
  - Train a machine learning model to predict the class  $r$  of a document-query pair

example	docID	query	cosine score	$\omega$	judgment
$\Phi_1$	37	linux operating system	0.032	3	relevant
$\Phi_2$	37	penguin logo	0.02	4	nonrelevant
$\Phi_3$	238	operating system	0.043	2	relevant
$\Phi_4$	238	runtime environment	0.004	2	nonrelevant
$\Phi_5$	1741	kernel layer	0.022	3	relevant
$\Phi_6$	2094	device driver	0.03	2	relevant
$\Phi_7$	3191	device driver	0.027	5	nonrelevant

Introduction to Information Retrieval Sec. 15.4.1

### Simple example: Using classification for ad hoc IR

- A linear score function is then
 
$$Score(d, q) = Score(\alpha, \omega) = a\alpha + b\omega + c$$
- And the linear classifier is
 
$$Decide\ relevant\ if\ Score(d, q) > \theta$$
- ... just like when we were doing text classification

Introduction to Information Retrieval Sec. 15.4.1

### Simple example: Using classification for ad hoc IR

Introduction to Information Retrieval

### More complex example of using classification for search ranking [Nallapati 2004]

- We can generalize this to classifier functions over more features
- We can use methods we have seen previously for learning the linear classifier weights

Introduction to Information Retrieval

### An SVM classifier for information retrieval

[Nallapati 2004]

- Let relevance score  $g(r|d,q) = \mathbf{w} \cdot \mathbf{f}(d,q) + b$
- SVM training: want  $g(r|d,q) \leq -1$  for nonrelevant documents and  $g(r|d,q) \geq 1$  for relevant documents
- SVM testing: decide relevant iff  $g(r|d,q) \geq 0$
- Features are *not* word presence features (how would you deal with query words not in your training data?) but scores like the summed (log) tf of all query terms
- Unbalanced data (which can result in trivial always-say-nonrelevant classifiers) is dealt with by undersampling nonrelevant documents during training (just take some at random) [there are other ways of doing this – cf. Cao et al. later]

Introduction to Information Retrieval

### An SVM classifier for information retrieval

[Nallapati 2004]

- Experiments:
  - 4 TREC data sets
  - Comparisons with Lemur, a state-of-the-art open source IR engine (Language Model (LM)-based – see IIR ch. 12)
  - Linear kernel normally best or almost as good as quadratic kernel, and so used in reported results
  - 6 features, all variants of tf, idf, and tf.idf scores

Introduction to Information Retrieval

### An SVM classifier for information retrieval

[Nallapati 2004]

Train \ Test		Disk 3	Disk 4-5	WT10G (web)
TREC Disk 3	Lemur	<b>0.1785</b>	<b>0.2503</b>	0.2666
	SVM	0.1728	0.2432	<b>0.2750</b>
Disk 4-5	Lemur	<b>0.1773</b>	<b>0.2516</b>	0.2656
	SVM	0.1646	0.2355	<b>0.2675</b>

- At best the results are about equal to Lemur
  - Actually a little bit below
- Paper's advertisement: Easy to add more features
  - This is illustrated on a homepage finding task on WT10G:
    - Baseline Lemur 52% success@10, baseline SVM 58%
    - SVM with URL-depth, and in-link features: 78% success@10

Introduction to Information Retrieval

Sec. 15.4.2

### “Learning to rank”

- Classification probably isn't the right way to think about approaching ad hoc IR:
  - Classification problems: Map to an unordered set of classes
  - Regression problems: Map to a real value [Start of PA4]
  - Ordinal regression problems: Map to an *ordered* set of classes
    - A fairly obscure sub-branch of statistics, but what we want here
- This formulation gives extra power:
  - Relations between relevance levels are modeled
  - Documents are good versus other documents for query given collection; not an absolute scale of goodness

Introduction to Information Retrieval

### “Learning to rank”

- Assume a number of categories  $C$  of relevance exist
  - These are totally ordered:  $c_1 < c_2 < \dots < c_j$
  - This is the ordinal regression setup
- Assume training data is available consisting of document-query pairs represented as feature vectors  $\psi_i$  and relevance ranking  $c_i$
- We could do **point-wise learning**, where we try to map items of a certain relevance rank to a subinterval (e.g. Crammer et al. 2002 PRank)
- But most work does **pair-wise learning**, where the input is a pair of results for a query, and the class is the relevance ordering relationship between them

Introduction to Information Retrieval

### Point-wise learning

- Goal is to learn a threshold to separate each rank

Introduction to Information Retrieval Sec. 15.4.2

### Pairwise learning: The Ranking SVM

[Herbrich et al. 1999, 2000; Joachims et al. 2002]

- Aim is to classify instance pairs as correctly ranked or incorrectly ranked
  - This turns an ordinal regression problem back into a binary classification problem in an expanded space
- We want a ranking function  $f$  such that
 
$$c_j > c_k \text{ iff } f(\psi_j) > f(\psi_k)$$
- ... or at least one that tries to do this with minimal error
- Suppose that  $f$  is a linear function
 
$$f(\psi_j) = \mathbf{w} \cdot \psi_j$$

Introduction to Information Retrieval Sec. 15.4.2

### The Ranking SVM

[Herbrich et al. 1999, 2000; Joachims et al. 2002]

- Ranking Model:  $f(\psi_j)$

Introduction to Information Retrieval Sec. 15.4.2

### The Ranking SVM

[Herbrich et al. 1999, 2000; Joachims et al. 2002]

- Then (combining  $c_j > c_k$  iff  $f(\psi_j) > f(\psi_k)$  and  $f(\psi_j) = \mathbf{w} \cdot \psi_j$ ):
 
$$c_j > c_k \text{ iff } \mathbf{w} \cdot (\psi_j - \psi_k) > 0$$
- Let us then create a new instance space from such pairs:
 
$$\Phi_u = \Phi(d_j, d_k, q) = \psi_j - \psi_k$$

$$z_u = +1, 0, -1 \text{ as } c_j >, =, < c_k$$
- We can build model over just cases for which  $z_u = -1$
- From training data  $S = \{\Phi_u\}$ , we train an SVM

Introduction to Information Retrieval Sec. 15.4.2

### Two queries in the original space

Introduction to Information Retrieval Sec. 15.4.2

### Two queries in the pairwise space

Introduction to Information Retrieval Sec. 15.4.2

### The Ranking SVM

[Herbrich et al. 1999, 2000; Joachims et al. 2002]

- The SVM learning task is then like other examples that we saw before
- Find  $\mathbf{w}$  and  $\xi_u \geq 0$  such that
  - $\frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_u$  is minimized, and
  - for all  $\Phi_u$  such that  $z_u < 0$ ,  $\mathbf{w} \cdot \Phi_u \geq 1 - \xi_u$
- We can just do the negative  $z_u$ , as ordering is antisymmetric
- You can again use libSVM or SVMlight (or other SVM libraries) to train your model (SVMrank specialization)

## Aside: The SVM loss function

- The minimization

$$\min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_u$$

and for all  $\Phi_u$  such that  $z_u < 0$ ,  $\mathbf{w} \cdot \Phi_u \geq 1 - \xi_u$

- can be rewritten as

$$\min_{\mathbf{w}} (1/2C) \mathbf{w}^T \mathbf{w} + \sum \xi_u$$

and for all  $\Phi_u$  such that  $z_u < 0$ ,  $\xi_u \geq 1 - (\mathbf{w} \cdot \Phi_u)$

- Now, taking  $\lambda = 1/2C$ , we can reformulate this as

$$\min_{\mathbf{w}} \sum [1 - (\mathbf{w} \cdot \Phi_u)]_+ + \lambda \mathbf{w}^T \mathbf{w}$$

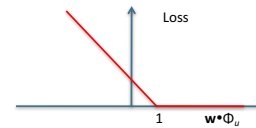
- Where  $[\ ]_+$  is the positive part (0 if a term is negative)

## Aside: The SVM loss function

- The reformulation

$$\min_{\mathbf{w}} \sum [1 - (\mathbf{w} \cdot \Phi_u)]_+ + \lambda \mathbf{w}^T \mathbf{w}$$

- shows that an SVM can be thought of as having an empirical “**hinge**” loss combined with a **weight regularizer** (smaller weights are preferred)



## Adapting the Ranking SVM for (successful) Information Retrieval

[Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, Hsiao-Wuen Hon SIGIR 2006]

- A Ranking SVM model already works well
  - Using things like vector space model scores as features
  - As we shall see, it outperforms standard IR in evaluations
- But it does not model important aspects of practical IR well
- This paper addresses two customizations of the Ranking SVM to fit an IR utility model

## The ranking SVM fails to model the IR problem well ...

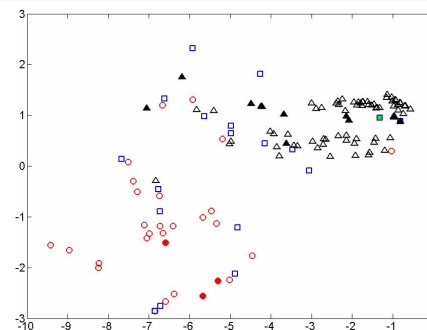
- Correctly ordering the most relevant documents is crucial to the success of an IR system, while misordering less relevant results matters little
  - The ranking SVM considers all ordering violations as the same
- Some queries have many (somewhat) relevant documents, and other queries few. If we treat all pairs of results for queries equally, queries with many results will dominate the learning
  - But actually queries with few relevant results are at least as important to do well on

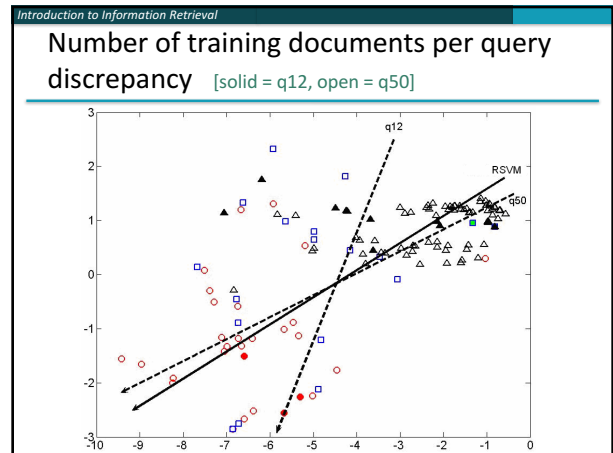
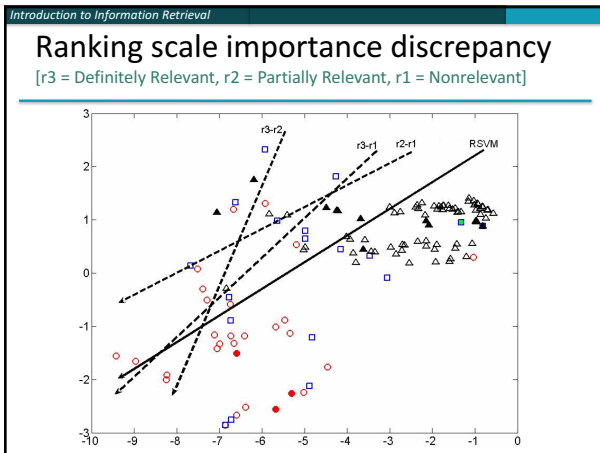
## Experiments use LETOR test collection

- <https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/>
  - From Microsoft Research Asia
  - An openly available standard test collection with pregenerated features, baselines, and research results for learning to rank
- It's availability has really driven research in this area
- OHSUMED, MEDLINE subcollection for IR
  - 350,000 articles
  - 106 queries
  - 16,140 query-document pairs
  - 3 class judgments: Definitely relevant (DR), Partially Relevant (PR), Non-Relevant (NR)
- TREC GOV collection (predecessor of GOV2, cf. IIR p. 142)
  - 1 million web pages
  - 125 queries

## Principal components projection of 2 queries

[solid = q12, open = q50; circle = DR, square = PR, triangle = NR]





Introduction to Information Retrieval

### IR Evaluation Measures

- Some evaluation measures strongly weight doing well in highest ranked results:
  - MAP (Mean Average Precision)
  - NDCG (Normalized Discounted Cumulative Gain)
- NDCG has been especially popular in machine learned relevance research
  - It handles multiple levels of relevance (MAP doesn't)
  - It seems to have the right kinds of properties in how it scores system rankings

Introduction to Information Retrieval

Sec. 8.4

### Normalized Discounted Cumulative Gain (NDCG) evaluation measure

- Query:  $q_i$
- DCG at position  $m$ :  $N_i = Z_i \sum_{j=1}^m (2^{r_j} - 1) / \log(1 + j)$
- NDCG at position  $m$ : average over queries
- Example
  - (3, 3, 2, 2, 1, 1, 1) rank  $r$
  - (7, 7, 3, 3, 1, 1, 1) gain  $2^{r_j} - 1$
  - (1, 0.63, 0.5, 0.43, 0.39, 0.36, 0.33) discount  $1 / \log(1 + j)$
  - (7, 11.41, 12.91, 14.2, 14.59, 14.95, 15.28)  $\sum_{j=1}^m (2^{r_j} - 1) / \log(1 + j)$
- $Z_i$  normalizes against best possible result for query, the above, versus lower scores for other rankings
  - Necessarily: High ranking number is good (more relevant)

Introduction to Information Retrieval

### Recap: Two Problems with Direct Application of the Ranking SVM

- Cost sensitivity: negative effects of making errors on top ranked documents
  - $d$ : definitely relevant,  $p$ : partially relevant,  $n$ : not relevant
  - ranking 1: p d p n n n n
  - ranking 2: d p n p n n n
- Query normalization: number of instance pairs varies according to query
  - q1: d p p n n n n
  - q2: d d p p n n n n n
  - q1 pairs:  $2*(d, p) + 4*(d, n) + 8*(p, n) = 14$
  - q2 pairs:  $6*(d, p) + 10*(d, n) + 15*(p, n) = 31$

Introduction to Information Retrieval

### These problems are solved with a new Loss function

$$\min_{\vec{w}} L(\vec{w}) = \sum_{i=1}^l \tau_{k(i)} \mu_{q(i)} \left[ 1 - z_i \langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \rangle \right]_+ + \lambda \|\vec{w}\|^2$$

- $\tau$  weights for type of rank difference
  - Estimated empirically from effect on NDCG
- $\mu$  weights for size of ranked result set
  - Linearly scaled versus biggest result set

Introduction to Information Retrieval

### Optimization (Gradient Descent)

$$\frac{\partial L}{\partial \vec{w}} = \sum_{i=1}^{\ell} \frac{\partial l_i(\vec{w})}{\partial \vec{w}} + 2\lambda \vec{w},$$

where  $\frac{\partial l_i(\vec{w})}{\partial \vec{w}} = \begin{cases} 0 & \text{if } z_i \langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \rangle \geq 1 \\ -z_i \tau_{k(i)} \mu_{q(i)} (\vec{x}_i^{(1)} - \vec{x}_i^{(2)}) & \text{if } z_i \langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \rangle < 1 \end{cases}$

Introduction to Information Retrieval

### Optimization (Quadratic Programming)

$$\min_{\vec{w}} L = \sum_{i=1}^{\ell} \tau_{k(i)} \mu_{q(i)} [1 - z_i \langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \rangle] + \lambda \|\vec{w}\|^2,$$

$$\min_{\vec{w}} M(\vec{w}) = \frac{1}{2} \|\vec{w}\|^2 + \sum_{i=1}^{\ell} C_i \xi_i$$

subject to  $\xi_i \geq 0, z_i \langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \rangle \geq 1 - \xi_i, i = 1, \dots, \ell$

where  $C_i = \frac{\tau_{k(i)} \mu_{q(i)}}{2\lambda}$

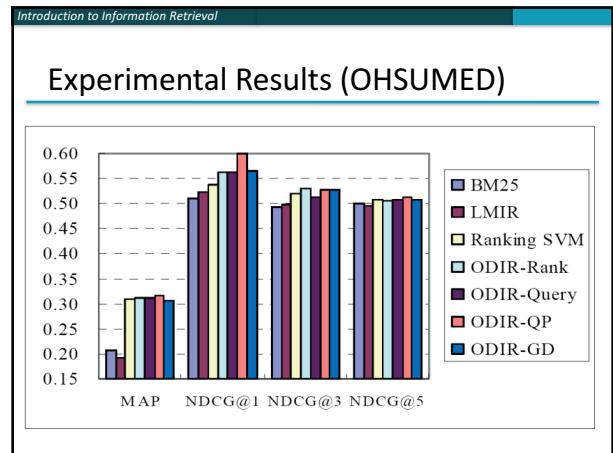
$$\max_{\alpha} L_D = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j z_i z_j \langle \vec{x}_i^{(1)} - \vec{x}_i^{(2)}, \vec{x}_j^{(1)} - \vec{x}_j^{(2)} \rangle$$

subject to  $0 \leq \alpha_i \leq C_i, i = 1, \dots, \ell$

Introduction to Information Retrieval

### Experiments

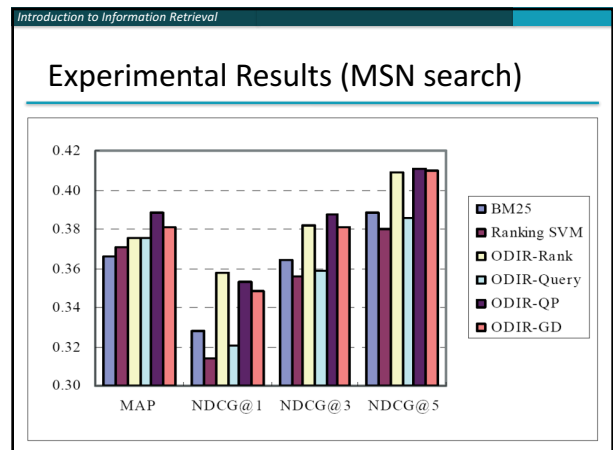
- OHSUMED (from LETOR)
- Features:
  - 6 that represent versions of tf, idf, and tf.idf factors
  - BM25 score (IJR sec. 11.4.3)



Introduction to Information Retrieval

### MSN Search [now Bing]

- Second experiment with MSN search
- Collection of 2198 queries
- 6 relevance levels rated:
  - Definitive: 8990
  - Excellent: 4403
  - Good: 3735
  - Fair: 20463
  - Bad: 36375
  - Detrimental: 310



Introduction to Information Retrieval

### Alternative: Optimizing Rank-Based Measures [Yue et al. SIGIR 2007]

- If we think that NDCG is a good approximation of the user's utility function from a result ranking
- Then, let's directly optimize this measure
  - As opposed to some proxy (weighted pairwise prefs)
- But, there are problems ...
  - Objective function no longer decomposes
    - Pairwise prefs decomposed into each pair
  - Objective function is flat or discontinuous

Introduction to Information Retrieval

### Discontinuity Example

- NDCG computed using rank positions
- Ranking via retrieval scores
- Slight changes to model parameters
  - Slight changes to retrieval scores
  - No change to ranking
  - No change to NDCG

NDCG = 0.63

	d <sub>1</sub>	d <sub>2</sub>	d <sub>3</sub>
Retrieval Score	0.9	0.6	0.3
Rank	1	2	3
Relevance	0	1	0

NDCG discontinuous w.r.t model parameters!

Introduction to Information Retrieval

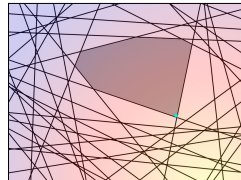
### Structural SVMs [Tsochantaridis et al., 2007]

- Structural SVMs are a generalization of SVMs where the output classification space is not binary or one of a set of classes, but some complex object (such as a sequence or a parse tree)
- Here, it is a complete (weak) ranking of documents for a query
- The Structural SVM attempts to predict the complete ranking for the input query and document set
- The **true labeling** is a ranking where the relevant documents are all ranked in the front, e.g.,
  - $y = \square \square \square \square \square \square \square \square$
- An **incorrect labeling** would be any other ranking, e.g.,
  - $y = \square \square \square \square \square \square \square \square$
- There are an **intractable number of rankings, thus an intractable number of constraints!**

Introduction to Information Retrieval

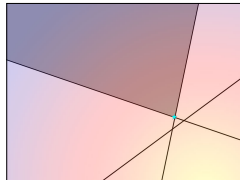
### Structural SVM training [Tsochantaridis et al., 2007]

Structural SVM training proceeds incrementally by starting with a working set of constraints, and adding in the most violated constraint at each iteration



**Original SVM Problem**

- Exponential constraints
- Most are dominated by a small set of "important" constraints



**Structural SVM Approach**

- Repeatedly finds the next most violated constraint...
- ...until a set of constraints which is a good approximation is found

Introduction to Information Retrieval

### Other machine learning methods for learning to rank

- Of course!
- I've only presented the use of SVMs for machine learned relevance, but other machine learning methods have also been used successfully
  - Boosting: RankBoost
  - Ordinal Regression loglinear models
  - Neural Nets: RankNet, RankBrain
  - (Gradient-boosted) Decision Trees

Introduction to Information Retrieval

### The Limitations of Machine Learning

- Everything that we have looked at (and most work in this area) produces *linear* models over features
- This contrasts with most of the clever ideas of traditional IR, which are *nonlinear* scalings and combinations (products, etc.) of basic measurements
  - log term frequency, idf, tf.idf, pivoted length normalization
- At present, ML is good at weighting features, but not as good at coming up with nonlinear scalings
  - Designing the basic features that give good signals for ranking remains the domain of human creativity
  - Or maybe we can do it with deep learning 😊



Introduction to Information Retrieval

The screenshot shows a Quora question page. The question is: "Why is machine learning used heavily for Google's ad ranking and less for their search ranking?". The question has 526 answers and 14 comments. The top answer is by Christopher Manning, who explains that ad ranking is machine learning based, while search ranking is rooted in functions written by humans using their intuition. The page also shows related questions and a sidebar with social media sharing options.

Introduction to Information Retrieval

## Summary

- The idea of learning ranking functions has been around for about 20 years
- But only more recently have ML knowledge, availability of training datasets, a rich space of features, and massive computation come together to make this a hot research area
- It's too early to give a definitive statement on what methods are best in this area ... it's still advancing rapidly
- But machine-learned ranking over many features now easily beats traditional hand-designed ranking functions in comparative evaluations [in part by using the hand-designed functions as features!]
- There is every reason to think that the importance of machine learning in IR will grow in the future.

Introduction to Information Retrieval

## Resources

- *IIR* secs 6.1.2–3 and 15.4
- LETOR benchmark datasets
  - Website with data, links to papers, benchmarks, etc.
    - <http://research.microsoft.com/users/LETOR/>
    - Everything you need to start research in this area!
- Nallapati, R. Discriminative models for information retrieval. *SIGIR 2004*.
- Cao, Y., Xu, J. Liu, T.-Y., Li, H., Huang, Y. and Hon, H.-W. Adapting Ranking SVM to Document Retrieval, *SIGIR 2006*.
- Y. Yue, T. Finley, F. Radlinski, T. Joachims. A Support Vector Method for Optimizing Average Precision. *SIGIR 2007*.