Structured Learning for Non-Smooth Ranking Losses

Soumen Chakrabarti Rajiv Khanna Uma Sawant Chiru Bhattacharyya

IIT Bombay and IISc Bangalore

Training setup

- A set of queries; each query q comes with a set of documents
- A document may be relevant or irrelevant wrt q
- Multilevel relevance also possible, not considered here
- n_q^+ relevant ("good") docs D_q^+; n_q^- irrelevant ("bad") docs D_q^-
- Each doc represented as a feature vector $x_{qi} \in \mathbb{R}^d$; $d \approx 50 \dots 300$
- Learner estimates *model* $w \in \mathbb{R}^d$

Application to test data

- Given a query and an unlabeled doc set
- *Score* of doc i is $w^\top x_{qi}$
- Sort docs by decreasing score, present top-k

Evaluation criteria

- Ideally, all docs in D_q^+ should be ranked above any doc in D_q^-
- Penalty for imperfect rankings can be defined in many ways
- Let g, b range over good, bad docs

Area under curve (AUC): Related to number of flipped good-bad pairs

$$1 - \Delta_{\text{AUC}} = \frac{1}{n^+ n^-} \sum_{g,b} [\![g \text{ is ranked above } b]\!]$$

Mean average precision (MAP):

$$1 - \Delta_{\text{MAP}} = \frac{\text{number of good docs up to } g}{\text{number of docs up to } g}$$

Mean reciprocal rank (MRR): Let r_1 be rank of first good doc; then

$$1 - \Delta_{\text{MRR}} = \begin{cases} 1/r_1, & r_1 \le k \\ 0, & \text{otherwise} \end{cases}$$

(no credit for 2nd and subsequent good docs)

Normalized discounted cumulative gain (NDCG): Discounted cumulative gain for q is $\text{DCG}(q) = \sum_{0 \le i < k} G(q,i) D(i)$

- $G(q,i)$ is the gain or relevance of document i for query q, $z_{qi} \in \{0,1\}$
- $D(i)$ is the discount factor:

$$D(i) = \begin{cases} 1 & 0 \le i \le 1 \\ 1/\log_2(1+i) & 2 \le i < k \\ 0 & k \le i \end{cases}$$

(decaying credit for good doc at lower ranks)

- The ideal ranking has $\text{DCG}^*(q) = \sum_{i=0}^{\min\{n_q^+, k\}-1} G(q,i) D(i)$
- Normalize DCG for imperfect ranking with DCG of perfect ranking:

$$\text{NDCG}(q) = \frac{\text{DCG}(q)}{\text{DCG}^*(q)} = \frac{\sum_{0 \le i < k} z_{qi} D(i)}{\text{DCG}^*(q)}$$

(In all cases, average over all queries)

Loss cannot be decomposed

- In standard binary classification, the loss is $\sum_i [\![y_i \ne y_i^*]\!]$, where y_i^* is the true label of instance i and y_i is the learner-assigned label
- Adds up over instances
- In contrast, if y_q^* is the perfect ranking and y another ranking, Δ is a function of y as a whole
- Can write as $\Delta(y_q^*, y) = \Delta_q(y)$, say
- If y represents a total order, then there are $(n^+ + n^-)!$ possibilities
- If y represents a relative order between good and bad docs, there are $2^{n^+ n^-}$ possibilities

Structured SVM

- Let $\phi(x_q, y) \in \mathbb{R}^d$ be a *feature map* over all D_q and a ranking y of docs in D_q
- Learn model vector $w \in \mathbb{R}^d$
- Score of ranking y is $w^\top \phi(x_q, y)$
- *Inference problem*: $\arg\max_y w^\top \phi(x_q, y)$
- Max-margin optimization:

$$\arg\min_{w; \xi \ge \vec{0}} \tfrac{1}{2} w^\top w + \tfrac{C}{|Q|} \sum_q \xi_q \quad \text{s.t.} \qquad (1)$$

$$\forall q, y \ne y_q^* : \; w^\top \phi(x_q, y_q^*) \ge w^\top \phi(x_q, y) + \Delta(y_q^*, y) - \xi_q.$$

(want y^* to beat all other ys)

- Avoid exponential number of primal constraints by solving approximate dual [3]
- To do this, must solve *loss-augmented inference* ("argmax") problem efficiently:

$$\arg\max_y w^\top \phi(x, y) + \Delta(y)$$

- Yue *et al.* [4] solved for MAP, we solve MRR and NDCG here

Feature map design

- *Vector* x_{qi} from domain knowledge
- But *map* ϕ is key to learning to rank
- $\phi_{\text{po}}(x, y) = \frac{1}{n^+ n^-} \sum_{g,b} y_{gb} (x_g - x_b)$ has been used For AUC and MAP
- Can show that

$$\phi_{\text{po}}(x, y^*) - \phi_{\text{po}}(x, y) = \psi(x, y^*) - \psi(x, y),$$
where $\psi(x, y) = 2 \frac{1}{n_q^+ n_q^-} \sum_g \sum_{b: b \succ g} (x_b - x_g)$

- Recasting helps us propose alternative feature map for MRR:

$$\phi_{\text{mrr}}(x, y) = \sum_{b: b \succ g_0(y)} (x_b - x_{g_0(y)})$$

(only top good doc, no scaling)

- ϕ should be matched to Δ

Argmax algo for MRR

- $\arg\max_y \Delta_q(y) + w^\top \phi(x_q, y)$
- $1, 1/2, 1/3, 1/k, 0$ only possible values of MRR
- For a given value of MRR, say $1/r$, first good doc must be at rank r
- Docs at rank $1, \dots, r-1$ must be bad
- Docs after rank r can be in any order
- For a given configuration $\underbrace{b, \dots, b}_{r-1}, \underbrace{g}_{r}, \underbrace{?, ?, \dots}_{\text{rest}}$ need to fill good and bad slots to maximize $w^\top \phi(x_q, y)$
- Bad docs b at $1, \dots, r-1$ with largest $w^\top x_b$
- Good doc g with smallest $w^\top x_g$ at position r
- MRR $= 0$ handled separately
- Add up Δ and $w^\top \phi$ for each possible Δ and take maximum

Argmax algo for NDCG

- Assume two relevance levels (good and bad)
- Δ_{NDCG} is unchanged if two good or two bad documents are interchanged
- Therefore the y that maximizes $\Delta_q(y) + w^\top \phi(x_q, y)$ has good (and bad) docs in decreasing score order
- Find optimal merge of good and bad lists each sorted by decreasing $w^\top x_{qi}$
- Dynamic programming and greedy solutions

SVMcombo

- Ultra-optimizing for one Δ not good for mixed workload
- Anyway targeted Δ not great in experiments
- Do the different Δ_ls conflict, or is it possible for a single w to do well for a number of them?
- Optimal for all Δ_l pushes all good to top, so we are hopeful

$$\arg\min_{w; \xi \ge \vec{0}} \tfrac{1}{2} w^\top w + \tfrac{1}{|Q|} \sum_q \sum_l C_l \xi_q^l \quad \text{s.t.}$$

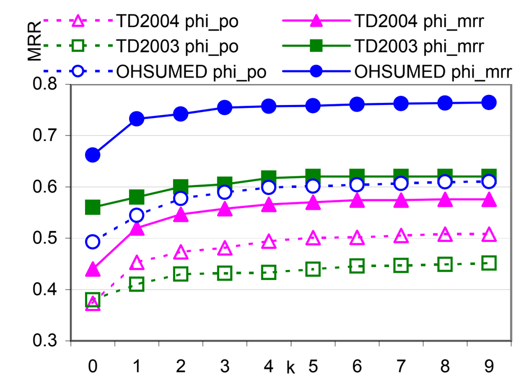
$$\forall l, q, \forall y \ne y_q^* : w^\top \delta\phi(y_q^*, y; x_q) \ge \Delta_l(y_q^*, y) - \xi_q^l$$

Other approaches

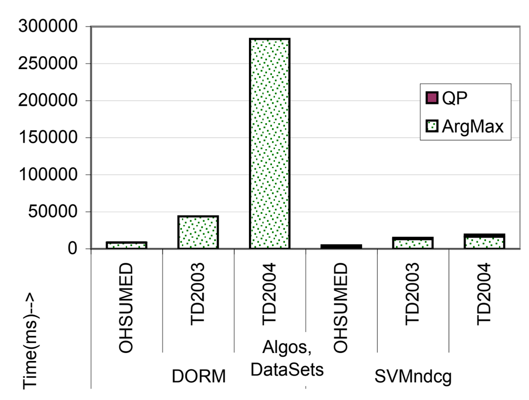
- SVMmap [4]: Directly optimize for Δ_{MAP}
- DORM [1]: structured SVM for Δ_{NDCG} based on Hungarian assignment of docs to ranks
- McRank [2]: boosted regression trees

Experiments and summary

- ϕ_{mrr} much better than ϕ_{po} for Δ_{MRR}



- SVMndcg is much faster than DORM



- SVMndcg, SVMmrr faster than McRank

Dataset	McRank tree	McRank boost	McRank total	SVMndcg	SVMmrr
OHSUMED	1034	67	1102	4.8	30.6
TD2003	9730	383	10113	14.9	125
TD2004	8760	548	9308	19.1	148

	OHSUMED			TD2003			TD2004			TREC2000			TREC2001		
	MRR10	NDCG10	MAP	MRR10	NDCG10	MAP	MRR10	NDCG10	MAP	MRR10	NDCG10	MAP	MRR10	NDCG10	MAP
AUC	.799	.635	.582	.510	.349	.256	.639	.501	.420	.607	.448	.267	.632	.441	.264
MAP	.808	**.642**	**.586**	.618	.411	*.314*	.614	.496	*.412*	**.696**	**.469**	**.277**	.636	**.450**	**.272**
NDCG	.790	*.636*	.581	.587	*.372*	.302	.631	*.457*	.374	.517	*.323*	.175	.608	*.356*	.171
NDCG-NC	**.818**	.640	.582	.595	.404	.306	.611	.486	.404	.685	.455	.265	.624	.443	.264
MRR	*.795*	.623	.570	*.628*	.405	.330	*.629*	.441	.383	*.670*	.410	.244	*.643*	.426	.230
COMBO	.813	.635	.578	**.667**	**.434**	**.345**	**.647**	.458	.384	.695	.465	**.277**	**.647**	.449	**.272**
DORM	.807	.637	.583	.587	.362	.290	.474	.340	.297	.662	.413	.243	.621	.435	.250
McRank	.701	.565	.527	.650	.403	.232	.588	**.529**	**.453**						

- "Using the correct training loss" may be worse than using an incorrect training loss!
- Multicriteria learning (SVMcombo) most robust and may be safest for search applications
- Structured listwise learning to rank is competitive with other approaches
- Feature map design needs more insight and improvement

References

[1] O. Chapelle, Q. Le, and A. Smola. Large margin optimization of ranking measures. In *NIPS 2007 Workshop on Machine Learning for Web Search*, 2007.

[2] P. Li, C. J. C. Burges, and Q. Wu. McRank: Learning to rank using multiple classification and gradient boosting. In *NIPS*, pages 845–852, 2007.

[3] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6(Sep):1453–1484, 2005.

[4] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *SIGIR Conference*, 2007.

KDD 2008, August 24–27, 2008, Las Vegas, Nevada, USA
Author contact: soumen@cse.iitb.ac.in
Supported by Microsoft External Research Initiative