Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
Università Ca'Foscari Venezia

LambdaRank Gradients are Incoherent

Federico Marcuzzi, Claudio Lucchese and Salvatore Orlando

iNEST Spoke 6 "Tourism, Culture and Creative Industries" RT1: Ricerca Industriale (RI). Sub. RT1.2: User-generated content, big data and machine learning for a smarter tourism industry.

Abstract
Information Retrieval systems are widely used in a vast number of scenarios, such as tourism, education, research, etc. In particular, for tourism applications, information systems such as web search engines and recommendation systems are used to allow a user (i.e., the tourist) to look for tourist destinations, hospitality facilities, and general territorial information. Nowadays, the core of information systems is machine learning, in particular, learning to rank in the case of web search engines. Learning to rank learns a ranker that, given a user's needs (i.e., the query), returns a list of results ordered by relevance to the query, with the most relevant results at the top of the returned list. One of the most effective and efficient approaches for learning to rank is the well-known LambdaMART algorithm. LambdaMART defines a gradient for each document w.r.t. the information retrieval metric to be optimised, the so-called lambdas. Intuitively, each lambda describes, with some degree of approximation, how much a document score should be pushed up or down to improve the ranking. In this work, we show that lambdas can be incoherent w.r.t. the metric optimised: a document with high relevance to the user query can receive a downward push larger than a document with lower relevance. In addition, optimising truncated metrics in order to speed up the training time can exacerbate this discrepancy and lead to worse model learning. This behaviour goes far beyond the expected degree of approximation. Moreover, we show how the practice of truncated metric optimisation introduces an unwanted unfair comparison between items (facilities, accommodation, etc.) equally relevant to the user query. We analyse the idiosyncrasies of LambdaMART gradients, and we introduce some strategies to remove the exacerbation of gradient incoherency and the unfair item comparison. We empirically demonstrate on publicly available datasets that the proposed approach leads to a fairer training process and to models that can achieve statistically significant improvements in terms of NDCG.

1 Introduction

Information Retrieval (IR) is the field of research focused on retrieving useful information from a huge collection of data. Several areas related to computer science have developed IR strategies.

Two of the most famous and widely used applications of IR strategies are Web Search Engines and Recommendation Systems. Nowadays, web search engines and recommendation systems are used in a large variety of scenarios, such as research, education, entertainment, tourism, etc. In the context of tourism, web search engines and recommendation systems are used to book accommodation, look for activities and cultural events in a specific tourist area, or find general territorial information for tourists.

Nowadays, these information retrieval systems make extensive use of machine learning and deep learning approaches to perform the aforementioned tasks. Among these approaches, there is learning to Rank (LTR), one of the most applied approaches in IR. LTR is a machine learning approach that includes supervised learning algorithms to construct rankers that satisfy user needs by sorting by relevance a huge set of items that mostly contain irrelevant items and only a small fraction of relevant ones. In other words, a ranker takes as input a user query (e.g., "accommodation in Venice") and returns a short sorted list of items relevant to the query (e.g., a list of accommodation in Venice). These items are extracted from a huge set of items (e.g., the World Wide Web) that are mostly non-relevant (e.g., on the entire World Wide Web there are many more accommodations outside Venice than in Venice).

In order to provide relevant results, LTR algorithms optimise information retrieval metrics. However, most of the information retrieval metrics are ranked-based (i.e., they rely on the order of the items), so they are discontinuous or flat everywhere. Consequently, gradient descent strategies to optimise such metrics can not be applied directly.

Despite the non-differentiability of IR metrics, many LTR algorithms are gradient-based, which either optimise an approximate version of the ranking metric or build gradients based on heuristics as in the case of LambdaMART algorithm defined by Burges (2010). LambdaMART optimises a non-differentiable loss function that it doesn't know by creating an ad-hoc gradient for each document based on heuristics that take into account the contribution of the document in the rank and the relationship it has with the other documents. Since LambdaMART is based on heuristics, its gradients are not exact. In this work, we want to show how LambdaMART and derivatives such as LambdaRank by Burges et al. (2006) and the metric-driven loss functions defined by Wang et al. (2018) have an inherent problem due to the heuristics used and can generate incoherencies in gradients.

Moreover, machine learning algorithms (and consequently learning to rank algorithms, too) can provide unfairness for certain groups of users, items, etc. The unfairness covers different scenarios, such as discrimination by gender and race, unfair exposure for the same item from different vendors in e-commerce, etc. Unfairness in machine learning can be caused by different aspects, such as bias in the training set that disadvantages one group over another (e.g., black people are discriminated more than non-back people), limited availability of resources (e.g., one position available but ten equally relevant candidates for the position), or the intrinsic design of the machine learning algorithm (e.g., not all the training instances are equally considered during the learning phase of the algorithm). We focus on the last aspect that generates unfairness in information retrieval systems.

In this work, we discovered three unexpected behaviours in the well-known LambdaRank algorithm and its derivatives, such as LambdaMART. *i)* we discovered that LambdaMART are affected by gradient incoherencies that compromise the model learning. A gradient incoherency is when an item with high relevance to the query receives a downward push larger than an item with lower relevance. *ii)* we discovered that the practice to optimise truncated information retrieval metrics (i.e., metrics optimised only in the top-k position (i.e., those of interest to the users) to reduce the training time (i.e., to improve the algorithm efficiency) exacerbate the phenomena of gradient incoherency that leads to worse model learning. *iii)* we discovered that truncated metric optimisation introduces an undesired unfair comparison of items/documents in the training set. In particular, we discovered that documents that are equally relevant to a query are treated

differently, to the detriment of equally relevant documents positioned lower down in the ranking and which would most need to be pushed upward.

The main contribution of this work is Lambda-*eX* an improvement of LambdaMART objective function, that *extends* the set of document pairs processed during training, to optimise truncated ranking metrics (i.e., a more efficient training) while avoiding unfair document comparison and the exacerbation in the number of gradients incoherencies. We empirically demonstrated through five publicly available datasets that Lambda-*eX* achieves statistically significant improvements in terms of NDCG than the baselines. Moreover, Lambda-*eX* removes the exacerbation of the gradient incoherencies introduced by truncated metric optimisation that compromise the exposure of documents with respect to equally relevant documents without gradient incoherencies.

2 Gradient Incoherency and Unfair Document Comparison

Gradient-based learning algorithms, such as artificial neural networks or gradient-boosted decision trees, run iterative updates to build a ranker that minimises a given cost function C. For instance, gradient-boosted decision trees iteratively learn a new tree that approximates $\partial C / \partial s_i$ for each document d_i in the training set D and its score s_i. Unfortunately, most IR metrics are rank-based: they depend on ranking π rather than on s_i. This makes the cost function either flat or non-differentiable. Note that π is the ranking over the documents $d_i \in D$ sorted in decreasing order of scores s_i predicted by the ranker, and $\pi[i]$ denotes the position of document d_i in the ranking.

LambdaRank's cost function defined by Burges et al. (2006) is one of the most relevant approaches used to tackle this problem, and it stems from the RankNet cost proposed by Burges et al. (2005), which is enhanced by considering the impact on the IR metric. The gradient is computed on the basis of pair-wise lambdas λ_{ij} as follows:

$$\lambda_i = \sum_{j:(i,j)\in I} \lambda_{ij} - \sum_{k:(k,i)\in I} \lambda_{ki} \tag{1}$$

where I is the set of ordered documents pairs (i, j) such that $y_i > y_j$, i.e., $I = \{(i, j) \mid d_i, d_j \in D \ \wedge \ y_i > y_j\}$. The value of λ_{ij} estimates the change on the cost function C when the distance between the two scores s_i and s_j is modified.

To manage user behaviour and increase training efficiency, real-world applications of information retrieval systems mostly try to optimise the effectiveness only for the first k results. IR metrics naturally provide a truncated version, i.e. NDCG@k is computed by considering only the contribution of the top-k ranked documents.

By training the model to optimise a truncated metric Z to a certain truncation level τ, pairs of documents ranked beyond τ are not considered since the corresponding contribution to the metric is equal to 0. Thus, in order to reduce the training time, the number of document pairs in $\mathcal{I}$ is limited while computing the gradients λ_i in Equation 1 by replacing the set I with $I_\tau = \{(i, j) \mid d_i, d_j \in D \wedge y_i > y_j \wedge \min(\pi[i], \pi[j]) \leq \tau\}$.

It is important to note that, although closely related, the truncation level τ is different from the metric cutoff k. The former affects the number of document pairs to process, and the latter affects the evaluation of the metric. Moreover, they may not be equal, i.e. τ may be slightly larger than k to process more pairs during the training phase.

Table 1 shows an example of LambdaMART gradients when maximising NDCG. The query has only three documents with their ranks $\pi[i]$ and scores s_i predicted by the model, and relevance label y_i. The top-ranked document with relevance equal to 4 and

Table 1
Detailed computation of LambdaMART gradients.

d_i	$\pi[i]$	y_i	s_i	λ_i	
d_1	1	4	0.02	$\lambda_1 = \lambda_{12} + \lambda_{13}$	$\approx 0.176 + 0.221 \approx 0.397$
d_2	2	0	0.01	$\lambda_2 = -\lambda_{12} - \lambda_{32}$	$\approx -0.176 - 0.004 \approx -0.180$
d_3	3	1	0.00	$\lambda_3 = -\lambda_{13} + \lambda_{32}$	$\approx -0.221 + 0.004 \approx -0.217$

is correctly pushed up by the gradient λ_1. Interestingly enough, the second and third documents are misranked with labels 0 and 1 respectively. The LambdaMART gradient is negative for both documents, but the document with the larger label is pushed down with greater strength. We may conclude that such gradients are not going to improve the ranking but rather increase the gap between the two misranked documents. We call this phenomenon *gradients incoherency*.

To explain in detail the reason for such behaviour, in Table 1 we report the computation of the document gradients λ_i as a function of the pair-wise λ_{ij} according to Equation 1 in case of the NDCG metric. Document d_1 has a positive gradient λ_1 as it is ranked higher than documents with smaller relevance labels. Document d_2 is the least relevant and receives a negative gradient contribution from both the other documents. Unexpectedly, document d_3 receives the strongest downward push even if it has a higher label than d_2. The reason is that swapping document d_1 with d_3 has a larger impact on the NDCG than swapping d_1 with d_2, resulting in $\lambda_{13} > \lambda_{12}$. LambdaMART prefers avoiding the risk of moving d_1 to the third position rather than pushing d_3 up to the second place. Indeed, this comes from the discount factor of NDCG metric that demotes documents' contributions in the lower ranks. These gradients clearly push the ranking away from the ideal configuration as we would prefer having λ_3 larger than λ_2.

The phenomenon of gradient incoherencies is significantly further exacerbated when optimising truncated metrics. This is because by removing (i, j) pairs from set I, the λ_i gradients of the relevant documents under τ will be incomplete since they will lose some λ_{ij}. As a result, relevant documents below τ will receive even less upward push.

Moreover, the substitution of the set I with I_τ introduces an unfair optimisation process among equally relevant documents. In fact, while optimising truncated metric, two equally relevant documents can receive a different number of λ_{ij} contributions, not experienced during un-truncated metric optimisation.

In Figure 1, it is possible to see a clear example of unfair document comparison introduced by truncated metric optimisation. In the example, a truncated metric optimisation is forced by implying $\tau = 1$. When truncated metric optimisation is used the document in position 1 with relevance 2 receives more contribution from the other documents than the document in position 2 while being equally relevant. In fact, the document in position 2 receives contribution $\lambda_i j = 0$ from all documents below the truncation level, while the document in position 1 receives contribution $\lambda_i j \geq 0$.

3 Lambda-e*X*

The main contribution of this work is Lambda-e*X*, a learning algorithm able to cancel the gradient incoherency exacerbation and avoid the introduction of unfair document comparisons. We claim that the exacerbation of the incoherencies and the unfair comparisons are due to missing computations of the λ_{ij} gradients. More specifically, relevant documents that are not ranked above the truncation level are not evaluated against all

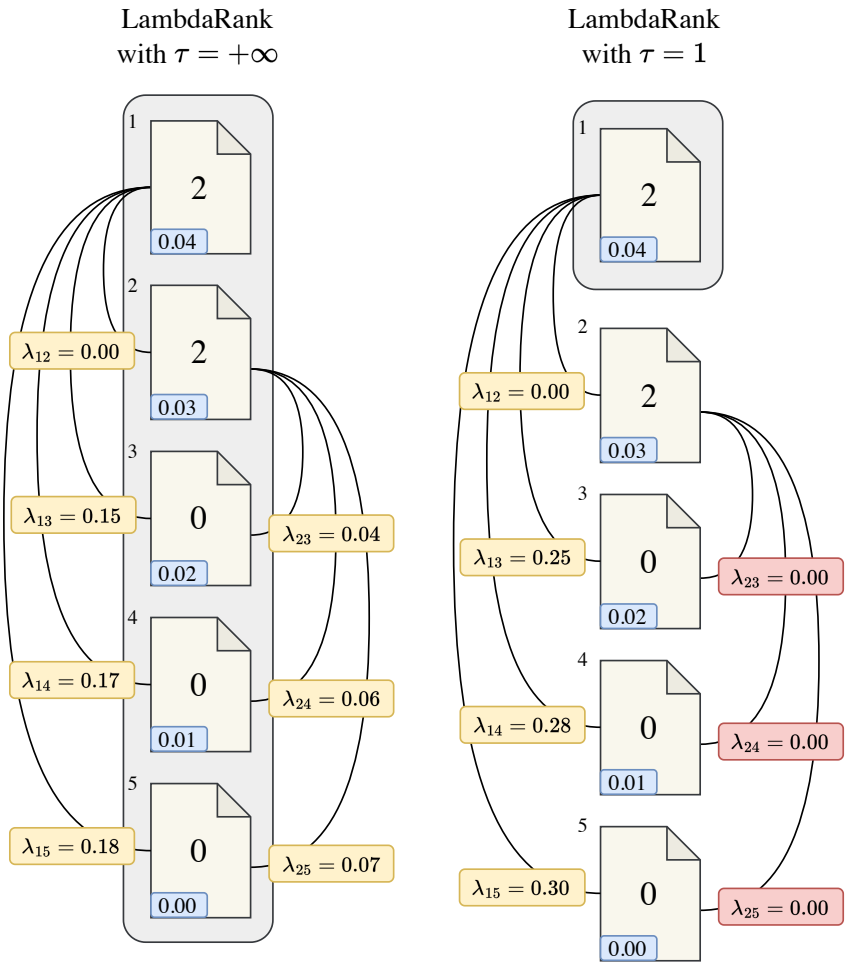

Figure 1: Examples of unfair document comparison. When truncated metric optimisation is used (e.g., $\tau = 1$), the document in position 1 with relevance 2 receives more contribution from the other documents than the document in position 2 while being equally relevant.

the candidate documents of the query but only against the top-k, and this ignores some of the λ_{ij} and under-estimates their gradient.

To keep the training focused on user behaviour and tackle the problem of gradient incoherencies and unfair document comparison, we designed Lambda-*eX*, which *extends* the set of document pairs considered by LambdaMART when computing gradients. To achieve this goal, we define a set of documents $X \subseteq D$ so as to include in X all the documents for which we want a *complete* gradient estimation, i.e., those that deserve or not to be ranked at the top positions. Moreover, the set $\mathcal{X}$ contains the documents for which we want a fair comparison. As mentioned, X is a subset of D, so not all the documents in D are provided with a fair comparison. However, due to a limited number of positions effectively viewed by the users, i.e., the top-k positions, we do not require that all documents have a fair comparison, but just those that should be ranked in the top-k position to maximise the user need, i.e., the relevance to the user query.

Lambda-*eX* thus computes each λ_i gradient as in Equation 1 but basing on the set:

$$I_X = \{(i,j) \mid d_i, d_j \in D \wedge y_i > y_j \wedge (d_i \in X \vee d_j \in X)\}$$

How does Lambda-*eX* populate the set X? Let k be the cutoff of the IR metric being optimised. Lambda-*eX* includes in X all the documents ranked in the top-k positions

by the current model and the contender documents below the cutoff k. The contender documents are those not placed in the top-k positions but required to maximise the ranking metric. Since the number of contender documents could be large, to limit the size of X to about k, Lambda-*eX* uses some criteria to select the contender documents to include in X. We propose three different ways to select the contender documents.

- `static`: let h be the number of documents that are required to maximise the metric Z, but the model did not rank among the top-k positions. This strategy adds in X the h most relevant documents not yet in the top-k. This strategy provides a partial fair document compassion since it compares the minimal number of documents that deserve to be in the top-k in order minimise the training time, i.e., the training efficiency.
- `random`: analogous to `static`, but ties are broken randomly instead of rank-based. A random selection allows the model to be fairer since it compares all contenders during training and improves model generalisation.
- `all`: analogous to `static`, but ties are not broken. If there are more than h documents with the desired relevant labels, they are all included in X. This enhances generalisation and fairness at the expense of efficiency. With the `all` strategy, all documents that should be ranked in the top-k are compared with the others, so providing the fairest strategy among the three.

We also propose two hybrid variants: `all-static` and `all-random`. The goal is to limit the size of X. Depending on the query, the `all` may potentially include all the relevant documents. To avoid such blow-up of X, the hybrid strategies roll back to either `static` or `random` in this degenerate case; otherwise, they implement the `all` strategy.

To evaluate the effectiveness of the proposed approach, we defined two baselines: LambdaMART that optimise the truncated version of NDCG metric with truncation level τ equal to the metric cutoff k and the other with $\tau = k + 3$ as suggested in Corporation (2023). We compared the baselines with each version of Lambda-*eX* defined in Section 3. In Table 2, we summarise the performance for each model. The results show how Lambda-*eX* obtains statistical significance improvements by overcoming the exacerbation in gradients incoherencies introduced by LambdaMART when optimising a truncated metric.

4 Conclusion

We have shown that LambdaMART is affected by incoherencies in the computation of the gradients, exacerbated by the truncation level. We showed that the training efficiency introduced by truncated metric optimisation comes at the expanses of fairness between equally relevant documents. We designed a new approach called Lambda-*eX* to counter these issues while keeping the focus on the metric to be optimised and the training efficiency. Through extensive experiments, we have shown that Lambda-*eX* is able to achieve a statistically significant improvement with respect to the baselines.

Table 2
Statistically significant improvement w.r.t. LambdaMART$_{\tau=k+3}$ according to Fisher's randomisation test defined by Fisher (1935) (with a two-sided p-value) are marked with bold ($p = 0.05$) and bold-italic ($p = 0.01$).

dataset	NDCG@k k	LambdaMART $\tau=k$	LambdaMART $\tau=k+3$	Lambda-*eX* static	random	all	all-static	all-random
Istella-X	5	73.32	75.35**	75.19**	75.17**	75.15**	75.19**	75.17**
Istella-S	5	70.19	70.64*	70.67**	70.71**	70.55	70.65*	70.64*
Istella-F	5	67.02	67.62**	67.55**	67.67**	67.50**	67.68**	67.71**
Yahoo! Set 1	5	75.35	75.85**	75.67	75.59	75.63	75.73	75.67
MSLR-30K	5	50.66	51.22	50.95	50.96	51.24	51.42*	51.38
Istella-X	10	77.53	78.61	78.61	78.61	78.61	78.61	78.61
Istella-S	10	76.35	76.71**	76.66**	76.70**	76.69**	76.72**	76.70**
Istella-F	10	71.85	72.39**	72.42**	72.46**	72.42**	72.35**	72.46**
Yahoo! Set 1	10	79.62	79.84	79.66	79.75	79.78	79.81	79.80
MSLR-30K	10	52.66	52.98	52.96	53.08	53.23*	53.19	53.14
Istella-X	15	79.00	79.45	79.44	79.48	79.44	79.44	79.48
Istella-S	15	80.63	80.73*	80.69	80.71*	80.75**	80.80**	80.73*
Istella-F	15	75.46	75.87**	75.94**	75.90**	75.92**	76.00**	76.00**
Yahoo! Set 1	15	82.01	82.03	81.94	82.07	82.04	82.07	82.04
MSLR-30K	15	54.60	54.67	54.82	54.93**	54.84	54.75	54.83

References

Burges, C. J. C. (2010), 'From ranknet to lambdarank to lambdamart: An overview'.

Burges, C. J. C., Ragno, R. & Le, Q. V. (2006), Learning to rank with nonsmooth cost functions, *in* B. Schölkopf, J. C. Platt & T. Hofmann, eds, 'Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006', MIT Press, pp. 193–200.

Burges, C. J. C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N. & Hullender, G. N. (2005), Learning to rank using gradient descent, *in* L. D. Raedt & S. Wrobel, eds, 'Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005', Vol. 119 of *ACM International Conference Proceeding Series*, ACM, pp. 89–96.

Corporation, M. (2023), *LightGBM Release 3.3.3.99*.

Fisher, R. (1935), *The design of experiments. 1935*, Oliver and Boyd, Edinburgh.

Wang, X., Li, C., Golbandi, N., Bendersky, M. & Najork, M. (2018), The lambdaloss framework for ranking metric optimization, *in* A. Cuzzocrea, J. Allan, N. W. Paton, D. Srivastava, R. Agrawal, A. Z. Broder, M. J. Zaki, K. S. Candan, A. Labrinidis, A. Schuster & H. Wang, eds, 'Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018', ACM, pp. 1313–1322.