On the Local Optimality of LambdaRank

Pinar Donmez
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
pinard@cs.cmu.edu

Krysta M. Svore
Microsoft Research
One Microsoft Way
Redmond, WA 98052
ksvore@microsoft.com

Christopher J. C. Burges
Microsoft Research
One Microsoft Way
Redmond, WA 98052
cburges@microsoft.com

ABSTRACT

A machine learning approach to learning to rank trains a model to optimize a target evaluation measure with repect to training data. Currently, existing information retrieval measures are impossible to optimize directly except for models with a very small number of parameters. The IR community thus faces a major challenge: how to optimize IR measures of interest directly. In this paper, we present a solution. Specifically, we show that LambdaRank [1], which smoothly approximates the *gradient* of the target measure, can be adapted to work with four popular IR target evaluation measures using the same underlying gradient construction. It is likely, therefore, that this construction is extendable to other evaluation measures. We empirically show that LambdaRank finds a locally optimal solution for mean NDCG@10, mean NDCG, MAP and MRR with a 99% confidence rate. We also show that the amount of effective training data varies with IR measure and that with a sufficiently large training set size, matching the training optimization measure to the target evaluation measure yields the best accuracy.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation, Theory

Keywords

Learning to Rank, Web Search

1. INTRODUCTION

Learning to rank is an increasingly popular area of research. Ranking is a mapping from a set of items to an ordered list; the ranking of Web search results is a common example. This task consists of a set of queries and a set of retrieved documents for each query. The document-query pairs are labeled according to a scale from not relevant to highly relevant, and the ranking systems use the training data to compute a model that outputs a rank order based on a real-valued scoring function $f(x)$. In Web search ranking, the cost function is typically defined with respect to a sorted order of documents at the query level, and averaged over a large number of queries.

The ranking problem generally employs a cost function (target evaluation measure) that is not necessarily the one used to train the system. Typical target measures used in IR (see [8] for a detailed list) depend only on the sorted list and the relevance levels of the listed items. These measures are generally either flat everywhere or non-differentiable with respect to the model parameters; hence they are difficult to optimize directly. One way to address this issue is to find a close smooth approximation to the target measure, and optimize it via gradient descent. However, this is quite challenging due to the sort component of the target ranking measures. LambdaRank [1] tackles the problem by defining a smooth approximation to the *gradient* of the target cost instead of searching for a smooth approximation to the target cost itself. The basic idea of LambdaRank is to specify rules determining how the rank order of documents should change. These rules are incorporated into a λ-gradient that defines the gradient of an implicit cost function only at the points of interest [1]. LambdaRank was originally proposed for NDCG (Normalized Discounted Cumulative Gain), but the method is general and works with any target cost function. Recently, LambdaRank was shown to satisfy a necessary, but not sufficient condition for local optimality, namely that the gradient vanishes at the learned weights [12].

In this paper, we define λ-gradients for four widely used IR measures, namely mean NDCG@10, mean NDCG, MAP and MRR, using the same underlying construction used for the NDCG λ-gradient in [1]. This construction is likely extendable to other IR measures as well. We empirically show, with a confidence bound, the local optimality of LambdaRank on these measures by monitoring the change in training accuracy as we vary the learned weights of the net. We change the weights by projecting in a random direction on a unit ball and moving the weights in that direction. If the accuracy decreases as the original net weights change, it means the learned weights are at a local optimum. By checking the accuracy decreases for several hundred random directions, we show, with 99% confidence, that the learned net weights are at a local optimum, using a Monte-Carlo test with one-sided error. We also find that the gradient vanishes at each learned weight by fixing all but one weight's value and varying that weight's value while checking for a decrease in accuracy on the training set. If the highest accuracy is achieved at the learned weight value, then the gradient has vanished. Our work is not only the first to show empirical optimality of a learning algorithm, but also the first to show optimality across several IR measures. In addition, it shows IR practitioners can now directly optimize for the IR measure they care about, and the model need not be limited to only a few parameters. We also show that with large enough amounts of training data, the best test accuracy is achieved when matching the training optimization measure to the target evaluation measure.

2. RELATED WORK

The ranking task has become increasingly popular among researchers in the past few years. Some approaches rely on structured output prediction such as the large margin methods of [9, 10]. The learned structures are mapped to the reals, and then the best structure is chosen to give the highest real-valued score among all possible outputs. Another line of work casts the ranking problem as ordinal regression, that is, learning the mapping of an input vector to a member of an ordered set of numerical ranks [5]. Like many other ranking algorithms, their cost functions depend on pairs of examples. Crammer and Singer [4] proposed a similar solution where the ranker is a perceptron whose output is a weight vector $\boldsymbol{w}$. Cao et al. [3] proposed a listwise approach to learning to rank where a cross-entropy loss is defined between two parametrized probability distributions of permutations. Qin et al. [7] proposed a method called RankCosine, which depends on a listwise loss function that takes the cosine similarity between the score vectors of the predicted result and the ground truth. In addition, there are methods that claim to directly optimize the evaluation measures, such as SVMMAP [13] and AdaRank [11]. SVMMAP incorporates MAP into the listwise optimization constraints which are exponentially large. SVMMAP tackles this problem by performing optimization only on a working set of constraints which is extended with the most violated constraint at each step. The resulting algorithm works in polynomial time. AdaRank, on the other hand, performs a boosting-type optimization where the IR measure is embedded into the loss function used in updating the distribution of the data. The authors claim a theoretical guarantee that the training error defined in terms of the IR measure will reduce constantly with some mild assumptions. More recently, a method to directly optimize IR measures using an approximation of the positions of documents was proposed in [6].

Another approach is to train on pairs of documents per query. RankNet [2] is a neural-net-based ranking algorithm that optimizes a cross-entropy cost function using gradient descent. It is trained on pairs of documents per query, where documents in a pair have different labels. The RankNet cost consists of a sigmoid followed by a pair-based cross-entropy cost. The training time of RankNet scales quadratically with the average number of pairs per query, and linearly with the number of queries. Thus, speeding up RankNet training becomes crucial especially for large training sets. LambdaRank [1] provides a significant training speed-up as well as a framework for optimizing a cost function while avoiding the difficulties of working with non-differentiable IR measures. In addition, the NDCG gradient at the weights learned by LambdaRank has been shown to vanish, a necessary but not sufficient condition for empirical optimality of LambdaRank on NDCG [12]. We empirically show not only that it optimizes NDCG, but also MAP and MRR.

3. IR MEASURES

IR measures are typically defined with respect to a permutation of documents for a given query. The relevance labels can be binary or multilevel. For binary measures, we assume labels $\{0, 1\}$ (1 for relevant, and 0 for non-relevant). Binary measures include Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and Winner Takes All (WTA) (see [8] for a more complete list). In this paper, we focus on four of the most commonly used IR metrics: MAP, MRR, mean NDCG, and mean NDCG@10.

Average Precision (AP) computes for each relevant document the precision at its position in the ranked list; these precisions are then averaged over all relevant documents for query i:

$$\mathrm{AP@}L_i = \frac{\sum_{r=1}^{L} l(r)\mathrm{P@}r}{R} \tag{1}$$

where r is the rank position, L is the truncation level, R is the number of relevant documents, $l(r)$ is the binary relevance label of the document at rank position r, and $\mathrm{P@}r$ is the precision up to rank position r, i.e. $\mathrm{P@}r = \frac{\sum_{i=1}^{r} l(i)}{r}$. Mean Average Precision is the average of the average precisions over all N queries, $\mathrm{MAP@L} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{AP@}L_i$. In our work, we report on $\mathrm{MAP@}\infty$ and denote it by MAP.

Reciprocal Rank (RR) for a given query is the reciprocal of the rank position of the highest ranking relevant document for the query. MRR is just the average of the reciprocal ranks over queries:

$$\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{r_i} \tag{2}$$

where N is the number of queries and r_i is the highest position of a relevant document for query i.

Unlike binary measures such as MAP and MRR, NDCG recognizes multilevel relevance labels. NDCG for a given query i is formulated as follows:

$$\mathrm{NDCG@}L_i = \frac{1}{Z}\sum_{r=1}^{L}\frac{2^{l(r)} - 1}{\log(1 + r)} \tag{3}$$

where $l(r) \in \{0, \ldots, 4\}$ is the relevance label of the document at rank position r and L is the truncation level to which NDCG is computed. Z is chosen such that the perfect ranking would result in $\mathrm{NDCG@}L_i = 1$. Mean $\mathrm{NDCG@}L$ is the normalized sum over all queries: $\frac{1}{N}\sum_{i=1}^{N} \mathrm{NDCG@}L_i$. NDCG is particularly suited for Web search applications since it accounts for multilevel relevance labels and the truncation level can be set to model user behavior. In our studies, we consider mean NDCG@10 and mean $\mathrm{NDCG@}\infty$. We denote mean $\mathrm{NDCG@}\infty$ as simply mean NDCG.

4. LAMBDARANK

In most machine learning tasks, a target cost is used to assess the accuracy of the system at test time, and an optimization cost, generally a smooth approximation to the target cost, is used to train the system. Ideally, the optimization cost matches the target cost, but typical IR target costs (e.g. MAP, MRR, mean NDCG, etc.) are either flat or non-differentiable everywhere. Hence, direct optimization of the target cost is quite challenging. LambdaRank [1] solves this problem by defining the gradients of a given cost function only at the points of interest. The gradients are defined by specifying rules about how swapping two documents, after sorting them by score for a given query, changes the cost. Although LambdaRank gradients were originally defined for mean NDCG, the gradient definition is general and can work with any target cost function. In this section, we define λ-gradients for four different IR measures.

4.1 λ-Gradient for Mean NDCG

A LambdaRank gradient, λ_j, is defined to be a smooth approximation to the gradient of a target cost with respect to the score of the document at rank position j. λ-gradients have a physical interpretation; documents are represented by point masses and λ-gradients are forces on those point masses [1]. On two documents in a pair in a query, the λ-gradients are equal and opposite, where a positive λ-gradient indicates a push toward the top of the list, and a negative λ-gradient indicates a push toward the bottom of the list. With a choice of suitable λ-gradient, the gradient of any target cost can be smoothly approximated for a given document.

The authors of [1] tried several alternatives for λ-gradients and chose the best according to accuracy on validation data (for a detailed list, the reader is referred to [1]). The best λ-gradient found in [1] is a combination of the derivative of the RankNet cost [2] scaled by the NDCG@L_q gain from swapping two documents i and j with differing labels for a given query q. We drop q below for brevity. The RankNet cost is a pairwise cross-entropy cost applied to the logistic of the difference of the model scores. Assume document i has score s_i and label l_i, document j has score s_j and label l_j, and $o_{ij} \equiv s_i - s_j$ is the score difference, then the RankNet cost can be written as follows:

$$C_{ij} \equiv C(o_{ij}) = -S_{ij}o_{ij} + \log(1 + e^{S_{ij}o_{ij}}) \tag{4}$$

where

$$S_{ij} = \begin{cases} +1 & \text{if } l_i > l_j \\ -1 & \text{if } l_i < l_j \end{cases} \tag{5}$$

The derivative of the RankNet cost according to score difference is

$$\delta C_{ij}/\delta o_{ij} = \delta C_{ij}/\delta s_i = -S_{ij}/(1 + e^{S_{ij}o_{ij}}) \tag{6}$$

The λ-gradient can now be written as follows:

$$\lambda_{ij} \equiv S_{ij}\left|\Delta\text{NDCG}\frac{\delta C_{ij}}{\delta o_{ij}}\right| \tag{7}$$

$$= S_{ij}\left|N(2^{l_i} - 2^{l_j})\left(\frac{1}{\log(1+r_i)} - \frac{1}{\log(1+r_j)}\right)\left(\frac{1}{1+e^{S_{ij}o_{ij}}}\right)\right|$$

where N is the reciprocal of the maximum DCG for the query and r_i and r_j are the rank positions of documents i and j, respectively. Note that the sign S_{ij} only depends on the labels of documents i and j and not on their rank positions. In addition, if $l_i > l_j$, then document i is more relevant than document j and document i must move up the ranked list to reduce the cost, so $S_{ij} = 1$ and the λ-gradient for document i is positive. The λ-gradient for a single document is computed by marginalizing over the pairwise λ-gradients, $\lambda_i = \sum_{j \in P} \lambda_{ij}$, where the sum is over all pairs P for query q which contain document i. In each case below, λ_i is the sum of the pairwise λ-gradients.

4.2 λ-Gradient for MAP

LambdaRank is designed to work with any target cost function, as long as the λ-gradient can be defined. We design a λ-gradient for MAP based on the same principles as the one designed for mean NDCG, with the exception that ΔNDCG@L is substituted with ΔAP@L.

The λ-gradient for MAP uses the RankNet cost, scaled by the AP (Average Precision) gain found by swapping two documents i and j at rank positions r_i and r_j. We assume $L = \infty$, and drop it from equations for brevity. Assume documents i and j are misranked by the current net, i.e., $r_i > r_j$ but $l_i > l_j$[1], then $S_{ij} = 1$ and we can drop it for brevity. We have

$$\lambda_{ij} = \left|\frac{1}{R}\left(\sum_{k=r_j}^{r_i} l(k)\text{P@}k - \sum_{k=r_j}^{r_i} l'(k)\text{P}'\text{@}k\right)\left(\frac{1}{1+e^{o_{ij}}}\right)\right| \tag{8}$$

where $l(k) = 1$ if the document at rank position k is relevant, and 0 otherwise; P@k is the precision at rank k; R is the number of relevant documents for that query. $l'(k)$ is the relevance value after we swap the documents at positions r_i and r_j. In fact, $l'(k) = l(k)$ for all $k \in \{r_j + 1, ..., r_i - 1\}$, $l'(r_i) = l(r_j)$, and $l'(r_j) = l(r_i)$. P$'$@k is the precision at the rank positions between r_j and r_i after the swap. We can rewrite the above formula as:

$$\lambda_{ij} = \left|\frac{1}{R}\left[\left(\frac{n+1}{r_j} - \frac{m}{r_i}\right) + \sum_{k=r_j+1}^{r_i-1}\frac{l(k)}{k}\right]\left(\frac{1}{1+e^{o_{ij}}}\right)\right| \tag{9}$$

where n and m ($n \leq m$) are the number of relevant documents at the top r_j and the top r_i positions, respectively.

4.3 λ-Gradient for MRR

The λ-gradient for MRR uses the RankNet cost scaled by the gain in Reciprocal Rank (RR) for query q, found by swapping documents i and j at the corresponding rank positions r_i and r_j, for any $\{i, j\}$. Assume document i is relevant and document j is non-relevant, thus $l_i > l_j$ and $S_{ij} = 1$, then

$$\lambda_{ij} = \left|\Delta RR(r_i, r_j)\left(\frac{1}{1+e^{o_{ij}}}\right)\right| \tag{10}$$

ΔRR(r_i, r_j) calculates the difference in the reciprocal rank of the top relevant document as a result of the swap:

$$\Delta\text{RR}(r_i, r_j) = \begin{cases} \frac{1}{r_j} - \frac{1}{r} & \text{if } r_j < r \leq r_i \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where r is the rank of the top relevant document in the ordered list. Clearly, there is no RR gain (or loss) unless the rank of the top relevant document shifts after the swap.

5. LOCAL OPTIMALITY TESTING

We verify local optimality by showing that the training accuracy of LambdaRank decreases as the weights' learned

[1]Throughout, we assume higher (better) rank means lower rank index.

values are changed (either increased or decreased in value)[2]. We project iid random directions $\vec{r}_1, \vec{r}_2, ..., \vec{r}_k$ on a unit ball by first sampling each dimension of a vector from a Gaussian distribution[3] with 0 mean and unit variance and then projecting the vector onto the unit sphere. We modify the net weights in each direction as follows: let $\vec{w}$ be the vector of net weights, and $A_M(\vec{w})$ be the accuracy of the net with weights w with respect to a given evaluation measure M. We use a Monte-Carlo test with one-sided error. Assume a Bernoulli random variable Z that takes value 1 with probability p_Z and 0 with probability $1 - p_Z$. p_Z is the probability that the accuracy increases for some random direction r_i; i.e., $A_M(\vec{w}) \leq A_M(\vec{w} + \eta\vec{r}_i)$ for small $\eta > 0$. Given Z, we define a geometric random variable X with parameter p_Z to be the number of independent trials required, i.e., the number of random directions tested, before $Z = 1$. Requiring $(1 - \delta)\%$ confidence yields:

$$\begin{aligned} \mathbf{Pr}(X \leq N) = 1 - (1 - p_Z)^N &= 1 - \delta \\ \Rightarrow N &= \lceil \ln(\delta) / \ln(1 - p_Z) \rceil \end{aligned}$$

where N denotes the number of directions to test to find an increasing direction with probability $1 - \delta$. Since p_Z is unknown in our case, we assume $p_Z \geq \epsilon$ for small ϵ. Assuming $p_Z \geq \epsilon$, we obtain the following relation:

$$p_Z \geq \epsilon \Rightarrow \mathbf{Pr}(X \leq K) \geq 1 - \delta \text{ where } K = \lceil \ln(\delta) / \ln(1 - \epsilon) \rceil$$

which gives the minimum number of random directions K to be tested in order to find an increasing direction with $(1 - \delta)\%$ confidence. Let us define the null hypothesis H_0 as the assumption that $p_Z \geq \epsilon$, then we obtain $\mathbf{Pr}(X \leq K \mid H_0) \geq 1 - \delta$. If for K random directions no increasing direction is found, we reject H_0. H_0 may still be true even if we did not encounter an increasing direction, but this probability is upper bounded by δ. Requiring that $\epsilon = 0.01$ and $\delta = 0.01$, we obtain $K = \ln(0.01) / \ln(1 - 0.01) = 459$.

If we can show that varying the net weights in all 459 random directions results in worse training accuracy than the learned set of weights, then the learned set of weights represents a local optimum with 99% confidence. We analyze the change in training accuracy at 10 different step sizes $\eta \in \{0.1, 0.2, ..., 1\}$ as we change the net weights $\vec{w}$ in a given random direction r_i. Our empirical analysis (see Section 7) shows that there is a smooth decrease in the target costs as the learned weights change; hence, the choice of the step size does not seem to be restrictive [4].

6. DATASETS

We conducted experiments on three different datasets; two are real Web datasets from a commercial search engine and one is an artificial dataset [2] created to remove any variance caused by the quality of features and/or relevance labels. The artificial data was generated as decribed in [2] from random cubic polynomials. It has 300 features, 50 URLs per query, and a random 10K/5K/10K train/valid/test split on queries. We refer to it as the Artificial Data. The first Web search dataset has 158.7 URLs per query, 420 features, and 10K/5K/10K query train/valid/test splits. We call this dataset the 10K Web Data. The second Web dataset contains 22K/2K/10K query train/valid/test splits, 158.6 URLs per query, and 420 features. We refer to this dataset as the 22K Web Data.

All the document-query pairs are assigned integer labels between 0 (the least relevant) and 4 (the most relevant). For the binary measures MAP and MRR, we transform the multilevel relevance labels to binary by converting all labels between 2 and 4 (inclusive) to relevant (1), and all the rest to non-relevant (0).

[2]A direct analysis of the local optimality, involving the gradient or the hessian, for the IR target measures is problematic due to non-differentiability of the loss function.

[3]Any spherically symmetric distribution could be used.

[4]We found consistent results with $\eta \in \{0.001, 0.002, ...\}$, but omit results due to space limitations.

Table 1: MAP scores with standard error (SE) of different λ-gradients on the validation set.

λ-gradient	MAP $\pm$ SE
RankNetWeightPairs	0.462±0.0048
LocalGradient	0.435±0.0048
LocalCost	0.427±0.0049
SpringSmooth	0.424±0.0048
DiscreteBounded	0.401±0.0049

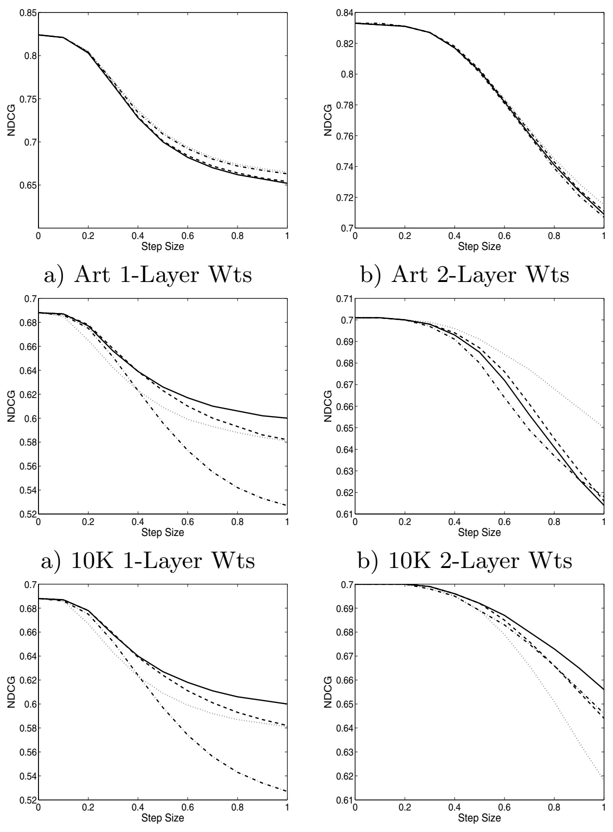
Table 2: MRR scores with standard error (SE) of different λ-gradients on the validation set.

λ-gradient	MRR $\pm$ SE
RankNetWeightPairs	0.524±0.0059
LocalCost	0.515±0.0060
LocalGradient	0.512±0.0059
SpringSmooth	0.498±0.0058
DiscreteBounded	0.471±0.0059

7. EXPERIMENTS

Through our experiments we seek to (1) show the local optimality of LambdaRank on three datasets for four IR measures and (2) determine if matching the training measure to the target evaluation measure yields the best test accuracy, and if so, how much training data is required. We begin by determining the best λ-gradient construction.

Following [1], we define 13 λ-gradient constructions for each IR measure and then train each construction for each measure on a 5K query Web set. We choose the construction with the best validation accuracy on a 5K query validation set. Tables 1 and 2 report the validation accuracy of a selective subset of λ-gradients for MAP and MRR, respectively. 'RankNetWeightPairs' is the construction detailed in Section 4. 'SpringSmooth' is a smoothed version of 'RankNetWeightPairs' where the gain obtained by swapping a pair is lower-bounded by 1. 'LocalGradient' estimates the gradient by the change in accuracy with respect to the difference in scores between two adjacent documents in an ordered list. A margin is added to handle very small score differences. 'LocalCost' uses a cost based on a document's neighbors to compute an estimate of the local gradient. 'DiscreteBounded' computes the change in accuracy when a document is moved to its ideal position in the ranked order, and the λ-gradient is upper-bounded by 1. For all four IR measures, 'RankNetWeightPairs' outperforms the other constructions with statistical significance on the validation set. We choose this construction for our λ-gradients. It is likely this construction can be extended to other IR measures as well.

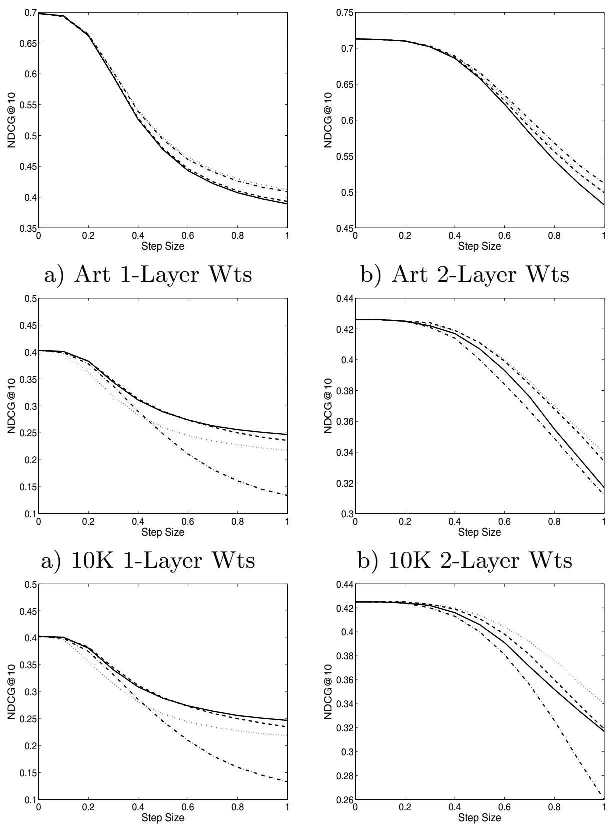


a) Art 1-Layer Wts b) Art 2-Layer Wts

a) 10K 1-Layer Wts b) 10K 2-Layer Wts

a) 22K 1-Layer Wts b) 22K 2-Layer Wts

Figure 1: Shifting single- and two-layer weights for mean NDCG. x-axis is the step size, η.



a) Art 1-Layer Wts b) Art 2-Layer Wts

a) 10K 1-Layer Wts b) 10K 2-Layer Wts

a) 22K 1-Layer Wts b) 22K 2-Layer Wts

Figure 2: Shifting single- and two-layer weights for mean NDCG@10. x-axis is the step size, η.

For all experiments, we varied the learning rate between 10^{-7} to 10^{-3}, and picked the rate that gave the best validation accuracy. Each algorithm was run for 700 epochs with a random restart if validation accuracy was constant over 50 epochs. If the training accuracy decreased at a given epoch, the learning rate was reduced by a factor of 0.8 with 30% probability. The output of each algorithm is a set of learned model weights. We report results in this paper based on single-layer nets and two-layer nets with 10 hidden nodes. We denote the training of LambdaRank for the different measures by *LambdaRankNDCG, LambdaRankNDCG@10, LambdaRankMAP,* and *LambdaRankMRR.*

7.1 Empirical Optimality of LambdaRank

We determine, for each training measure, if the learned weights represent a local optimum by examining the training accuracy at the learned weights as well as the weights shifted in 459 random directions.We vary all weights together in both single-layer and 2-layer nets. In all figures, for readability, we graph only 4 random directions, but verify that all remaining directions cause a decrease in training accuracy. When the step size $\eta = 0$, the accuracy corresponds to the training accuracy of the original (learned) net. We begin with mean NDCG.

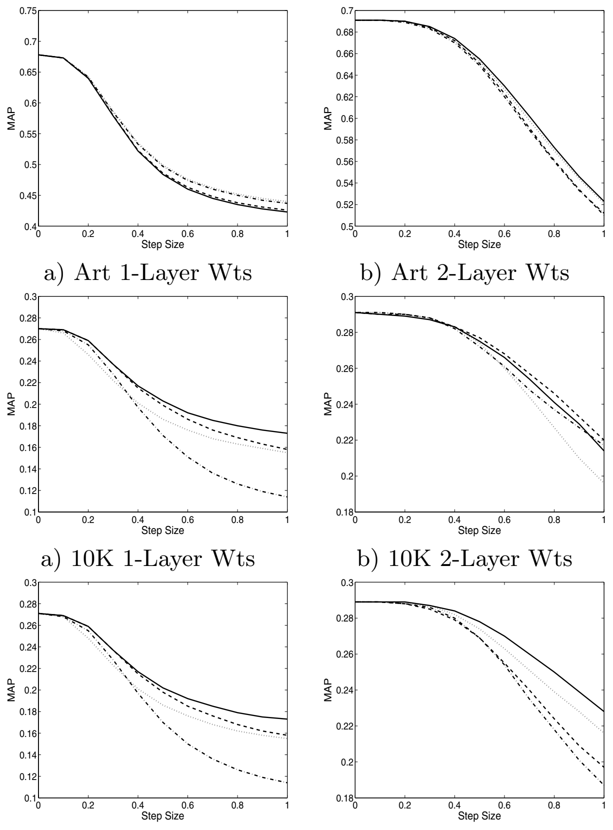
Figure 1 shows the change in mean NDCG on the three training sets when varying the weights of the single- and two-layer nets learned by *LambdaRankNDCG.* On all three training sets, mean NDCG decreases as we change the learned weights. The training accuracy is higher on the Artificial dataset, as expected, since it is much less noisy. In all cases, NDCG curves are smooth functions of the weights and it is apparent the learned weights result in the best accuracy.

Figure 2 shows the change in mean NDCG@10 on the three training sets when varying the weights of the single- and two-layer nets learned by *LambdaRankNDCG@10.* On all three training sets, mean NDCG@10 decreases as we change the values of the learned weights, indicating that the learned weights are indeed at a local optimum.

Figure 3 shows the results for weights learned by *LambdaRankMAP* on all three training sets for the single-layer and 2-layer nets. We see that all variations in learned weights cause a decrease in accuracy and satisfy the test for local optimality. The MAP score on the Artificial Data is higher than the MAP score on the Web datasets.

Lastly, we evaluate the local optimality of the weights learned by *LambdaRankMRR.* MRR training accuracy decreases as a relatively smooth function of the weights for both nets and all training sets, as shown in Figure 4, and therefore also satisfies the test for local optimality.

We also verify that the gradient vanishes at each learned weight by fixing all weights at their learned values except one and then randomly varying the one weight under consideration. As the weight value varies, we again check that training accuracy decreases with all variations. We find in all cases that the gradients vanish at all learned weights. We do not include figures due to space constraints.

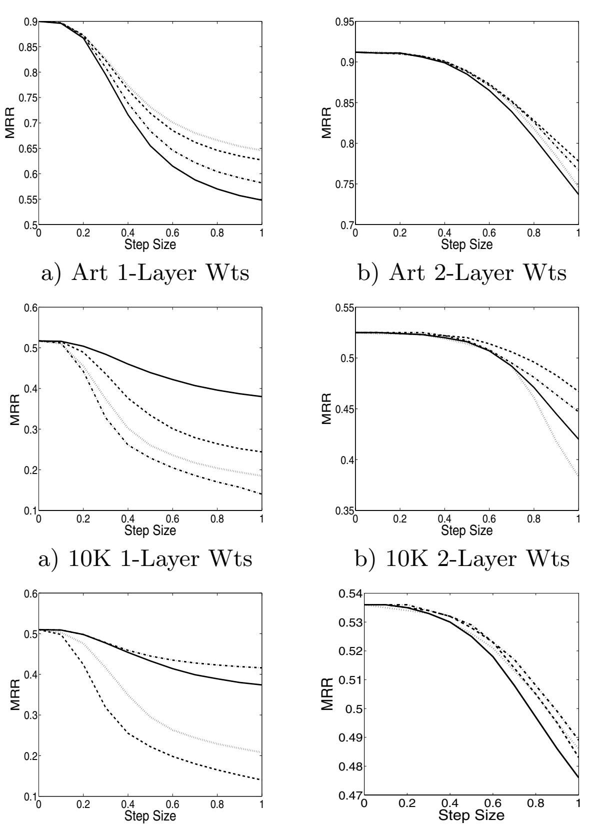


a) Art 1-Layer Wts b) Art 2-Layer Wts

a) 10K 1-Layer Wts b) 10K 2-Layer Wts

a) 22K 1-Layer Wts b) 22K 2-Layer Wts

Figure 3: Shifting single- and two-layer weights for MAP. x-axis is the step size, η.



a) Art 1-Layer Wts b) Art 2-Layer Wts

a) 10K 1-Layer Wts b) 10K 2-Layer Wts

a) 22K 1-Layer Wts b) 22K 2-Layer Wts

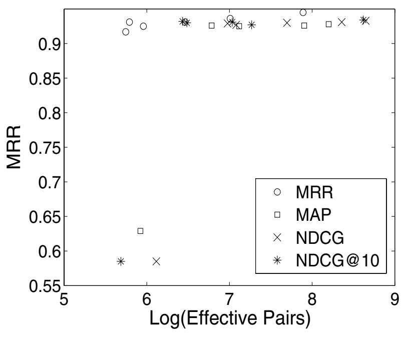
Figure 4: Shifting single- and two-layer weights for MRR. x-axis is the step size, η.

7.2 Matching Training and Target Measures

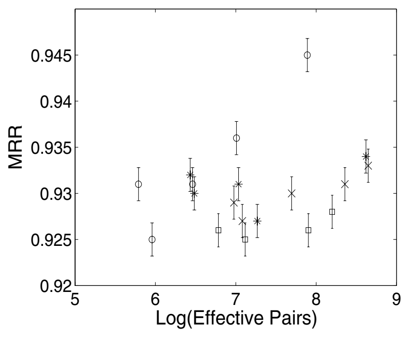
We investigate if matching training and target measures yields the best test accuracy, for all four IR measures. For all tables in this section, the columns in left-to-right order list the target (test) measure, the training measure, the accuracy (evaluated using the target measure) on the test set, and the statistical significance of the test-train pair's test accuracy relative to the test accuracy of the pair with the same target measure as training measure (Y: yes, N: no, –: the matching test-train pair). We used a paired t-test to determine significance, with 95% confidence.

In Table 3, we report test accuracy of 2-layer nets on the 10K Web data. When testing on mean NDCG, training on mean NDCG gives the best test accuracy, with statistical significance. Similarly, we find that when testing on MAP, training on MAP gives the best test accuracy, with statistical significance. In the case of mean NDCG@10, we find the highest test accuracy comes from training on mean NDCG. Simiarly, we find that the highest test accuracy for MRR comes from training on mean NDCG. We could be inclined to conclude that matching the training measure to the target measure does not necessarily yield the best test accuracy.

However, to correctly interpret the table, we must consider the number of *effective* training pairs, that is, the number of pairs with a non-zero λ-gradient that contribute to training. MAP and MRR are binary measures, while mean NDCG and mean NDCG@10 are multilevel relevance measures. The number of pairs that contribute to LambdaRank training for a binary measure is far fewer than the number for a multilevel relevance measure. For the 10K Web data, the upper bound on the number of effective pairs per epoch in the multilevel case is ~73 million compared to ~27 million in the binary case. For MRR, the pairwise λ-gradient is non-zero only when the pair contains a non-relevant document ranked higher than the top relevant document (see Eqn 11); no other pairs contribute to training so the number of effective pairs per epoch is around 1–2 magnitudes less than the binary upper bound. MAP, on the other hand, is within a constant of the upper bound since pairs at all positions may contribute non-zero λ-gradients. Mean NDCG is within a constant of the multilevel upper bound since it considers documents in all positions, but mean NDCG@10 learns from fewer pairs since it only considers pairs which contain at least one document in the top 10 rank positions. As a result, mean NDCG has more effective pairs (more learning opportunity) than mean NDCG@10, which has more than MAP, which has more than MRR. The 22K Web data has at most 60 and 162 million pairs per epoch in the binary and multilevel cases, respectively. Table 4 shows the accuracy on the test data of from 2-layer net training on the 22K data. We see that matching the training and target measures for both mean NDCG and MAP yields the best test accuracy with statistical significance. For MRR and NDCG@10, on the other hand, it does not. However, for MRR, learning from more pairs does decrease the differ-



a) Training pairs vs. MRR



b) Detail of (a)

Figure 5: Number of effective training pairs ($\log 10$) versus MRR test accuracy. Results from four trained models are reported. We do not plot error bars since the standard error for each point is less than 0.0018.

Table 3: Test accuracies on 10K Web Data for 2-layer LambdaRank trained on different training measures. Bold indicates statistical significance at 95% confidence against all other test-train pairs.

Test	Train	Test Score ± SE	Sig.
NDCG	**NDCG**	$\mathbf{0.723 \pm 0.001}$	–
	NDCG@10	0.722 ± 0.001	Y
	MAP	0.718 ± 0.001	Y
	MRR	0.704 ± 0.001	Y
NDCG@10	NDCG	0.442 ± 0.002	N
	NDCG@10	0.422 ± 0.002	–
	MAP	0.435 ± 0.002	Y
	MRR	0.403 ± 0.002	Y
MAP	NDCG	0.335 ± 0.002	Y
	NDCG@10	0.335 ± 0.002	Y
	MAP	$\mathbf{0.337 \pm 0.002}$	–
	MRR	0.314 ± 0.002	Y
MRR	NDCG	0.546 ± 0.004	Y
	NDCG@10	0.548 ± 0.004	Y
	MAP	0.546 ± 0.004	Y
	MRR	0.526 ± 0.004	–

Table 4: Test accuracies on 22K Web Data for 2-layer LambdaRank trained on different training measures. Bold indicates statistical significance at 95% confidence against all other test-train pairs.

Test	Train	Test Score ± SE	Sig.
NDCG	**NDCG**	$\mathbf{0.726 \pm 0.001}$	–
	NDCG@10	0.724 ± 0.001	Y
	MAP	0.723 ± 0.001	Y
	MRR	0.709 ± 0.001	Y
NDCG@10	NDCG	0.452 ± 0.002	Y
	NDCG@10	0.448 ± 0.002	–
	MAP	0.447 ± 0.002	N
	MRR	0.415 ± 0.002	Y
MAP	NDCG	0.341 ± 0.002	Y
	NDCG@10	0.338 ± 0.002	Y
	MAP	$\mathbf{0.343 \pm 0.002}$	–
	MRR	0.322 ± 0.002	Y
MRR	NDCG	0.550 ± 0.004	Y
	NDCG@10	0.547 ± 0.004	Y
	MAP	0.550 ± 0.004	Y
	MRR	0.539 ± 0.004	–

ence in accuracy between MRR and mean NDCG (0.02 to 0.011 points MRR) and for NDCG@10 we see a similar decrease between mean NDCG@10 and mean NDCG (0.02 to 0.004 points NDCG@10). Since both MRR and NDCG@10 learn from fewer effective pairs than NDCG@10, it seems likely that with more training pairs, matching the training and target measures will yield the best accuracy. To address this, we created various sizes of artificial training data to determine if learning on roughly similar numbers of effective pairs would reveal matching training and target measures to yield the best accuracy. Figure 5 plots the number of effective pairs versus the MRR test accuracy for four different training measures. For similar numbers of effective pairs, training for MRR results in better MRR test accuracy, with statistical significance. We find similar results for NDCG@10 (not shown here). Thus, we conclude that when training on the same number of effective pairs, it is indeed best to match the training measure to the target measure.

When a sufficiently large training sample is not available, it may be best to use a finer-grained training measure, e.g. mean NDCG, since it takes advantage of the most training pairs. Robertson and Zaragoza [8] claim different measures have different local optima, and likely have many local optima. They point out that a measure that responds to many flips (e.g. MAP and mean NDCG) will have many local optima whereas a measure that responds to fewer flips (e.g. MRR) will have fewer optima, but larger flat areas. These are the large areas of the parameter space that it cannot distinguish [8]. In other words, MAP and mean NDCG are more flexible than MRR, and mean NDCG has more granularity.

8. DISCUSSION AND CONCLUSION

Since we find all learned weights to be at local optima, two questions immediately arise: are weights at earlier epochs also local optima and are weights learned for one measure a local optimum for another measure? We looked at the NDCG@10 training accuracy of 2-layer NDCG@10-learned weights at epoch 7 and 12 on the 10K data. We found the weights at both epochs to be local optima. We then examined the mean NDCG@10 training accuracy of the 2-layer MRR-learned weights and found that they too are at

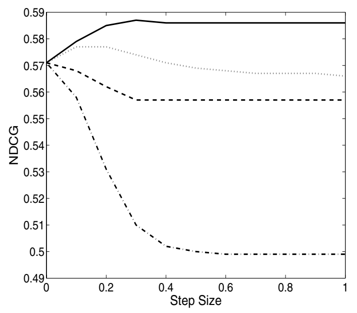


Figure 6: Shifting random single-layer weights for mean NDCG. x-axis is the step size, η.

a local optimum. However, initializing learning for mean NDCG@10 at the early epoch weights and the MRR-learned weights results in immediate mean NDCG@10 training accuracy improvement, indicating that learning escapes each of these local optima and climbs to a different, potentially better, local optimum. The escape occurs due to the stochasticity of the batch LambdaRank learning (both full batch and batch per query). Eventually, training converges, and these are the local optima reported in Section 7.1. In light of these results, we perform a sanity check to verify that a net with random weights is not at a local optimum. Figure 6 plots the step size η versus the mean NDCG training accuracy on the 10K data of a single-layer net for four of 459 random directions. Confidently, we find that the random weights are not at a local optimum, since training accuracy increases as we move away from the initial random weights in two of the four plotted random directions. Although LambdaRank starts from random weights, it appears that within a very small number of training epochs that the learning in fact guides the weight values into a region containing many local optima. The stochasticity in LambdaRank allows the algorithm to then repeatedly escape local optima and climb, and ultimately converge, to a superior local optimum. Our work shows that the learned weights upon convergence are locally optimal and points to the importance of the stochasticity in LambdaRank. This result is significant since LambdaRank builds upon smooth approximations to any target gradient rather than a direct optimization and still converges to a local optimum for various IR measures.

In conclusion, the direct optimization of IR measures has been very challenging, causing IR practitioners to build models with one (like BM25) or a few parameters that can be optimized using grid-search. However, it was recently shown that learning a model on many weak features can significantly improve test accuracy [1, 2]. In this paper, we have shown that four IR measures can in fact be optimized directly, and our λ-gradient construction is very likely extendable to other IR measures. We also show that with enough training data, matching the training measure to the target measure results in the best test accuracy. Our results open up a world of possibilities for directly optimizing sophisticated models on large numbers of features for possibly any IR measure of interest.

An interesting direction for future work is to revisit the 13 λ-gradient constructions described in Section 7 and determine how the number of effective training pairs varies across these constructions for various IR measures. Some of these constructions are not pair based in fact, and they may result in lower training accuracy due to the small number of effective pairs used for learning. Furthermore, it may be that 'RankNetWeightPairs' employs the most effective training pairs of the 13 constructions. It would also be fruitful to learn if other λ-gradient constructions, and possibly other learning to rank algorithms, converge to a local optimum, and escape local optima during learning.

9. REFERENCES

[1] C.J.C. Burges, R. Ragno, and Q.V. Le. Learning to rank with nonsmooth cost functions. In *Neural Information Processing Systems (NIPS)*, 2006. See also MSR Technical Report MSR-TR-2006-60.

[2] C.J.C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005.

[3] Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai, and H. Li. Learning to rank: From pairwise to listwise approach. In *International Conference on Machine Learning (ICML)*, pages 129–136, 2007.

[4] K. Crammer and Y. Singer. Pranking with ranking. In *Neural Information Processing Systems (NIPS)*, 2001.

[5] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132, 2000.

[6] T. Qin, T.Y. Liu, and H. Li. A general approximation framework for direct optimization of information retrieval measures. *Microsoft Technical Report MSR-TR-2008-164*, 2008.

[7] T. Qin, X.-D. Zhang, M.-F. Tsai, D.-S. Wang, T.-Y. Liu, and H. Li. Query-level loss functions for information retrieval. *Information Processing and Management*, 44(2):838–855, 2007.

[8] S. Robertson and H. Zaragoza. On rank-based effectiveness measures and optimization. *Information Processing and Management*, 10:321–339, 2007.

[9] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005.

[10] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning (ICML)*, 2004.

[11] J. Xu and H. Li. Adarank: A boosting algorithm for information retrieval. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 391–398, 2007.

[12] Y. Yue and C.J.C Burges. On using simultaneous perturbation stochastic approximation for ir measures, and the empirical optimality of lambdarank. *NIPS Machine Learning for Web Search Workshop*, 2007.

[13] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2007.