On Using Simultaneous Perturbation Stochastic Approximation for Learning to Rank, and the Empirical Optimality of LambdaRank

Yisong Yue
Dept. of Computer Science
Cornell University
Ithaca, NY 14850
`yyue@cs.cornell.edu`

Christopher J. C. Burges
Microsoft Research
Microsoft Corporation
Redmond, WA 98052
`cburges@microsoft.com`

Microsoft Research Technical Report MST-TR-2007-115

Abstract

One shortfall of existing machine learning (ML) methods when applied to information retrieval (IR) is the inability to directly optimize for typical IR performance measures. This is in part due to the discrete nature, and thus non-differentiability, of these measures. When cast as an optimization problem, many methods require computing the gradient. In this paper, we explore conditions where the gradient might be numerically estimated. We use Simultaneous Perturbation Stochastic Approximation as our gradient approximation method. We also examine the empirical optimality of LambdaRank, which has performed very well in practice.

1 Introduction

In recent years, the problem of learning to rank has gained importance in the fields of machine learning (ML) and information retrieval (IR). Ranking functions trained using ML techniques are currently in use by commercial web search engines. Due to the growth of interest in this area, the amount of training data available for learning has likewise grown. One shortfall of existing ML methods is the inability to directly optimize for IR performance measures, such as mean average precision and normalized discounted cumulative gain (NDCG) [9].

Gradient descent is a common and effective method for directly optimizing an objective function within some search space. When casting learning to rank as an optimization problem, one can consider the search space to be the space of possible parameter values for some ranking function. Unfortunately, one cannot easily use IR measures as the objective function since, in general, they are not differentiable with respect to the ranking function parameters. As a result, ML methods typically optimize for a surrogate objective function (which is differentiable, and often convex) or use an approximate gradient.

Given the availability of large training sets, we investigate whether the NDCG gradient might be numerically approximated. Numerical approximation requires measuring the change in the objective function over a small interval in the search space. This can become extremely expensive when dealing with high dimensional search spaces. We therefore use Simultaneous Perturbation Stochastic Approximation (SPSA) as our gradient approximation method, since is it very efficient and requires only two objective function evaluations for gradient approximation. We find that NDCG does become significantly smoother with additional training data, but still not enough to effectively perform gradient descent. However, we do anticipate that datasets of sufficient size might become available in the foreseeable future.

We also examine the potential optimality of LambdaRank. LambdaRank is a gradient descent method which uses an approximation to the NDCG "gradient", and has performed very well in practice. Our experiments show that the gradient approximation used by LambdaRank does in fact find a local optimum with respect to NDCG, even though the gradient used is a heuristic approximation.

This paper is organized as follows. We first discuss difficulties in directly optimizing NDCG and other common IR measures. We continue with an overview of related work as well as a description of LambdaRank. We then describe stochastic approximation techniques, most notably SPSA. Finally, we present experimental results and offer our conclusions and directions for future work.

2 Common Performance Measures for Information Retrieval

Performance measures used for information retrieval tasks are typically defined over rankings of documents for some given query. Relevance labels can be either binary (0 for non-relevant, 1 for relevant) or multilevel (0, 1, 2, . . .). Binary measures include Mean Average Precision, Mean Reciprocal Rank and Winner Takes All. See [17] for more details.

Normalized Discounted Cumulative Gain (NDCG) [9, 17] is a cumulative, multilevel measure that is usually truncated at a particular rank level. For a given query q_i, NDCG is computed as

$$\text{NDCG}_i \equiv N_i \sum_{j=1}^{T} \frac{2^{l_i(j)} - 1}{\log(1 + j)}, \tag{1}$$

where $l_i(j)$ is the label of the jth document in the ranking for q_i. The normalization constant N_i is chosen so that the perfect ranking would result in $\text{NDCG}_i = 1$, and T is the ranking truncation level at which NDCG is computed. NDCG is well suited for applications to Web search since it is multilevel and the truncation level can be set to model user behavior. Thus we will focus on NDCG in this paper.

2.1 Direct Optimization

Tuning model parameters to maximize performance is often viewed as an optimization problem in parameter space. In this setting, given a collection of training examples, we are concerned with optimizing NDCG with respect to the parameters of some ranking function.

Existing ranking functions usually score each document's relevance independently of other documents. The ranking is then computed by sorting the scores, as suggested by the Probability Ranking Principle [16]. Variations to these ranking functions' parameters will change the scores, but not necessarily the ranking. The measures discussed above are all computed over the rank positions of the documents. Therefore, the above measures have gradients that are zero wherever they are defined: that is, viewed as functions of the model score, typical IR measures are either flat or discontinuous everywhere.

However, what is optimized is usually the IR measure, averaged over all queries in the training set. Given enough data, we might hope that the corresponding function becomes smooth enough for empirical gradients to be defined after all. This paper explores conditions under which an NDCG gradient might exist and whether SPSA can be used to efficiently perform stochastic gradient descent. We focus on neural nets as our function class, which were also considered in [1, 2, 5].

3 Related Work

Previous approaches to directly optimizing IR measures either used grid search, coordinate ascent, or steepest ascent using finite difference approximation methods [14, 15] (see Section 4 for discussion on stochastic approximation methods). Metzler & Croft [15] used a Markov Random Field ranking method and showed that MAP is empirically concave when using a parameter space with two degrees of freedom. In this study, we consider parameter spaces with much larger degrees of freedom.

Direct optimization becomes difficult with large datasets and parameter spaces with many degrees of freedom. Most other approaches choose instead to optimize an alternative smooth objective function. Perhaps the most straightforward approach is learning to predict the relevance level of individual documents using either regression or multiclass classification [13]. Another popular approach learns using the pairwise preferences between documents of different relevance levels [10, 7, 6, 2, 8, 4, 3]. While these methods perform reasonably well in practice and are computationally convenient, they do not optimize for IR measures directly and offer no performance guarantees. Some more recent studies focus on minimizing relaxed upper bounds of IR performance loss [22, 12, 21]. These methods do offer partial performance guarantees.

Another important class of approaches uses approximations to conceptualize a gradient for IR performance measures (despite these measures being non-differentiable in general). Of these, we examine LambdaRank by Burges et al. [1], as it performs very well in practice and, like this study, uses neural nets for its function class. We discover that LambdaRank appears to find a local optimum for NDCG.

3.1 LambdaRank

LambdaRank is a general gradient descent optimization framework that only requires the gradient to be defined, rather than the objective function. In the case of learning to rank, we focus on document pairs (i, j) of different relevance classes (i more relevant than j). The derivative of such pair with respect to the neural net's output scores is defined as

$$\lambda_{ij} = N\left(\frac{1}{1+e^{s_i-s_j}}\right)\left|\left(2^{\ell_i}-2^{\ell_j}\right)\left(\frac{1}{\log(1+r_i)}-\frac{1}{\log(1+r_j)}\right)\right|,$$

where s_i is the output score, ℓ_i is the relevance label, and r_i is the (sorted) rank position of document i. The normalization factor N is identical to the one in (1). Let D_i^+ and D_i^- denote the set of documents with higher and lower relevance classes than i, respectively. The total partial derivative with respect to document i's output score is

$$\lambda_i \equiv \sum_{j \in D_i^-} \lambda_{ij} - \sum_{j \in D_i^+} \lambda_{ji}. \tag{2}$$

For each document, the LambdaRank gradient computes the NDCG gain from swapping rank positions with every other document (of a different relevance class) discounted by a function of the score difference between the two documents. This discount function is actually the RankNet gradient [2] (see (9) in Section 5.2.1 below). The objective function of LambdaRank can in principle be left undefined, since only the gradient is required to perform gradient descent, although for a given sorted order of the documents, the objective function is simply a weighted version of the RankNet objective function [2].

4 Stochastic Approximation

Assuming an objective function $L : \mathcal{R}^d \to \mathcal{R}$, an optimum w^* of L satisfies the property that the gradient vanishes at that point,

$$g(w^*) \equiv \frac{\partial L(w^*)}{\partial w^*} = 0.$$

In cases when the gradient is not directly computable and evaluations of L are noisy, stochastic approximation techniques are often used to approximate the gradient.

Given an approximation $\hat{g}(w)$ to the true gradient, L can be iteratively optimized by stepwise gradient descent,

$$w_{k+1} = w_k + a_k \hat{g}_k(w_k), \tag{3}$$

where $a_k \in \mathcal{R}$ is the optimization step size at iteration k.

The most common stochastic approximation technique, Finite Difference Stochastic Approximation (FDSA) [11], approximates each partial derivative separately as

$$\hat{g}_k(w_k)_i = \frac{L(w_k + c_k e_i) - L(w_k - c_k e_i)}{2c_k},$$

where $c_k \in \mathcal{R}$ is the approximation step size and $e_i \in \mathcal{R}^d$ is the unit vector along the ith axis. This method requires $2d$ function evaluations at each iteration, which can be prohibitively expensive if L is non-trivial to compute (e.g., the foward propagation required to score documents and sort required to compute NDCG).

4.1 SPSA

We now describe Simultaneous Perturbation Stochastic Approximation (SPSA), which was first proposed by Spall [19, 20]. SPSA is an efficient method for stochastic gradient approximation. In contrast to FDSA, which performs $2d$ function evaluations per iteration, the simplest form of SPSA requires only 2.

As the name suggests, a simultaneous perturbation vector $\Delta_k \in \mathcal{R}^d$ is used in each iteration. Given Δ_k, the gradient approximation is computed as

$$\hat{g}_k(w_k) = \begin{bmatrix} \frac{1}{\Delta_{k1}} \\ \frac{1}{\Delta_{k2}} \\ \cdots \\ \frac{1}{\Delta_{kd}} \end{bmatrix} \cdot \frac{L(w_k + c_k\Delta_k) - L(w_k - c_k\Delta_k)}{2c_k}.$$

Following the conditions stated in [19], Δ_k is a vector of d mutually independent mean-zero random variables $(\Delta_{k1}, \Delta_{k2}, \ldots, \Delta_{kd})$ satisfying $|\Delta_{kl}| \leq \alpha_0$ almost surely and $E|\Delta_{kl}^{-1}| \leq \alpha_1$ for some finite α_0 and α_1. As suggested in [18], we choose each Δ_{kl} to be symmetrically Bernoulli distributed ($+1$ or -1 with equal probability).

When the objective function is extremely noisy or non-linear, multiple SPSA gradients can be computed and averaged together at each iteration.

4.2 Correctness Results

This section highlights two correctness results for SPSA. Details are available in [19]. The two key results are that (A) SPSA produces an unbiased estimate of the true gradient and that (B) the accumulated sampling error vanishes.

Let the bias of $\hat{g}_k(\cdot)$ be defined as

$$b_k(w_k) = E[\hat{g}_k(w_k)|w_k] - g(w_k), \tag{4}$$

where $g(\cdot)$ is the true (unknown) gradient of L. Here the expectation is over both the Bernoulli variables, and also over zero mean additive noise in the measurements of L. We require that L be thrice differentiable and satisfying $|L^{(3)}_{i_1 i_2 i_3}| \leq \alpha_2$ for some finite α_2. In order for SPSA to produce unbiased estimates of $g(\cdot)$, it must be the case that $b_k(\cdot) \to 0$ as k grows. This leads us to Proposition A:

Proposition A: Assume the conditions on Δ_k and L stated above, and consider all $k \geq K$ for some $K < \infty$. Then the bias term defined in (4) behaves like $O(c_k^2)$ (where c_k is the approximation step size).

Proposition A is proved as Lemma 1 in [19]. As long as the c_k are chosen to be decreasing in k, then the bias of the SPSA estimate will vanish asymptotically.

Let the sampling error of $\hat{g}_k(w_k)$ be defined as

$$\epsilon_k(w_k) = \hat{g}_k(w_k) - E[\hat{g}_k(w_k)|w_k]. \tag{5}$$

Then we can rewrite (3) as

$$w_{k+1} = w_k + a_k[g(w_k) + b_k(w_k) + \epsilon_k(w_k)]. \tag{6}$$

Proposition B: Let $\epsilon_k(\cdot)$ be defined as in (5). As $k \to \infty$, let

$$a_k \to 0, \quad c_k \to 0, \quad \sum_{k=0}^{\infty} a_k = \infty, \quad \sum_{k=0}^{\infty} \left(\frac{a_k}{c_k}\right)^2 < \infty. \tag{7}$$

Then

$$\forall \eta > 0, \qquad \lim_{k\to\infty} Pr\left[\sup_{m\geq k} \left\|\sum_{i=k}^{m} a_i \epsilon_i(w_i)\right\| \geq \eta\right] = 0. \tag{8}$$

Proposition B is proved as part of Proposition 1 in [19]. By defining our step sizes a_k, c_k appropriately to satisfy (7), the bias term $b_k(w_k)$ and the accumulated sum of sampling errors $\sum_k a_k \epsilon_k(w_k)$ in (6) both vanish asymptotically. The result in (8) is stronger than it may at first appear – note from (7) that we require the sum of a_k to be unbounded despite having $a_k \to 0$ as k increases. Thus it is non-trivial to prove (8). We refer the reader to Proposition 1 in [19] for a formal analysis of the strong convergence of SPSA.

4.3 Rate of Convergence

If evaluations of L is the computation bottleneck for gradient approximation, then each iteration of SPSA will be d times faster than FDSA. One important consideration is whether SPSA's per-iteration convergence rate is less than d times slower. Spall [19] empirically showed that one can reasonably expect SPSA's convergence rate (by iteration count) to be much less than d times that of FDSA. When measured in terms of evaluations of L, SPSA is then much faster. In this paper we will empirically evaluate the convergence rate of SPSA vs. FDSA on a large Web search dataset.

5 Experiments

We performed experiments on two datasets: an artifical dataset and a real Web search dataset. Our experiments used neural nets trained with SPSA, FDSA and LambdaRank. Our experiments were designed to investigate three questions: (A) whether SPSA converges faster than FDSA for Web search data, (B) whether NDCG becomes empirically smooth given enough data, and therefore become trainable using SPSA, and if so then (C) whether SPSA can achieve results competitive with LambdaRank. We also report some results using RankNet [2], although it has already been shown to be outperformed by LambdaRank [1].

For LambdaRank, we varied the learning rate from 1e-7 to 1e-2. We used a validation set to choose the best model for evaluation on the test set. We fixed the size of the hidden layer to 10 nodes for all two layer nets. For SPSA and FDSA, we tried a number of step size sequences. We found FDSA to be less sensitive to the choice of step sizes. Following [19], our step size sequences follow the form

$$a_k = \frac{a_0}{A + k^\alpha}, \quad c_k = \frac{c_0}{k^\gamma},$$

where α and γ are chosen to satisfy the conditions stated in (7). For our experiments, we only report the results using $\alpha = 0.602$ and $\gamma = 0.101$, as they achieved the best convergence rates on the training set and were also suggested by [19, 20]. We also only report results using $A = 50$. The specific value of A is not important as it is intended to help avoid instabilities in the early iterations [20]. We chose a_0 and c_0 to achieve the best convergence rate on the training set.

We denote an SPSA variant using F function evaluations per iteration as SPSA:F. The basic SPSA algorithm is thus named SPSA:2.

5.1 Datasets

We used both an artifical dataset as well as a "real" dataset generated from a commercial search engine. We name the datasets Artificial and Web. These are identical to the similarly named datasets used in [1].

Artificial. We used artificial data to remove any variance stemming from the quality of the features or of the labeling. We followed the prescription given in [2] for generating random cubic polynomial data. However, here we use five levels of relevance instead of six, a label distribution corresponding to real datasets, and more data, all to more realistically approximate a

Web search application. We used 50 dimensional data, 50 documents per query, and 10K/5K/10K queries for train/valid/test respectively.

Web. This data is from a commercial search engine and has 367 dimensions, with on average 26.1 documents per query. The data was created by shuffling a larger dataset and then dividing into train, validation and test sets of size 10K/5K/10K queries, respectively.

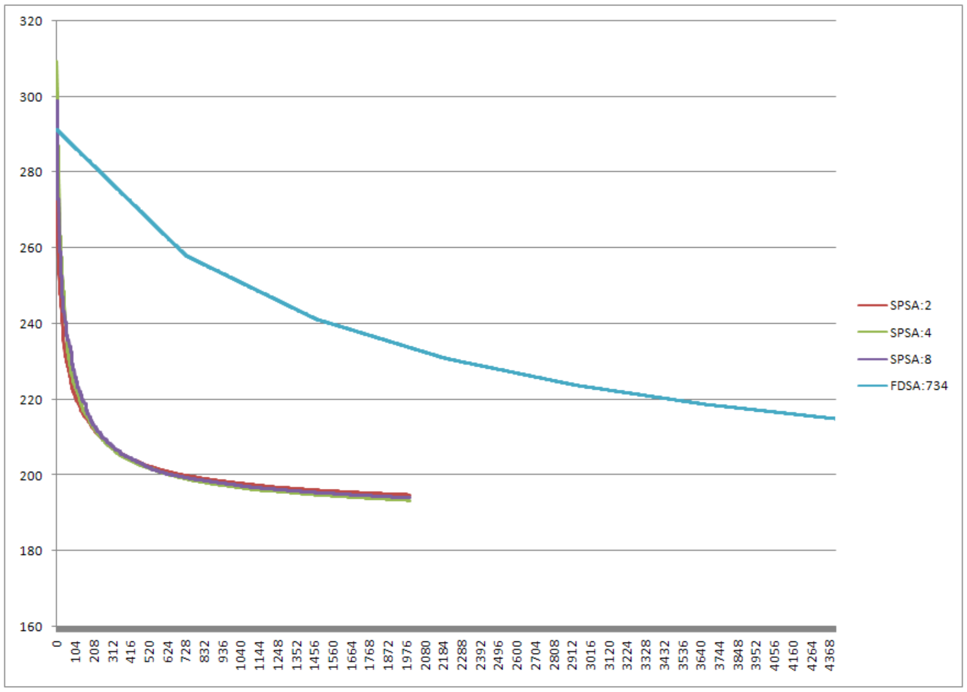

Figure 1: Cross Entropy w.r.t. Number of Function Evaluations on Web Training Data

5.2 SPSA vs. FDSA

We empirically evaluated the convergence speed of SPSA vs. FDSA on the Web dataset for minimizing pairwise cross entropy as a “sanity check”. Pairwise cross entropy is the objective used for RankNet training [2]. We chose this metric since the objective function is differentiable, and so the non-smoothness of the cost function is removed as a possible factor, enabling us to cleanly compare FDSA and SPSA. For this experiment, we chose for our function class single layer neural networks.

The Web data contains 367 features, causing FDSA to perform 734 objective function evaluations per iteration. We evaluated three SPSA variants. SPSA:2 performs one gradient approximation (2 function evaluations) per iteration. SPSA:4 and SPSA:8 perform two and four gradient approximations, respectively. We report the mean pairwise cross entropy values number of function evaluations over ten runs of each method.

5.2.1 Pairwise Cross Entropy

For any pair of documents (i, j) with document i having higher relevance than j, the cross entropy is computed as

$$C_{ij} \equiv s_j - s_i + \log(1 + e^{s_i - s_j}),$$

where s_i is the neural network output score of document i. The partial derivatives can then be expressed as

$$\frac{\partial C_{ij}}{\partial s_i} = -\frac{\partial C_{ij}}{\partial s_j} = \frac{-1}{1 + e^{s_i - s_j}}. \tag{9}$$

The total pairwise entropy C is the sum all C_{ij} values. Let D_i^+ and D_i^- denote the set of documents with higher and lower relevance labels than i, respectively. The total partial derivative for s_i can be written as

$$\frac{\partial C}{\partial s_i} = \sum_{j \in D_i^-} \frac{\partial C_{ij}}{\partial s_i} + \sum_{j \in D_i^+} \frac{\partial C_{ji}}{\partial s_i}.$$

We empirically verified that the resulting FDSA gradient is virtually identical to the closed form solution. This allows us to use the gradient formulation in place of the FDSA gradient, and is much faster computationally.

5.2.2 Convergence Speed Results

The performance difference of SPSA vs FDSA is quite striking. Figure 1 shows the pairwise cross entropy value of FDSA and three variants of SPSA plotted against number of function evaluations. Here, we see that all variants of SPSA converge significantly faster than FDSA. Recall that in this setting, performing a single function evaluation requires forward propagating all the training examples to compute the output scores and then performing a pairwise difference to compute the cross entropy. It is interesting to note that performing averages of the SPSA gradient estimates neither helped nor hurt the convergence rate.

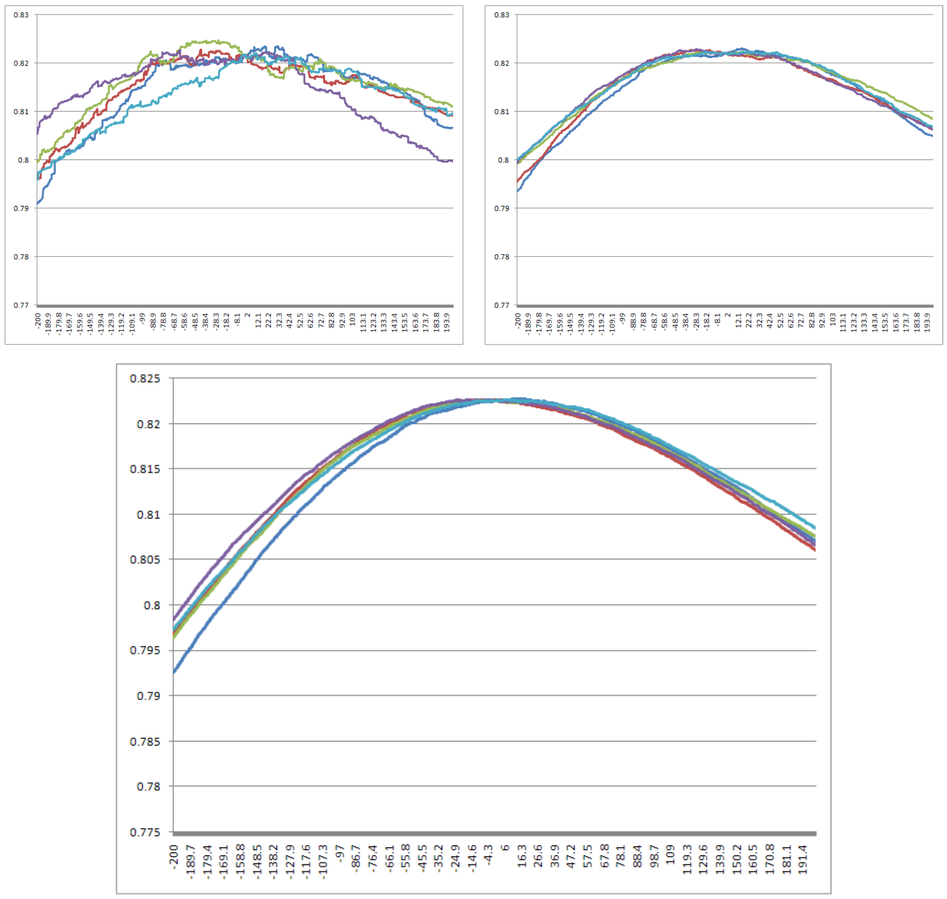

Figure 2: NDCG w.r.t. Shifting Top 5 Weights of Single Layer Net on 100, 1000 and 10000 Queries of Artificial Data

5.3 Smoothness of NDCG

SPSA assumes the objective function (in this case NDCG) is thrice differentiable. While the NDCG is either flat or discontinuous (and so non-differentiable) everywhere for a single query, it may become empirically smooth when averaged over a sufficient number of queries, just as any smooth function may be approximated as a linear combination of step functions. We can investigate this empirically by taking a trained LambdaRank net and comparing the change mean NDCG of the training set as one weight of the net is varied with all the other weights held fixed. We performed this comparison on both single and two layer nets for both the Artifical and Web

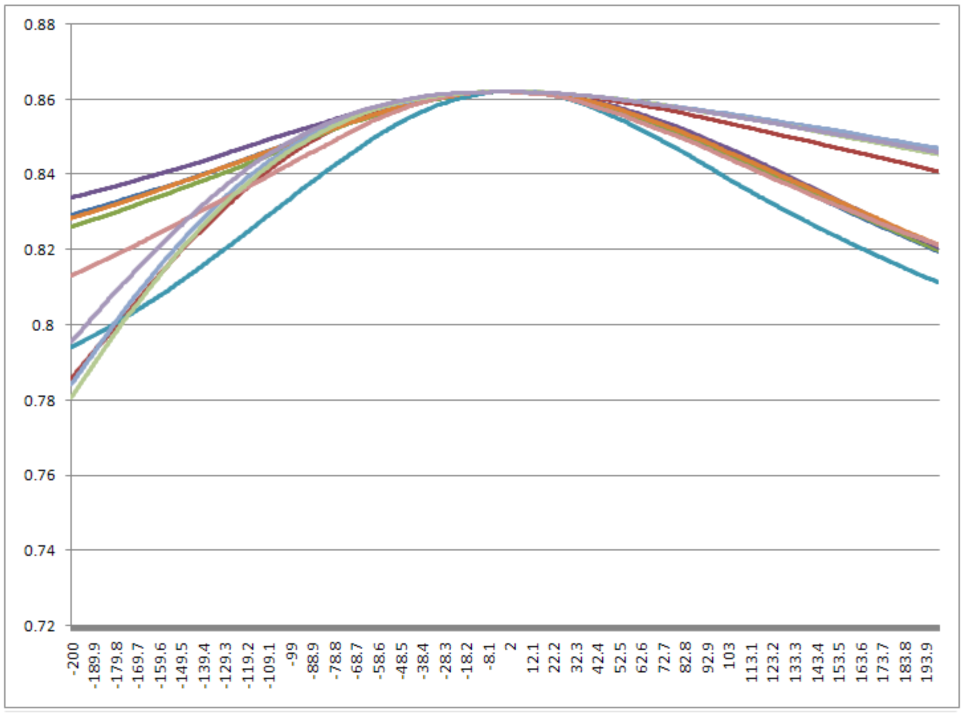

Figure 3: NDCG w.r.t. Shifting Hidden-Output Weights of Two Layer Net on Artificial Data

datasets. In all our comparisons, we varied each individual weight by a percentage of its trained weight.

We first compared the NDCG change when varying the weights of a single layer net. We report this comparison of the top five weights by absolute value. Figure 2 shows this result for a 100 query subsample, a 1,000 query subsample, and the entire 10,000 query Artificial training set. We observe that the NDCG function curve consistently becomes smoother as the query sample increases.

We also compared the NDCG change when varying the weights of a two layer net. We report this comparison for all ten weights connecting the hidden units to the output unit. Figures 3 & 4 show the results for the entire Artificial and Web training sets. Again, we observe that the NDCG function curves are relatively smooth on a macro scale. The inherent discontinuity of NDCG is observable only at a local scale.

However, the curves still contain numerous smaller local optima even when averaged over 10,000 queries. Figure 5 shows a blown up section

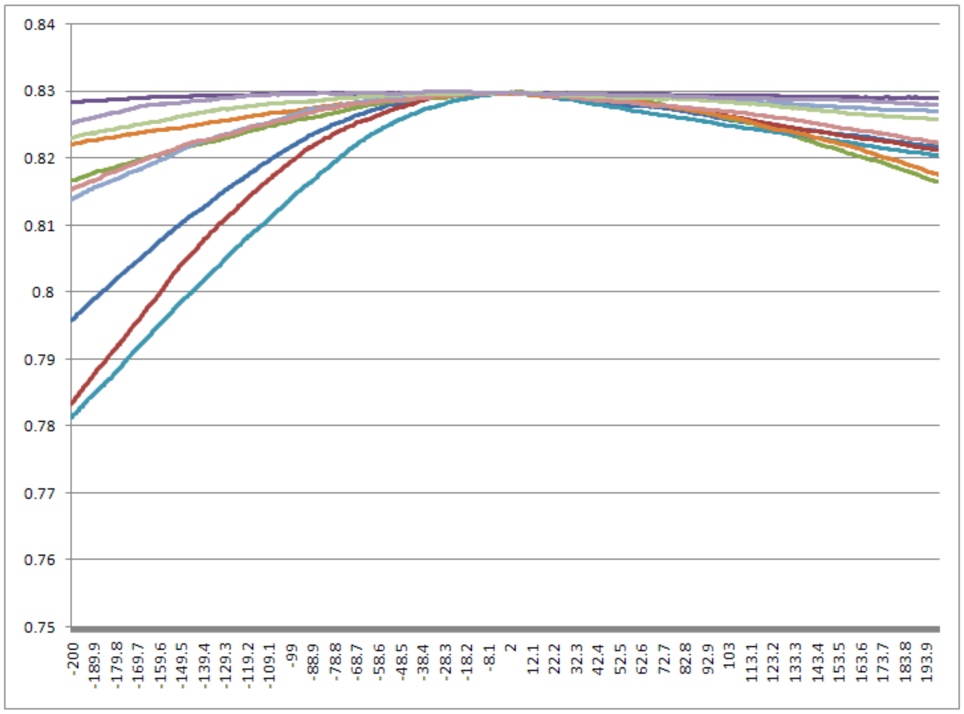

Figure 4: NDCG w.r.t. Shifting Hidden-Output Weights of Two Layer Net on Web Data

from Figure 4. Here we see that the discontinuities are very noticeable on a smaller scale. In order for SPSA to work well, the scale of the discontinuities must be smaller than the step size used to approximate the gradient.

Not surprisingly, our results for SPSA are significantly worse than both LambdaRank and RankNet. Given the added complexity of sorting when computing NDCG, we instead optimized SPSA for NDCG@10 instead of the non-truncated version. Each objective function evaluation still incurred a significant computational cost. As such, we only trained SPSA:4 on the Web data for single layer nets. We performed a limited parameter search for appropriate a_0 and c_0 values and chose the ones which gave the best validation performance. The NDCG@10 comparison is shown in Table 1.

Nonetheless, these results are encouraging. They suggest the feasibility of collecting a sufficient number of queries whereby the NDCG "gradient" becomes smooth enough to be effectively approximated by SPSA. More generally, they suggest that many objective functions previously considered infeasible to optimize directly might yield computable gradient approxima-

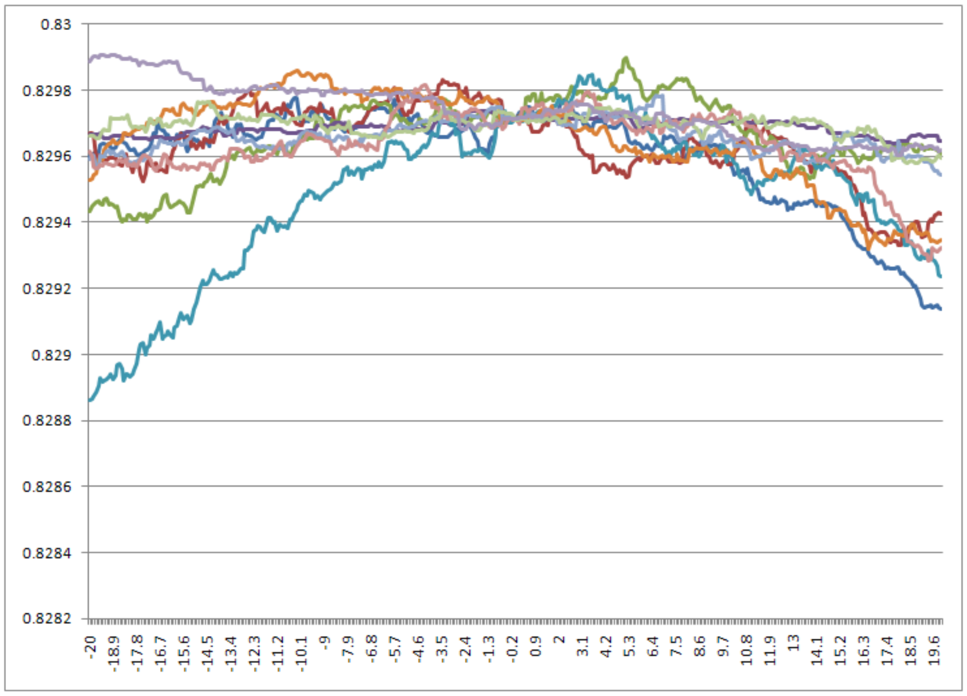

Figure 5: Blown Up Version of Figure 4

tions when given sufficient training data.

Method	Train	Valid	Test
LambdaRank	0.721	0.713	0.707
RankNet	0.715	0.709	0.699
SPSA:4	0.690	0.682	0.677

Table 1: NDCG@10 for Linear Nets on Web Data

5.4 Empirical Optimality of LambdaRank

The evaluation of the smoothness of NDCG also yields another interesting observation. The local optimum found using the LambdaRank gradient (2) also corresponds very closely to a local optimum of NDCG. Perhaps we should not find this result too surprising, since the true NDCG "gradient" should reflect the instantaneous change in NDCG as the scores of the documents vary, and the LambdaRank gradient can be interpreted as measuring a smoothed or convolved version of the change in NDCG, where the smoothing is a function of the distance between two documents' respective scores. However it should be noted that the LambdaRank gradient is not simply a smoothed version of the FDSA gradient; it will contain contributions from documents that have very different scores, if the current ranker puts them in the wrong order. In fact, in prior work we performed experiments using exactly the FDSA gradient (with respect to the model score) as the LambdaRank gradient, and found that this did not work as well as the published LambdaRank gradient [1]. However roughly speaking, LambdaRank incorporates smoothing into its approximation of the NDCG gradient whereas SPSA requires empirical smoothing (averaging over enough data).

This result suggests that, since LamdbaRank already finds a local NDCG optimum, then it will be non-trivial to improve on LambdaRank performance using (two layer) neural networks as the function class.

6 Conclusions & Future Work

We've presented evidence demonstrating a trend towards smoothness of NDCG as the dataset size grows. While we cannot exactly characterize of the smoothness of NDCG, we find it reasonable to expect, in the foreseeable future, having enough data to make methods such as SPSA effective. The scale of the inherent discontinuities need only be smaller than the step sized used for gradient approximation. Given the current training data available, we find that SPSA does not compare well with LambdaRank.

We also showed empirically that LambdaRank finds a local optimum for NDCG, despite using a (smooth) approximation of the NDCG gradient. Given these results, it appears difficult to improve on LambdaRank NDCG performance using (two layer) neural networks as the ranking function class.

These results also beg the question of whether LambdaRank has additional theoretical properties. One such question to ask is: given some distribution over the space of examples (queries and relevance labels), does a local optimum with respect to the LambdaRank gradient imply anything about the true gradient of the expected NDCG over that distribution?

We finally note that SPSA is a very general optimization framework. The objective function to be optimized need only satisfy (or approximately satisfy) the condition that its third derivatives exist, and be bounded, in order for SPSA to work in practice. While optimizing for NDCG is a well-studied problem, for many IR optimization problems SPSA may work reasonably well.

Acknowledgements

The authors would like to thank John Platt for his helpful comments as well as for first pointing us to SPSA.

References

[1] C. Burges, R. Ragno, and Q. Le. Learning to rank with non-smooth cost functions. In *Proceedings of NIPS'06*, 2006.

[2] C. Burges, T. Sheked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of ICML'05*, Bonn, Germany, August 2005.

[3] Y. Cao, J. Xu, T. Liu, H. Li, Y. Huang, and H. Hon. Adapting ranking svm to document retrieval. In *Proceedings of SIGIR'06*, 2006.

[4] B. Carterette and D. Petkova. Learning a ranking from pairwise preferences. In *Proceedings of SIGIR'06*, 2006.

[5] R. Caruana, S. Baluja, and T. Mitchell. Using the future to "sort out" the present: Rankprop and multitask learning for medical risk evaluation. In *Proceedings of NIPS'96*, 1996.

[6] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2004.

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

[8] A. Herschtal and B. Raskutti. Optimising area under the roc curve using gradient descent. In *Proceedings of ICML'04*, 2004.

[9] K. Jarvelin and J. Kekalainen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of SIGIR'00*, 2000.

[10] T. Joachims. A support vector method for multivariate performance measures. In *Proceedings of ICML'05*, 2005.

[11] J. Kiefer and J. Wolfowitz. Stochastic estimation of a regression function. *Annals of Mathematical Statistics*, 23:462–466, 1952.

[12] Q. Le and A. Smola. Direct optimization of ranking measures. arXiv:0704.3359, 2007.

[13] P. Li, C. Burges, and Q. Wu. Learning to rank using classification and gradient boosting. Technical report, Microsoft Research, 2007.

[14] D. Metzler. Direct maximization of rank-based metrics. Technical report, CIIR, 2007.

[15] D. Metzler and B. Croft. A markov random field for term dependencies. In *Proceedings of SIGIR'05*, 2005.

[16] S. Robertson. *The Probability Ranking Principle in IR*, pages 281–286. Morgan Kaufmann Publishers Inc., 1997.

[17] S. Robertson and H. Zaragoza. On rank-based effectiveness measures and optimisation. Technical report, Microsoft Research, 2006.

[18] P. Sadegh and J. Spall. Optimal random perturbations for stochastic approximation using a simultaneous perturbation gradient approximation. In *Proceedings of the American Control Conference*, 1997.

[19] J. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37:332–341, 1992.

[20] J. Spall. Implementation of the simultaneous perturbation algorithm for stochastic approximation. *IEEE Transactions on Aerospace and Electronic Systems*, 34:817–823, 1998.

[21] J. Xu and H. Li. A boosting algorithm for information retrieval. In *Proceedings of SIGIR'07*, 2007.

[22] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of SIGIR'07*, 2007.