Ranking via Sinkhorn Propagation

Ryan Prescott Adams* **and Richard S. Zemel**

Department of Computer Science
University of Toronto

e-mail: rpa@cs.toronto.edu; zemel@cs.toronto.edu

Abstract: It is of increasing importance to develop learning methods for ranking. In contrast to many learning objectives, however, the ranking problem presents difficulties due to the fact that the space of permutations is not smooth. In this paper, we examine the class of *rank-linear* objective functions, which includes popular metrics such as precision and discounted cumulative gain. In particular, we observe that expectations of these gains are completely characterized by the marginals of the corresponding distribution over permutation matrices. Thus, the expectations of rank-linear objectives can always be described through locations in the Birkhoff polytope, i.e., doubly-stochastic matrices (DSMs). We propose a technique for learning DSM-based ranking functions using an iterative projection operator known as Sinkhorn normalization. Gradients of this operator can be computed via back-propagation, resulting in an algorithm we call Sinkhorn propagation, or *SinkProp*. This approach can be combined with a wide range of gradient-based approaches to rank learning. We demonstrate the utility of SinkProp on several information retrieval data sets.

1. Introduction

The task of ranking is straightforward to state: given a query and a set of documents, produce a "good" ordering over the documents based on features of the documents and of the query. From the point of view of supervised machine learning, we define the "goodness" of a ranking in terms of graded relevances of the the documents to the query, using an objective function that rewards orderings that rank the more relevant documents highly. The task is to take a training set of queries in which the documents are labeled with known relevances, and build an algorithm that is capable of producing orderings over queries in which the relevance labels are unknown.

There are several aspects of this problem, however, that make it difficult to train a ranking algorithm and which have recently led to a flurry of insightful, creative approaches. The first thorny aspect of the learning-to-rank problem is that, unlike typical supervised classification and regression problems, we wish to learn a *family* of functions with varying domain and range. This difficulty arises from the property that queries may have varying numbers of documents; the set of input features scales with the number of documents and the size of the output ordering also changes.

The second difficulty in learning to rank is that the space of permutations grows rapidly as a function of the number of documents. A fully Bayesian decision-theoretic approach to the ranking problem would construct a rich

*http://www.cs.toronto.edu/~rpa

joint distribution over the relevances of documents in a query and then select the optimal ordering, taking this uncertainty into account. While this is feasible for small queries, this optimization quickly becomes intractable. It is therefore more desirable to develop algorithms can directly produce permutations, rather than relevances.

Finally, any objective defined in terms of orderings over relevance-labeled documents is necessarily piecewise-constant with respect to the parameters of the underlying function. That is, it may not be possible to compute the gradient of a training gain in terms of the parameters that need to be learned. Without such a gradient, many of the powerful machine learning function approximation tools, such as neural networks, become infeasible to train.

In this paper, we develop an end-to-end framework for supervised gradient-based learning of ranking functions that overcomes each of these difficulties. We examine the use of doubly-stochastic matrices as differentiable relaxations of permutation matrices, and define popular ranking objectives in those terms. We show that the expected value of an important class of ranking gains are completely preserved under the doubly-stochastic interpretation. We further demonstrate that it possible to propagate gradient information backwards through an *incomplete Sinkhorn normalization* to allow learning of doubly-stochastic matrices. We also show how this leads to flexibility in the choice of pre-normalization matrices and enables our approach to be integrated with other powerful ranking methods, even as the number of per-query documents vary. We note in particular that this approach is well-suited to take advantage of the recent developments in the training of deep neural networks.

2. Optimizing Expected Ranking Objectives via Doubly-Stochastic Matrices

In a learning-to-rank problem, the training data are N sets, called *queries*, the nth of which has size J_n. The items within each of these query sets are called *documents*. The features of document j in query n are denoted as $\boldsymbol{x}_j^{(n)} \in \mathcal{X}$. In the training set, each document also has a *relevance* label $r_j^{(n)} \in \{0, 1, 2, \ldots, R\}$, where R indicates the maximum relevance. We denote the vector of relevances for query n as $\boldsymbol{r}^{(n)}$.

A ranking of the documents for a given query can be represented as a permutation, mapping each document j to a rank k. Each permutation is an element of $\mathcal{S}_J$, the symmetric group of degree J. The aim then is to learn a family of functions that output permutations: $f_J : \mathcal{X}^J \to \mathcal{S}_J$, for $J \in \{1, 2, \ldots\}$, where, as above, $\mathcal{X}$ is a set of features associated with each of the J items.

2.1. Ranking Objective Functions

The most critical component when learning to rank is the definition of an objective function which identifies "good" and "bad" orderings of documents.

For a query of size J with known relevances, we identify three well-studied scoring functions over the group $\mathcal{S}_J$: the order-K *normalized discounted cumulative gain* (NDCG@K) [1], the order-K *precision* (P@K) [2], and the *rank biased precision* (RBP) [3].

Writing $s[k]$ to indicate the index of the document at rank k in permutation s, NDCG@K is given by

$$\begin{aligned}
\mathcal{L}_{\mathsf{NDCG@}K}(s\,;\,\boldsymbol{r}) &= \frac{\mathrm{DCG@}K(s\,;\,\boldsymbol{r})}{\max_{s'} \mathrm{DCG@}K(s'\,;\,\boldsymbol{r})} \\
\mathrm{DCG@}K(s\,;\,\boldsymbol{r}) &= \sum_{k=1}^{K} \mathcal{G}(r_{s[k]})\,\mathcal{D}(k) \\
\mathcal{G}(r) &= 2^r - 1 \\
\mathcal{D}(k) &= \begin{cases} 1 & \text{if } k = 1, 2 \\ \log_2(-k) & \text{if } k > 2 \end{cases}.
\end{aligned}$$

The P@K objective, for binary relevances is

$$\mathcal{L}_{\mathsf{P@}K}(s\,;\,\boldsymbol{r}) = \frac{1}{K} \sum_{k=1}^{K} r_{s[k]}$$

and RBP is

$$\mathcal{L}_{\mathsf{RBP}}(s\,;\,\boldsymbol{r}) = (1-\alpha) \sum_{k=1}^{J} r_{s[k]} \alpha^{k-1},$$

where $\alpha \in [0, 1]$ is a "persistence" parameter. The natural training objective, then, is to find a family of functions $\mathcal{F} = \{f_J : J \in \{1, 2, \ldots\}\}$ that maximizes the aggregate empirical gain, subject to some regularization penalty $Q(\mathcal{F})$:

$$\mathcal{F}^\star = \operatorname*{argmax}_{\mathcal{F}=\{f_J\}} \left\{ -Q(\mathcal{F}) + \sum_{n=1}^{N} \mathcal{L}(f_{J_n}(\{\boldsymbol{x}_j^{(n)}\}_{j=1}^{J_n})\,;\,\{r_j^{(n)}\}_{j=1}^{J_n}) \right\}. \quad (1)$$

2.2. Doubly-Stochastic Matrices as Marginals Over Permutations

As discussed in Section 1, one of the difficulties with the objective in Eq. (1) is that it is discontinuous. That is, changes in f_J only effect the training data via the discrete ordering, and so are piecewise-constant modulo the regularization $Q(\mathcal{F})$. One way to address this is to replace the objectives of the previous section with expectations of these objectives, where the expectations are with respect to a distribution over rankings [4]. That is, given a distribution over permutations, denoted by $\psi(s)$, and the relevances $\boldsymbol{r}$, the expected gain is

$$\mathbb{E}_\psi[\mathcal{L}(\boldsymbol{r})] = \sum_{s \in \mathcal{S}_J} \psi(s)\,\mathcal{L}(\boldsymbol{r}). \quad (2)$$

Many ranking objectives, however, are characterized by element-wise sums over the associated permutation matrix $\boldsymbol{S}$, i.e., $S_{j,k} = \delta_{j,s[k]}$, where $\delta_{j,k}$ is the Kronecker delta function. We call such objectives *rank-linear*, and they have the general form

$$\mathcal{L}(s\,;\,\boldsymbol{r}) = \sum_{j=1}^{J}\sum_{k=1}^{J} S_{j,k}\,\ell(r_j, k).$$

This has been observed previously in the literature, e.g., [5, 6]. One remarkable aspect of rank-linear objectives, however, is that the expectation in Eq. (2) is completely captured by the marginal probability that $S_{j,k}=1$:

$$\mathbb{E}_{\psi}[\mathcal{L}(\boldsymbol{r})] = \sum_{j=1}^{J}\sum_{k=1}^{J} \ell(r_j, k)\,\mathbb{E}_{\psi}[S_{j,k}]. \tag{3}$$

The entry-wise marginal distributions necessary to describe this expectation form a doubly-stochastic matrix (DSM). A DSM is a nonnegative square matrix in which each column and each row sum to one. Permutation matrices are special cases of doubly-stochastic matrices. By Birkhoff's Theorem, every doubly stochastic matrix can be expressed as the convex combination of at most J^2-2J+2 permutations (see, e.g., [7]). It is natural, then, to think of a DSM as a relaxation of a permutation that incorporates precisely the uncertainty that is appropriate for rank-linear gains. That is, the jth row of a DSM provides a properly-normalized marginal distribution over the rank of item j. As the columns must also sum to one, this set of J distributions is consistent in the sense that they cannot, e.g., attempt to place all documents in the first position. While the DSM does not indicate which permutations have non-zero probability, it does provide all the information that is necessary to determine the expected gain under rank-linear objectives.

We leverage this interpretation of DSMs as providing a set of consistent marginal rank distributions in the expected ranking objective. Let $\boldsymbol{\Pi} \in [0,1]^{J\times J}$ be a doubly stochastic matrix. We interpret entry $\Pi_{j,k}$ as the marginal probability that item j is at rank k. We use this to define a differentiable objective function

$$\mathbb{E}_{\boldsymbol{\Pi}}[\mathcal{L}_{\mathsf{NDCG@}K}(\boldsymbol{r})] = \frac{\sum_{j=1}^{J}\sum_{k=1}^{K}\Pi_{j,k}\,\mathcal{G}(r_j)\,\mathcal{D}(k)}{\max_{s'}\mathcal{L}_{\mathsf{DCG@}K}(s'\,;\,\boldsymbol{r})} \tag{4}$$

that is the expectation of NDCG@K under an (unknown) distribution over permutations that has the marginals described by $\boldsymbol{\Pi}$. The expected P@K can be defined similarly:

$$\mathbb{E}_{\boldsymbol{\Pi}}[\mathcal{L}_{\mathsf{P@}K}(\boldsymbol{r})] = \frac{1}{K}\sum_{j=1}^{J} r_j \sum_{k=1}^{K}\Pi_{j,k},$$

as can the expectation of rank biased precision:

$$\mathbb{E}_{\boldsymbol{\Pi}}[\mathcal{L}_{\mathsf{RBP}}(\boldsymbol{r})] = (1-\alpha)\sum_{j=1}^{J} r_j \sum_{k=1}^{J}\Pi_{j,k}\alpha^{k-1}.$$

Note that all three of these expectations recover their original counterparts when $\mathbf{\Pi}$ is a permutation matrix, which could then be thought of as providing a set of degenerate marginals.

2.3. Choosing a Single Permutation

Although we have defined differentiable rank-linear objectives in terms of doubly-stochastic matrices, our test-time task remains the same: we must produce a single ordering. Under our "consistent marginals" interpretation of doubly-stochastic matrices, the natural objective is the one the maximizes the log likelihood:

$$s^\star = \operatorname*{argmax}_{s \in \mathcal{S}_J} \sum_{k=1}^{J} \ln \Pi_{s[k],k}. \tag{5}$$

This maximization is a bipartite matching problem and can be solved in $\mathcal{O}(J^3)$ time using the Hungarian algorithm [8]. For queries of more than a few hundred documents, this cubic time complexity becomes a bottleneck. To speed up selection of a permutation, we use a "short-cut" bipartite matching scheme that uses a quadratic algorithm to compute a global ordering and then uses the Hungarian algorithm on only the top $P < J$ documents under the global ordering, resulting in $\mathcal{O}(K^2 + P^3)$ running time.

In the first pass, we compute the expected rank of each document under the marginal distributions implied by the doubly-stochastic matrix:

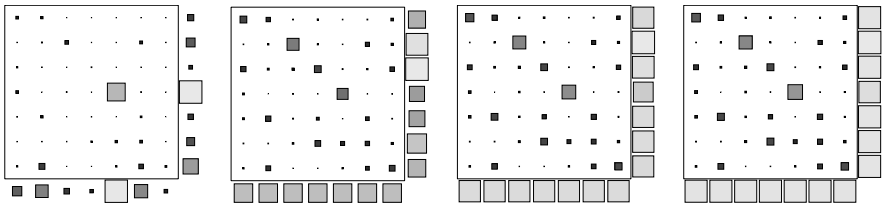
$$\mathbb{E}_{\mathbf{\Pi}}[\text{rank}_j] = \sum_{k'=1}^{K} \Pi_{j,k'} k'.$$

These expected ranks can be sorted to compute an ordering $\hat{s}$ for all J documents. In the second phase, the Hungarian algorithm is applied to the top P documents and the submatrix $\mathbf{\Pi}_{\hat{s}[1:P],1:P}$ using the score in Eq. (5). This provides an improved ordering for the top P documents, while keeping the remainder of the permutation fixed.

While this procedure offers no theoretical guarantees for the optimality of the global matching, it is well-suited to the ranking problem. First, metrics such as NDCG are most heavily influenced by the top-ranked documents in the ordering; focusing the expensive computations on this subset is a sensible heuristic. Second, the main way that this procedure would act pathologically is if the row-wise distributions were highly multimodal, i.e., significant mass split between very high and very low ranks. For the types of functions we explore in Section 4, however, this kind of behavior is unlikely.

3. Learning to Rank with Sinkhorn Gradients

Having relaxed our objective functions from permutations to doubly-stochastic matrices, we are no longer seeking to learn functions $f_J : \mathcal{X}^J \to \mathcal{S}_J$ (which output permutations), but instead a family of functions $g_J : \mathcal{X}^J \to \mathcal{W}_J$,



(a) Initial matrix (b) First iteration (c) Second iteration (d) Third iteration

Fig 1: Hinton diagrams of three iterations of Sinkhorn normalization. The row and column sums are shown as squares on the right and bottom of the box. The matrix quickly balances: in (d) the row- and column-wise sums are almost indistinguishable.

where $\mathcal{W}_J$ is the set of $J \times J$ doubly stochastic matrices (the Birkhoff polytope). Such functions are difficult to construct, however, as it is not straightforward to parameterize doubly-stochastic matrices. This is in contrast to, for example, right-stochastic matrices for which we can easily construct a surjective function using a row-wise softmax. There is, however, an iterative projection procedure known as a *Sinkhorn normalization* [9], which takes a nonnegative square matrix and converts it to a doubly-stochastic matrix by repeatedly normalizing rows and columns. More formally, we define row and column normalization functions:

$$\mathcal{T}_{\mathsf{R}}(\boldsymbol{A}) = \boldsymbol{A} \oslash (\boldsymbol{A}\mathbf{1}\mathbf{1}^{\mathsf{T}}) \qquad \mathcal{T}_{\mathsf{C}}(\boldsymbol{A}) = \boldsymbol{A} \oslash (\mathbf{1}\mathbf{1}^{\mathsf{T}}\boldsymbol{A}),$$

where $\oslash$ is the Hadamard (elementwise) division and $\mathbf{1}$ is a vector of ones. We can then define the iterative function

$$\mathcal{Z}^i(\boldsymbol{A}) = \begin{cases} \boldsymbol{A} & \text{if } i = 0 \\ \mathcal{T}_{\mathsf{R}}(\mathcal{T}_{\mathsf{C}}(\mathcal{Z}^{i-1}(\boldsymbol{A}))) & \text{otherwise} \end{cases}.$$

The function $\mathcal{Z}^\infty : \mathbb{R}_+^{J\times J} \to \mathcal{W}_J$ is a Sinkhorn normalization operator and, when it converges, it produces a doubly-stochastic matrix.

Sinkhorn and Knopp [10] presented necessary and sufficient conditions for convergence of this procedure to a unique limit. In summary, convergence is guaranteed for most non-negative matrices, but some non-negative matrices cannot be converted into doubly stochastic matrices because of their pattern of zeros (expressed as conditions on zero minors in the original matrix). $\mathcal{O}(V|\log\epsilon|)$ Sinkhorn steps suffice to reach ϵ-near double stochasticity if all the matrix entries are in $[1, V]$ [11]. Hence most applications of Sinkhorn normalization smooth the original matrix to facilitate convergence.

In this paper, we introduce the idea of an *incomplete* Sinkhorn normalization, which are the functions $\mathcal{Z}^i(\boldsymbol{A})$, for $i < \infty$. The incomplete Sinkhorn normalization allows us to define the objective functions of the previous section in terms of square matrices whose only constraints are that the entries must be nonnegative. That is, if we can now produce a matrix $\mathbb{R}_+^{J\times J}$ for each training query, this can be approximately normalized and the ranking functions can be evaluated. Most interestingly, however, it is possible to

compute the gradient of the training objective with regard to the initial unnormalized matrix. This can be done efficiently by propagating the gradient backward through the sequence of row and column normalizations, as in discriminative neural network training. We refer to this backpropgation as *SinkProp*.

As in backpropagation, in SinkProp we proceed through layers of computation and we assume that the output of each layer provides the input to a scalar function for which we are computing the gradient in terms of the layer's inputs. This function is here denoted $\mathcal{U}(\boldsymbol{A}')$ and we assume that the gradient $\partial\mathcal{U}(\boldsymbol{A}')/\partial\boldsymbol{A}'$ has already been computed. We can use this to compute the gradient of a row normalization via

$$\frac{\partial}{\partial A_{j,k}}\mathcal{U}(\mathcal{T}_{\mathsf{R}}(\boldsymbol{A})) = \sum_{k'=1}^{J}\left[\frac{\delta_{k,k'}}{\sum_{k''=1}^{J} A_{j,k''}} - \frac{A_{j,k'}}{\left(\sum_{k''=1}^{J} A_{j,k''}\right)^2}\right]\frac{\partial\mathcal{U}(\boldsymbol{A}')}{\partial A'_{j,k'}}.$$

The gradient of the column normalization is essentially identical, modulo appropriate transposition of indices. The function $\mathcal{U}(\cdot)$ here corresponds to the composition of any number of Sinkhorn normalizations with the rank-linear objective. This enables us to use a small amount of code to backpropagate through an arbitrary-depth incomplete Sinkhorn normalization.

4. Parameterizing the Pre-Sinkhorn Matrix

We have so far defined a differentiable training function that allows us to optimize a rank-linear objective in terms of an unconstrained nonnegative square matrix. The final piece of our framework is to define a family of functions $h_J : \mathcal{X}^J \to \mathbb{R}_+^{J\times J}$, for $J \in \{1, 2, \ldots\}$. There are several approaches that could be taken to this problem. We will focus on two examples in which the functions h_J can be computed from J evaluations of a single function $\phi : \mathcal{X} \to \mathbb{R}^D$.

4.1. Partitioned Probability Measures

One approach to the functions h_J is to construct them in terms of parametric probability densities $\pi(u \,|\, \theta)$ defined on $(0, 1)$. The parameters θ are taken to be in $\mathbb{R}^D$ for all J, so that they may be computed via $\phi(\boldsymbol{x})$. For any J, we can define J equally-spaced bins in $(0, 1)$, with edges $\rho_J[j] = j/J$, for $j \in \{0, 1, \ldots, J\}$. We then define the "output matrix" from h_J to be

$$A_{j,k} = \int_{\rho_J[k-1]}^{\rho_J[k]} \pi(u \,|\, \theta_j = \phi(\boldsymbol{x}_j))\,\mathrm{d}u. \tag{6}$$

In this construction, each row of the matrix arises by subdividing the mass of the row-specific cumulative probability distribution function whose parameterization is determined by $\phi(\boldsymbol{x}_j)$. Intuitively, this means that for any given row, the more mass that appears near zero, the greater the preference

for that document appearing higher in the ranking. Reasonable choices for $\pi(\cdot)$ include the beta distribution, the probit, and the logit-logistic (LL):

$$\pi_{\mathsf{LL}}(u \,|\, \theta = \{\mu, \sigma\}) = \frac{1}{4\sigma}\mathrm{sech}^2\left\{\frac{\ln(u/(1-u)) - \mu}{2\sigma}\right\}\left(\frac{1}{u} + \frac{1}{1-u}\right).$$

Both the PDF and CDF of the LL distribution are fast to compute. This overall construction is appealing as it is differentiable, and is a natural approach to regression of variable-sized square matrices.

4.2. Smoothed Indicator Functions

One deficiency of the approach of the previous section is that there is no interaction between the documents except through the Sinkhorn iterations. An alternative approach, inspired by SmoothRank [6], is to construct the pre-Sinkhorn matrix via:

$$A_{j,k} = \exp\left\{-\frac{1}{2\sigma^2}\left(\phi(\boldsymbol{x}_j) - \phi(\boldsymbol{x}_{\hat{s}[k]})\right)^2\right\}, \tag{7}$$

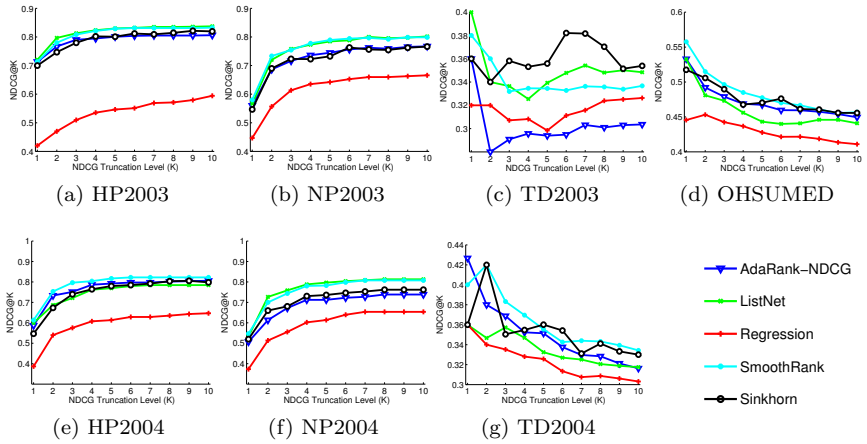
where $D{=}1$ and the permutation $\hat{s}$ arises from sorting the $\phi(\boldsymbol{x}_j)$. In the limit of $\sigma \to 0$ this recovers the permutation matrix implied by $\hat{s}$. In the limit $\sigma \to \infty$, the matrix becomes all ones. As discussed in [6], this function is continuous but not differentiable. However, it is almost everywhere differentiable, as the discontinuities in the first derivatives only occur at ties between the $\phi(\boldsymbol{x}_j)$. This has no practical effect for optimization purposes.

In both of these examples, it is the function $\phi(\cdot)$ that is of primary interest, and whose parameters are being optimized during training. Assuming that the document features are M-dimensional real-valued vectors, i.e., $\mathcal{X} = \mathbb{R}^M$, then a simple linear function is a reasonable base case: $\phi(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}$, where $\boldsymbol{W} \in \mathbb{R}^{M \times D}$. More interesting is the possibility of using deep neural networks and other more sophisticated function approximation tools.

5. Empirical Evaluation on LETOR Data

In this section we report on comparisons with other approaches to ranking, using seven data sets associated with LETOR 3.0 [12] benchmark. Following the standard LETOR procedures, we trained over five folds, each with distinct training, validation and test splits. We used the training objective of Eq. (4), with K set to be the number of documents in the largest query. This achieved the best performance in our experiments, similar to the results reported in [6]. These experiments used the smoothed indicator function approach described in Section 4.2. Optimization was performed using L-BFGS[1], annealing the smoothing constant σ as in [6]. We initialized with the MLE regression weights under squared loss. Early stopping was performed based on the NDCG of predictions on the validation set. We regularized the weights using the ℓ_2 distance to the MLE regression weights, selecting the

[1] `http://www.cs.ubc.ca/~schmidtm/Software/minFunc.html`



(a) HP2003 (b) NP2003 (c) TD2003 (d) OHSUMED

(e) HP2004 (f) NP2004 (g) TD2004

Fig 2: Results on the seven LETOR 3.0 data sets.

regularization penalty using the validation set. To reduce training time, the training data were resampled into a larger number of smaller queries. Each initial query was turned into twenty derived queries whose documents were sampled with replacement from the original. The number of documents in each derived query was Poisson distributed, with mean determined by the original number of documents, up to a maximum of 200. Before performing the Sinkhorn normalization, a small constant ($\approx 10^{-6}$) was added to each entry in the matrix. Five Sinkhorn iterations were performed. When specific rank predictions were made at test time, the short-cut Hungarian method was used as described in Section 2.3, with $P = 200$.

The results for the seven different data sets are shown in Figure 2. Several publicly available baselines[2] are shown for comparison: AdaRank [13], ListNet [14], SmoothRank [6] and basic regression. In each figure, the testing NDCG score is shown as a function of truncation level. As can be seen in the graphs, the Sinkhorn approach is generally competitive with the state-of-the-art. On the TD2003, the Sinkhorn normalization appears to offer a substantial advantage.

6. Related Work

SinkProp builds on a rapidly expanding set of approaches to rank learning. Early learning-to-rank methods employed surrogate gain functions, as approximations to the target evaluation measure were necessary due to its aforementioned non-differentiability. More recently, methods have been developed to optimize expectation of the target evaluation measure, including SoftRank [4]. BoltzRank [15] and SmoothRank [6]. These methods all attempt to maximize the expected gain of the ranking under any of the gain functions described above. The crucial component of each is the estimate of the distribution over rankings: SoftRank uses a rank-binomial

[2] `http://research.microsoft.com/en-us/um/beijing/projects/letor/`

approximation, which entails sampling and sorting ranks; BoltzRank uses a fixed set of sampled ranks; and SmoothRank uses a softmax on rankings based on a noisy model of scores. SinkProp can be viewed as another method to optimize expected ranking gain, but the effect of the scaling is to concentrate the mass of the distribution over ranks on a small set, which peaks on the single chosen rank selected by the model at test time.

Sinkhorn scaling itself is a long-standing method with a wide variety of applications, including discrete constraint satisfaction problems such as Sudoku [16] and for updating probabilistic belief matrices [17]. It has also been used as a method for finding or approximating the matrix permanent, as the permanent of the scaled matrix is the permanent of original matrix times the entries of the row and column scaling vector [18]. Recently, Sinkhorn normalization has been employed within a regret-minimization approach for on-line learning of permutations [19].

Although this work represents the first approach to ranking that has directly incorporated Sinkhorn normalization into the training procedure, previously-developed methods have also found it useful. The SoftRank algorithm [4], for example, uses Sinkhorn balancing at test-time, as a post-processing step for the approximated ranking matrix. Unlike the approach proposed here, however, it does not take this step into account when optimizing the objective function. SmoothRank uses half a step of Sinkhorn balancing, normalizing only the scores within each column of the matrix, to produce a distribution over items at a particular rank.

7. Discussion and Future Work

In this paper we have presented a new way to optimize learning algorithms under ranking objectives. The conceptual centerpiece of our approach is the observation that the expectations of certain kinds of popular ranking objectives — ones we have dubbed *rank-linear* — can be evaluated exactly even if only marginal distributions are available. This means that the expected value of these ranking objectives can be computed from the doubly-stochastic matrix that arises from their location within the Birkhoff polytope. To actually learn the appropriate doubly-stochastic matrices, it is possible to apply a well-studied iterative projection operator known as a Sinkhorn normalization. Remarkably, gradient-based learning can still be efficiently performed in the unnormalized space by backpropagating through the iterative procedure, which we call *SinkProp*. We demonstrated the effectiveness of this new approach by applying it to seven information retrieval datasets.

There are several promising future directions for work in this area. In the context of ranking, the ability to backpropagate gradients of rank-linear objectives enables wide possibilities in retrieval of non-text documents. In particular, there have been significant recent advances in gradient-based training of neural networks for discrimination of images and speech (e.g., [20, 21]). The SinkProp approach could enable these networks to be trained to produce permutations over these more general types of objects. More broadly, we have shown that it is practical to backpropagate training gradients

through iterative projection operators. Such operators can be defined for a variety of structured outputs: matching problems and image correspondence tasks, for example. Finally, while the notion of rank-linearity is specific to the problem of learning permutations, it seems likely that it can be expanded to other types of structured-prediction tasks, leading to efficient computation of expected gains under DSM-like representations.

Acknowledgements

The authors wish to thank Sreangsu Acharyya, Daniel Tarlow and Ilya Sutskever for valuable discussions. RPA is a junior fellow of the Canadian Institute for Advanced Research.

References

[1] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20:422–446, October 2002.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Information Retrieval*. Addison-Wesley, 1999.

[3] Alistair Moffat and Justin Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems*, 27:2:1–2:27, December 2008.

[4] Michael J. Taylor, John Guiver, Stephen Robertson, and Tom Minka. SoftRank: Optimizing non-smooth rank metrics. In Marc Najork, Andrei Z. Broder, and Soumen Chakrabarti, editors, *Proceedings of the International Conference on Web Search and Data Mining*, pages 77–86, 2008.

[5] Quoc V. Le and Alex J. Smola. Direct optimization for ranking measures. Technical report, NICTA, 2007. `http://arxiv.org/abs/0704.3359`.

[6] O. Chapelle and M. Wu. Gradient descent optimization of smoothed information retrieval metrics. *Information Retrieval*, 13(3):216–235, 2010.

[7] T. E. S. Raghavan R. B. Bapat. *Nonnegative Matrices and Applications*. Cambridge Univ Press, 1997.

[8] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[9] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *Annals of Mathematical Statistics*, 35:876–879, 1964.

[10] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21:343–348, 1967.

[11] Philip A. Knight. The Sinkhorn-Knopp algorithm: Convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30:1:261–275, 2008.

[12] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 13:346–374, August 2010.

[13] Jun Xu and Hang Li. AdaRank: A boosting algorithm for information retrieval. In *Proceedings of the 30th Annual ACM SIGIR Conference on Research and Development on Information Retrieval*, pages 391–398, 2007.

[14] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136, 2007.

[15] M. N. Volkovs and R. S. Zemel. BoltzRank: Learning to maximize expected ranking gain. In *International Conference on Machine Learning*, pages 1089–1096, 2009.

[16] T. K. Moon, J. H. Gunther, and J. J. Kupin. Sinkhorn solves Sudoku. *IEEE Transactions on Information Theory*, 55:1741–1746, 2009.

[17] Hamsa Balakrishnan, Inseok Hwang, and Claire J. Tomlin. Polynomial approximation algorithms for belief matrix maintenance in identity management. In *In 43rd IEEE Conference on Decision and Control*, pages 4874–4879, 2004.

[18] M. Huber and J. Law. Fast approximation of the permanent for very dense problems. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 681–689, 2008.

[19] David P. Helmbold and Manfred K. Warmuth. Learning permutations with exponential weights. *Journal of Machine Learning Research*, 10:1705–1736, 2009.

[20] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning*, 2009.

[21] Abdel rahman Mohamed, George E. Dahl, and Geoffrey E. Hinton. Deep belief networks for phone recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.