Deep Spectral Ranking

İlkay Yıldız * Jennifer Dy * Deniz Erdoğmuş * Susan Ostmo †

J. Peter Campbell † Michael F. Chiang †† Stratis Ioannidis *

* ECE Dept., Northeastern Univ., Boston, MA, USA, {yildizi, jdy, erdogmus, ioannidis}@ece.neu.edu
† Casey Eye Inst., Oregon Health and Science Univ., Portland, OR, USA, {ostmo, campbelp}@ohsu.edu
†† National Eye Inst., National Institutes of Health, Bethesda, MD, USA, {chiangm}@ohsu.edu

Abstract

Learning from ranking observations arises in many domains, and siamese deep neural networks have shown excellent inference performance in this setting. However, SGD does not scale well, as an epoch grows exponentially with the ranking observation size. We show that a spectral algorithm can be combined with deep learning methods to significantly accelerate training. We combine a spectral estimate of Plackett-Luce ranking scores with a deep model via the Alternating Directions Method of Multipliers with a Kullback-Leibler proximal penalty. Compared to a state-of-the-art siamese network, our algorithms are up to 175 times faster and attain better predictions by up to 26% Top-1 Accuracy and 6% Kendall-Tau correlation over five real-life ranking datasets.

1 Introduction

Learning from ranking observations arises in many domains, including, e.g., econometrics (McFadden 1973, Ryzin and Mahajan 1999), psychometrics (Thurstone 1927, Bradley and Terry 1952), sports (Elo 1978), and medicine (Tian et al. 2019), to name a few. Ranking observations are (potentially noisy and incomplete) orderings of subsets of samples. Given a dataset of such ranking observations, probabilistic inference typically assumes the existence of an underlying total ordering and aims to recover it. The Plackett Luce model (Plackett 1975) is a prominent parametric model in this setting: it postulates that (a) each sample is parametrized by a score and (b) the probability that a sample is ranked higher than a set of alternatives is proportional to this score.

Ranking regression assumes that Plackett-Luce scores are a parametric function of sample features, regressed from ranking observations. Maximum-Likelihood Estimation (MLE) in this setting has received significant attention in the literature, via both shallow (Joachims 2002, Pahikkala et al. 2009, Guo et al. 2018, Tian et al. 2019) and deep neural network (DNN) models (Burges et al. 2005, Chang et al. 2016, Dubey et al. 2016, Han 2018, Yıldız et al. 2019). Siamese neural networks (Bromley et al. 1994) perform extremely well in this setting: for example, Yıldız et al. (2019) train a 5.9M parameter siamese neural network from pairwise comparisons on just 80 images, attaining 0.92 AUC.

Figure 1: Training time and Top-1 prediction accuracy (indicated next to each marker) of DSR-KL, DSR-l2, and siamese network vs. query size (K) on the Movehub-Cost dataset partitioned w.r.t. rank partitioning. Siamese network training grows exponentially with K; both our algorithm, DSR-KL, and the one by Yıldız et al. (2020) scale more gracefully w.r.t. K. However, DSR-KL has a considerably higher accuracy, comparable to (or better than) the one attained by the less efficient siamese network.
Deep Spectral Ranking

An efficient algorithm for ranking regression over shallow models was recently proposed by Yıldız et al. (2020). Building on earlier work by Maystre and Grossgauser (2015), Yıldız et al. (2020) employ the Alternating Directions Method of Multipliers (ADMM) (Boyd et al., 2011) to separate the learning of regression model parameters from sample scores. On one hand, this separation allows them to express the MLE of scores as the stationary distribution of a continuous-time Markov-Chain, and learn scores from an $O(n^K)$-sized dataset using a fast spectral method. On the other hand, the separation allows regressing the linear model parameters from the scores, which are $O(n)$. This separation yields significant performance improvements; their algorithm is up to 579 times faster than traditional optimization methods for MLE, including, e.g., Newton’s method (Boyd and Vandenberghe, 2004).

Unfortunately, this approach does not generalize well to deep models. Particularly, Yıldız et al. (2020) regress scores by minimizing a squared-error proximal penalty, resulting in a regression sub-problem. It is well-known that in positive-valued regression, the small bounded range of the squared loss makes DNN training prone to vanishing gradients and reaching stationary points far from local minima (Caruana and Niculescu-Mizil, 2004; Golik et al., 2013; Bosman et al., 2020). To address these issues, we make the following contributions:

- We replace the standard $\ell^2$-norm proximal penalty in ADMM with Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951). This leads to training the DNN regressor with max-entropy loss and an additional linear term at each ADMM iteration.
- We prove that this new penalty is still amenable to fast inference via a spectral approach, generalizing Yıldız et al. (2020). Our resulting algorithm, DSR-KL, accelerates ranking regression problem for DNN models by incorporating this spectral solver. To the best of our knowledge, we are the first in bridging the gap between state-of-the-art DNN models and efficient spectral algorithms for ranking regression.
- We show experimentally that our method significantly outperforms standard siamese networks and the $\ell_2$-penalty ADMM used by Yıldız et al. (2020).

These performance benefits are illustrated in Figure 1. Siamese network training grows exponentially with $K$; both our algorithm, DSR-KL, and the one by Yıldız et al. (2020) scale more gracefully. However, DSR-KL has a considerably higher accuracy, comparable to the one attained by (the less efficient) siamese network.

2 Related Work

Ranking regression via the Plackett-Luce (PL) (Plackett, 1975) model has a long history, mostly focusing on pairwise comparisons, i.e., restricted to $K = 2$; this case is also known as the Bradley-Terry (BT) model (Bradley and Terry, 1952). Among shallow regressors, RankSVM (Joachims, 2002) learns a target ranking from features via a linear Support Vector Machine (SVM), with constraints imposed by all possible comparisons. Pahikkala et al. (2009) propose a regularized least-squares based algorithm for learning to rank from comparisons. Several works (Joachims, 2002; Pahikkala et al., 2009; Guo et al., 2018; Tian et al., 2019) regress comparisons via maximum-likelihood estimation (MLE) over BT (Bradley and Terry, 1952) models.

Deeper models for regressing comparisons have also been extensively explored in the BT setting (Burges et al., 2005; Chang et al., 2016; Dubey et al., 2016; Doughty et al., 2018; Yıldız et al., 2019). Burges et al. (2005) estimate comparisons via a fully connected neural network called RankNet, using the BT model to construct a cross-entropy loss. Several works (Chang et al., 2016; Dubey et al., 2016; Doughty et al., 2018; Yıldız et al., 2019) learn from comparisons via siamese networks (Bromley et al., 1994). Chang et al. (2016) and Yıldız et al. (2019) learn from comparisons using MLE on the BT model. Dubey et al. (2016) predict image comparisons, via a loss function combining cross-entropy and hinge loss. Doughty et al. (2018) learn similarities and comparisons of videos via hinge loss. All of the above methods generalize to rankings under the PL model, but suffer from the complexity issues...
outlined in the introduction.

Ye et al. (2018, 2019; Zhao and Liao, 2019) focus on learning from star/relevance ratings rather than rankings; they train a neural network called ListNet that treats ratings as Plackett-Luce scores. Particularly, given a dataset of n ratings, ListNet constructs all \( \binom{n}{K} \) possible K-sized rankings; it is then trained by minimizing the cross entropy between the Plackett-Luce probabilities that a given K-ranking comprises the top samples among all n, with scores being (a) the predicted scores and (b) the ground-truth ratings, respectively. Hence, although ListNet solves a very different problem than ranking regression, it is related to the siamese methods mentioned above through its (exponential in \( K \)) objective.

Virtually all shallow and deep models for ranking regression rely on classic optimization methods for direct parameter inference, resulting in prohibitively slow training for large rankings. An efficient algorithm for ranking regression over shallow models was recently proposed by Yıldız et al. (2020). Yıldız et al. (2020) solve the ranking regression problem via Alternating Directions Method of Multipliers (ADMM) (Boyd et al. 2011) with an \( \ell_2 \) proximal penalty. This allows them to express the MLE of scores as the stationary distribution of a Markov-Chain (MC), extending the approach developed by Maystre and Grossglauser (2015) in the feature-less setting. Our approach extends Yıldız et al. (2020) by generalizing (i) the regression model to DNNs, and (ii) the proximal penalty of ADMM to Kullback-Leibler (KL) divergence (Kullback and Leibler 1951); this has significant performance implications (c.f. Sec. 5). To the best of our knowledge, we are the first to accelerate ranking regression of deep models via a spectral method.

ADMM is typically used when training deep neural networks to enforce sparsity. One application is compressed sensing, in which a signal is recovered from undersampled measurements via a DNN transformation (Sun et al. 2016; Yang et al. 2020; Li et al. 2018; Ma et al. 2019; Liu et al. 2018). The training objective combines a standard regression loss with a norm penalty on measurements, including, e.g., \( \ell_1 \), \( \ell_2 \), Frobenius, or nuclear norm. Another application is model pruning (Ye et al. 2018, 2019; Zhao and Liao 2019): ADMM is used to enforce weight sparsity by replacing norm penalties with (possibly non-convex) low-cardinality set constraints. Zhao et al. (2018) train a DNN classifier to recover samples distorted by additive adversarial noise, using an objective that combines a classification loss with an \( \ell_0 \), \( \ell_1 \), \( \ell_2 \), or \( \ell_\infty \) norm penalty.

We use ADMM with a Kullback-Leibler (KL) divergence proximal penalty as an alternative to \( \ell_2 \). Wang and Banerjee (2014) show that ADMM with a Bregman divergence proximal penalty converges with linear rate for convex objectives. Wang et al. (2018) extend this convergence guarantee to local optima of non-convex problems. Several works (Shi et al. 2017; Yu et al. 2018; Yu and Açıkgöz 2019) employ a Bregman ADMM algorithm for distributed optimization of separable problems.

3 Problem Formulation

Plackett-Luce Model. We introduce here the Plackett-Luce discriminative model that we use in our analysis; before describing it for ranking observations, we first consider the simpler “maximal choice” setting. Consider a dataset of \( n \) samples, indexed by \( i \in [n] \equiv \{1, \ldots, n\} \). Every sample \( i \in [n] \) has a corresponding \( d \)-dimensional feature vector \( x_i \in \mathbb{R}^d \). There exists an underlying total ordering of these \( n \) samples. A labeler of this dataset acts as a (possibly noisy) oracle revealing this total ordering: when presented with a query \( A \subseteq [n] \), i.e., a set of alternatives, the noisy labeler chooses the maximal sample in \( A \) w.r.t. the underlying total ordering. Formally, our “labeled” dataset \( D = \{(c_\ell, A_\ell)\}_{\ell=1}^m \) consists of \( m \) observations \( (c_\ell, A_\ell) \), \( \ell \in [m] \equiv \{1, \ldots, m\} \), where \( A_\ell \subseteq [n] \) is the \( \ell \)-th query submitted to the labeler and \( c_\ell \in A_\ell \) is her respective \( \ell \)-th maximal choice (i.e., the label). For every sample \( i \in [n] \), we denote by \( W_i = \{\ell \in [m] \mid i \in A_\ell, c_\ell = i\} \) the set of observations where sample \( i \in [n] \) is chosen, and by \( L_i = \{\ell \in [m] \mid i \in A_\ell, c_\ell \neq i\} \) the set of observations where sample \( i \in [n] \) is not chosen.

The Plackett-Luce model asserts that every sample \( i \in [n] \) is associated with a non-negative deterministic score \( \pi_i \in \mathbb{R}_+ \). Given \( \pi = [\pi_i]_{i \in [n]} \in \mathbb{R}^n_+ \): (i) observations \( (c_\ell, A_\ell) \), \( \ell \in [m] \) are independent, and (ii) for a query \( A_\ell \), the probability that sample \( c_\ell \in A_\ell \) is chosen is:

\[
P(c_\ell \mid A_\ell, \pi) = \frac{\pi_{c_\ell}}{\sum_{j \in A_\ell} \pi_{j}} = \frac{\pi_{\ell}}{\sum_{j \in A_\ell} \pi_{j}}, \quad (1)
\]

where, in the last equation, we abuse notation to write the score of the chosen sample as \( \pi_{c_\ell} \equiv \pi_{c_\ell} \). Note that \( P(c_\ell \mid A_\ell, \pi) = P(c_\ell \mid A_\ell, s\pi) \), for all \( s > 0 \); thus, w.l.o.g., we assume (or enforce via rescaling) that Plackett-Luce scores satisfy \( 1^\top \pi = 1 \), i.e., \( \pi \) is a distribution over \([n]\).

Plackett-Luce readily extends to datasets of (partial) ranking observations. In this setting, when presented with a query \( A_\ell \subseteq [n] \) of \( K = |A_\ell| \) samples, the labeler ranks the samples in \( A_\ell \) into an ordered sequence \( \alpha_1^\ell > \cdots > \alpha_K^\ell \). Under the Plackett-Luce model, this ranking is expressed as \( K - 1 \) maximal choice queries:
\( \alpha_1^t \) over \( A_t \), \( \alpha_2^t \) over \( A_t \setminus \{ \alpha_1^t \} \), etc., so that:

\[
P(\alpha_1^t \succ \alpha_2^t \ldots \succ \alpha_K^t | A_t, \pi) = \prod_{t=1}^{K-1} \left( \pi_{\alpha_t^t} / \sum_{v=t}^{K} \pi_{\alpha_v^t} \right). \tag{2}\]

The product form of (2) implies that a ranking observation in response to a query \( A_t \) can be converted to \( K - 1 \) independent maximal-choice observations (again, \( \alpha_1^t \) over \( A_t \), \( \alpha_2^t \) over \( A_t \setminus \{ \alpha_1^t \} \), and so forth), each governed by \( \{1\} \), that yield the same joint probability. MLE over a dataset of ranking observations thus reduces to MLE over a dataset of maximal choice observations. For notational simplicity, we present our analysis over maximal-choice datasets in Sec. 4, keeping the above reduction in mind.

**Parameter Inference and Regression.** Given observations \( D \), Maximum Likelihood Estimation (MLE) of the Plackett-Luce scores \( \pi \in \mathbb{R}_+^n \) amounts to minimizing the negative log-likelihood:

\[
\mathcal{L}(D | \pi) \equiv \sum_{\ell=1}^{m} \left( \log \sum_{j \in A_\ell} \pi_j - \log \pi_\ell \right). \tag{3}\]

To regress scores \( \pi \) from sample features \( X = [x_1, \ldots, x_n]^T \in \mathbb{R}^{n \times d} \), we assume that there exists a function \( \tilde{\pi} : \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow [0, 1] \) (e.g., a DNN), parametrized by \( W \in \mathbb{R}^{d'} \), such that:

\[
\pi_i = \tilde{\pi}(x_i; W), \quad \text{for all } i \in [n]. \tag{4}\]

Then, MLE amounts to minimizing the following objective w.r.t. \( W \in \mathbb{R}^{d'} :\)

\[
\mathcal{L}(D | W) \equiv \sum_{\ell=1}^{m} \left( \log \sum_{j \in A_\ell} \tilde{\pi}(x_j; W) - \log \tilde{\pi}(x_\ell; W) \right). \tag{5}\]

In the case of deep models, virtually all state-of-the-art ranking regression methods minimize Eq. (5) via SGD [Chang et al., 2016; Dubey et al., 2016; Doughty et al., 2018; Yıldız et al., 2019]. Note that the objective (5) corresponds to a siamese network architecture (Bromley et al., 1994) with base network \( \tilde{\pi} \); for each observation \( c_\ell, A_\ell \) in \( D \), the base network needs to be evaluated (and back-propagation needs to happen over) all samples in \( A_\ell \). Hence, if each query \( A_\ell \) has size \( K = |A_\ell| \), the siamese network contains \( K \) identical base networks, each receiving the feature vector \( x_i \) of a sample \( i \in A_\ell \); this is further exacerbated in ranking queries, since each corresponds to \( K - 1 \) maximal choice queries. Overall, the siamese nature of objective (5) implies that \( K \) samples need to be loaded in RAM to process a single ranking or maximal choice query. This, in addition to the fact that the size of possible rankings/maximal queries in \( D \) grows as \( n^K \) = \( O(n^K) \) (c.f. Figures 1 and 2c), can make the cost of a single SGD epoch over Eq. (5) prohibitive.

**4 Deep Spectral Ranking Algorithm.**

We wish to devise an efficient algorithm to minimize (5). To do so, we extend the Alternating Directions Method of Multipliers (ADMM) [Boyd et al., 2011] used by Yıldız et al. (2020). To this end, we rewrite the minimization of (5) as:

\[
\begin{align*}
\text{Minimize } & \mathcal{L}(D | \pi) \equiv \sum_{\ell=1}^{m} \left( \log \sum_{j \in A_\ell} \pi_j - \log \pi_\ell \right) \tag{6a} \\
\text{subject to: } & \pi = \tilde{\pi}(X; W), \quad \pi \geq 0, \tag{6b}
\end{align*}
\]

where \( \pi \in \mathbb{R}_+^n \) is the Lagrangian dual variable corresponding to the equality constraint in (6b), \( \rho \geq 0 \) is a penalty parameter, and \( D_p(\cdot | \cdot) \) is a proximal penalty term. This term satisfies: (i) \( D_p(\pi | \tilde{\pi}) \geq 0 \) for all \( \pi, \tilde{\pi} \in \mathbb{R}_+^n \), and (ii) \( D_p(\pi | \pi) = 0 \) if and only if \( \pi = \tilde{\pi} \). Classic ADMM involves using the usual \( \ell_2 \) proximal penalty, i.e., \( D_p(\pi | \tilde{\pi}) = \| \pi - \tilde{\pi} \|_2^2 \). We depart from this by considering more generic \( D_p \); as we discuss below, using KL-divergence instead, i.e., \( D_p(\pi | \tilde{\pi}) = \sum_{i=1}^{m} \pi_i \log \frac{\pi_i}{\tilde{\pi}_i} \) has significant advantages in our setting.

**Decoupling Optimization and a Spectral Algorithm.** In its general form, ADMM alternates between optimizing \( \pi \) and \( W \) until convergence via the following primal-dual algorithm on the augmented Lagrangian:

\[
\begin{align*}
\pi^{k+1} & = \arg \min_{\pi \in \mathbb{R}_+^n} \mathcal{L}(D | \pi) + y^\top (\pi - \tilde{\pi}(X; W)) \tag{8a} \\
W^{k+1} & = \arg \min_{W \in \mathbb{R}^{d'}} D_p(\pi^{k+1} | \tilde{\pi}(X; W)), \tag{8b}
\end{align*}
\]

This has the following immediate computational advantages. First, step (8b) is equivalent to:

\[
W^{k+1} = \arg \min_{W \in \mathbb{R}^{d'}} \rho D_p(\pi^{k+1} | \tilde{\pi}(X; W)) - y^{k+1} \top \tilde{\pi}(X; W) \tag{9}
\]

Note that this operation does not depend on \( D \); the model \( \tilde{\pi}(X; W) \) is regressed directly from the present score estimates \( \pi^{k+1} \) via penalty \( D_p \), with the additional linear dual term. In particular, as discussed below, \( D_p \) leads to a least squares regression in the case of the \( \ell_2 \)-proximal penalty, and a max-entropy penalty in the case of KL-divergence; both can be
solved efficiently via SGD. Crucially, an epoch of this SGD iterates over the $O(n)$ samples, instead of the $O(n^K)$ ranking observations, which would be the case under the siamese approach.

Most importantly, step (8a)–which does depend on $D$–admits a highly efficient spectral implementation; ADMM [3] therefore delegates solving the “expensive” portion of the problem to a highly efficient algorithm. This is a consequence of the following theorem, which we prove in Appendix A.

**Theorem 4.1.** Given $\pi^k \equiv \pi(X; W^k) \in \mathbb{R}^n$ and $y^k \in \mathbb{R}^n$, a stationary point $\pi = \pi^k$ of the Augmented Lagrangian (L) satisfies the balance equations of a continuous-time Markov Chain (MC) with transition rates:

$$
\mu_{ji}(\pi) = \begin{cases} 
\lambda_{ji}(\pi) + \sum_{\ell \in [n]} 2\sigma_i(\pi)\sigma_j(\pi) - \sum_{\ell \in [n]} \sigma_j(\pi), & \text{if } j \in [n]_+ \text{ and } i \in [n]_-
n\lambda_{ji}(\pi), & \text{otherwise},
\end{cases}
$$

(10)

where

$$
\sigma_i(\pi) = \frac{\partial D_p(\pi \| \hat{\pi}^k)}{\partial \pi_i} + y^k_i, \text{ for } i \in [n] (11)
$$

$$
\lambda_{ji}(\pi) = \sum_{\ell \in W \cap L_j \cap A_i} (\sum_{i \in \ell} \sigma_i - 1) \geq 0, \text{ for } i, j \in [n], (12)
$$

and $([n]_+, [n]_-)$ is a partition of $[n]$ such that $\sigma_i(\pi) \geq 0$ for all $i \in [n]_+$ and $\sigma_i(\pi) < 0$ for all $i \in [n]_-$. Theorem 4.1 generalizes Theorem 4.2 of Yildiz et al. (2020), which holds only for an $\ell_2$ penalty for $D_p$; Maystre and Grossglauser (2015) had established a similar result for stationary points of $L(D|\pi)$. Our theorem implies that a stationary point can be obtained through the following iterative algorithm, called ILSRX by Yildiz et al. (2020). Let $sad(M)$ be the stationary distribution of an MC with transition matrix $M = [\mu_{ji}(\pi)]_{i,j \in [n]}$, where $\mu_{ji}(\pi)$ are given by (10). When matrix $M$ is fixed (i.e., the transition rates are known), the vector $sad(M)$ is a solution to the linear system defined by the balance equations $\pi = M\pi$ and $\mathbf{1}^\top \pi = 1$, as it is a distribution. However, the transition matrix $M = M(\pi)$ in Theorem 4.1 is itself a function of $\pi$, and is therefore a priori unknown. Thus, following Yildiz et al. (2020); Maystre and Grossglauser (2015), we can find $\pi$ through:

$$
\pi^{q+1} = sad(M(\pi^q)), \text{ for } q = 0, 1, 2, \ldots (13)
$$

We note that the dataset $D$ appears in the computation of the rate matrix $M$, via sets $W_i$ and $L_j$; the computation of these weights can be easily parallelized. The linear system determined by the balance equations involves $n$ unknowns and can be computed efficiently by uniformizing $M$, i.e., increasing self-transition rates until all states have the same outgoing rate, and finding the leading left eigenvector via, e.g., the power method (Lei et al. 2016). Yildiz et al. (2020) focus on the case where (i) Plackett-Luce scores are affine functions of sample features, i.e., $\pi(X; W) = WX$, and (ii) $D_p(\pi \| \hat{\pi}) = \| \pi - \hat{\pi} \|^2_2$. The generalization to KL-divergence is naturally suited to Problem (6); this is because the scores computed by (13) form, by construction, a distribution over $[n]$.

Our main technical contribution is to show that this generalization to KL penalty is still amenable to a spectral solution via (13); in practice, this also leads to a significantly improved performance over an $\ell_2$-proximal penalty (by up to 56% Top-1 accuracy and 25% Kendall-Tau correlation, as discussed in Sec. 5). As shown in Eq. (15) below, the KL-divergence penalty leads to training $\pi(X; W)$ with a max-entropy loss and an additional linear term at each ADMM iteration. Compared to the $\ell_2$-proximal penalty by Yildiz et al. (2020), max-entropy has been observed to converge faster and lead to better fitting (Caruana and Niculescu-Mizil 2004; Golik et al. 2013, Bosman et al. 2020). Finally, ADMM with Bregman divergence proximal penalties, including KL divergence, has been shown to offer local convergence guarantees for non-convex problems (Wang et al. 2018); this further motivates us to employ KL over other distance measures between probability distributions.

**Overall Algorithm.** Putting everything together, our Deep Spectral Ranking (DSR) algorithm solving Eq. (6) is summarized in Algorithm 1. We initialize all samples with equal scores, i.e., $\pi = \frac{1}{n}\mathbf{1}$. We iteratively update $\pi$, $W$, and $y$ via Eq. (5) until convergence. At each ADMM iteration $k$, we initialize $\pi$ with $\pi^{k-1}$, and update $\pi$ via ILSRX given by Eq. (13). Then, we initialize the DNN parameters $W$ with $W^x$, and fine tune the DNN $\pi(X; W)$ via SGD over Eq. (9). Each iteration of DSR involves the three steps in Eq. (3). One iteration of ILSRX at step (8a) is $O(Km+n^2)$ for constructing the transition rates via Eq. (10) and for finding the stationary distribution $\pi$ via, e.g., a power method, respectively. The update of $y$ given $\pi$ and $\pi = O(n)$. Finally, constructing the loss function (9) at each epoch of step (8b) is $O(n)$, while each epoch goes over $O(n)$ samples of dimension $d$ to train the $d$ weights.

We implement DSR with the two proximal penalty functions $D_p(\pi \| \hat{\pi})$ mentioned above: Kullback-Leibler (KL) divergence (Kullback and Leibler 1951) and $\ell_2$ norm. We describe implementation specifics for each of these cases below.

**KL proximal penalty.** For $D_p(\pi \| \hat{\pi}) = \sum_{i=1}^n \pi_i \log \frac{\pi_i}{\hat{\pi}_i}$,
Table 1: No. of samples ($n$), no. of features ($d$), no. of observations ($m$), and query size ($K$) for datasets with image or numerical features ($X$).

we have:

$$
\sigma_i(\pi) = \rho \frac{\partial D_p(\pi \| \tilde{\pi})}{\partial \pi_i} + y_i^k = \rho(1 + \log \frac{\pi_i}{\tilde{\pi}_i}) + y_i^k,
$$

for all $i \in [n]$ in the transition rates \cite{10}. Moreover, the optimization \cite{9} for training $\tilde{\pi}(X; W)$ becomes:

$$
\arg \min_{W \in \mathbb{R}^d} \sum_{i=1}^{n} \left(-\frac{y_i^k}{\rho} \log \frac{\pi_i}{\tilde{\pi}_i} - \pi_i^{k+1} \log \tilde{\pi}_i \right),
$$

which corresponds to max-entropy with an additional linear term.

**$\ell_2$ proximal penalty.** For $D_p(\pi \| \tilde{\pi}) = \| \tilde{\pi} - \pi \|_2^2$:

$$
\sigma_i(\pi) = \rho \frac{\partial D_p(\pi \| \tilde{\pi})}{\partial \pi_i} + y_i^k = 2\rho(\pi_i - \tilde{\pi}_i^k) + y_i^k,
$$

for all $i \in [n]$ in the transition rates \cite{10}. Moreover, \cite{9} becomes a least squares minimization of the form:

$$
\arg \min_{W \in \mathbb{R}^d} \| \tilde{\pi}(X; W) - (\pi^{k+1} + \frac{1}{\rho} y^k) \|_2^2.
$$

5 Experiments

5.1 Experiment Setup

We evaluate DSR on real-life datasets summarized in Table 1 additional details are given in App. B. We partition each dataset into training and test sets in two ways. In rank partitioning, we partition the dataset w.r.t. [m], using 90% of the m observations for training, and the remaining 10% for testing. In sample partitioning, we partition [n], using 90% of the n samples for training, and the remaining 10% for testing. In this setting, observations containing samples from both training and validation/test sets are discarded. We perform 3-fold cross validation on the resulting training sets.

We implement five competing algorithms. DSR with KL penalty (DSR-KL), DSR with $\ell_2$ norm penalty (DSR-$\ell_2$), and siamese network competitor regress scores via a DNN $\tilde{\pi}(X; W)$ with parameters $W$. SR-$\ell_2$ \cite{11} and SR-KL use an affine regressor $\pi = X\beta + b$ with parameters $\beta$ and $b$, and solve \cite{6} via ADMM with $\ell_2$ and KL penalties, respectively.

We execute all experiments on NVIDIA V100 GPUs with Intel Gold 6132@2.60GHz CPUs and 128GB RAM. We explain the network architecture and the training procedure of $\tilde{\pi}(X; W)$ in Section 5.2 and the implementation details of all algorithms, including, e.g. the convergence criteria in Section 5.3. To evaluate the convergence speed of each algorithm, we measure the elapsed time, including time spent in initialization, in seconds (Time). Moreover, we measure the prediction performance by Top-1 accuracy (Top-1 Acc.) and Kendall-Tau correlation (KT) \cite{12} on validation and test sets (c.f. Section 5.4). We report averages and standard deviations over folds for all algorithms except for the siamese network method: as training takes several hours, we execute only one fold.

5.2 Network Architecture and Training

To evaluate each method on ROP, we choose $\tilde{\pi}(X; W)$ as a state-of-the-art convolutional neural network architecture, namely GoogleLeNet \cite{13}, followed by a fully connected output layer comprising a single neuron with sigmoid activation. We initialize the convolutional layers with weights pre-trained on the ImageNet dataset \cite{14}. For other datasets, we design $\tilde{\pi}(X; W)$ as a fully-connected architecture with relu activation for hidden layers and an output layer comprising a single neuron with sigmoid activation. We add $\ell_2$ regularizers to all layers. For each configuration of (i) $\ell_2$ regularization parameter varying in $[2 \times 10^{-5}, 2 \times 10^{-2}]$, (ii) learning rate varying in $[10^{-4}, 10^{-2}]$, and (iii) number of layers varying in $[1, 10]$ for fully connected $\tilde{\pi}(X; W)$, we run each method until convergence (c.f. Section 5.3 for criteria). We determine the best set of hyperparameters w.r.t. the prediction performance via 3-fold cross validation.

1Our code is publicly available at \url{https://github.com/neu-spiral/DeepSpectralRanking}
5.3 Algorithms

DSR-KL and DSR-l2 are explained in Section 4 and summarized in Algorithm 1. We compute the stationary distribution at each iteration of ILSRX (c.f. Eq. (13)) using the power method (Lei et al., 2016). At each ADMM iteration, we fine tune the DNN regressor \( \hat{\pi}(X; W) \) via Adam optimization (Kingma and Ba [2015]) over Eq. (3). We check the convergence of \( L \) on the training set and Kendall-Tau correlation (KT) evaluations on the validation set (c.f. Section 5.4) as the stopping criterion for: (i) fine-tuning \( \hat{\pi}(X; W) \) at each ADMM iteration, and (ii) overall DSR-KL algorithm. We declare convergence when KT on validation set does not change for 5 iterations, for maximum total of 50 iterations, with relative tolerance \( r_{\text{tol}} = 10^{-4} \). We use the same relative tolerance for the stopping criterion of the power method.

To aid convergence in practice (Boyd et al. [2011]), we update the dual variable at each ADMM iteration with a multiplicative smoothing parameter \( \gamma^k = 1/k \). We also adapt the penalty parameter as:

\[
\rho^{k+1} = \begin{cases} 
\tau \rho^k, & \text{if } \|\hat{\pi}^k - \pi^k\|_2 > \beta \|\hat{\pi}^k - \hat{\pi}^{k-1}\|_2 \\
\rho^k, & \text{otherwise,}
\end{cases}
\]

where \( \tau = 2 \) and \( \beta = 10 \).

The siamese network competitor minimizes Eq. (3) w.r.t. \( W \) via SGD. We train a siamese network architecture with base network \( \hat{\pi} \) on observations \( D \) via Adam optimization (Kingma and Ba [2015]) over Eq. (3). We employ the same convergence criterion and experiment setup as DSR-KL and DSR-l2 to train and optimize the siamese network.

Finally, we implement two spectral algorithms that regress Plackett-scores via an affine model, i.e., \( \hat{\pi} = X\beta + b1 \): SR-l2 proposed by Yıldız et al. [2020], and its variant SR-KL that uses KL proximal penalty instead of \( \ell_2 \) norm penalty for ADMM. As the stopping criterion for both algorithms, we use \( \|\hat{\pi}^k - \pi^k\|_2 < r_{\text{tol}} \|\hat{\pi}^k\|_2 \) and \( \|(X\beta^k + b^k1) - (X\beta^{k-1} + b^{k-1}1)\|_2 < r_{\text{tol}} \|X\beta^k + b^k1\|_2 \). Following Yıldız et al. [2020], we set the ADMM penalty parameter as \( \rho = 1 \).

5.4 Evaluation Metrics

Let the test set be \( D_{\text{rank}} = \{(\alpha^\ell, A_\ell) | \ell \in \{1, ..., m_{\text{test}}\}\} \), where \( \alpha^\ell = \alpha_1^\ell \succ \alpha_2^\ell \succ \cdots \succ \alpha_K^\ell \) is an ordered sequence of the samples in \( A_\ell \). Given \( A_\ell \), we predict the \( \ell \)-th choice as \( \hat{c}_\ell = \arg \max_{i \in A_\ell} \hat{\pi}_i \). We calculate the Top-1 accuracy (Top-1 Acc.) as:

\[
\text{Top-1 Acc.} = \frac{\sum_{\ell=1}^{m_{\text{test}}} \mathbbm{1}(\hat{c}_\ell = \alpha_1^\ell)}{m_{\text{test}}} \in [0, 1].
\]  

We also predict the ranking as \( \hat{\alpha}^\ell = \arg \max_i [\hat{\pi}_i]_{i \in A_\ell} \), i.e. sequence of the samples in \( A_\ell \) ordered w.r.t. their estimated scores. We calculate Kendall-tau correlation (KT) (Kendall, 1938) as a measure of the correlation between each true ranking \( \alpha^\ell \) and predicted ranking \( \hat{\alpha}^\ell \), \( \ell \in \{1, ..., m_{\text{test}}\} \). For observation \( \ell \), let \( T_\ell = \sum_{i=1}^{K} \sum_{s=i}^{K} \mathbbm{1}(\hat{\alpha}_i^\ell > \alpha_s^\ell \land \alpha_i^\ell > \alpha_s^\ell) \) be the number correctly predicted ranking positions, and \( F_\ell = \sum_{i=1}^{K} \sum_{s=i}^{K} \mathbbm{1}(\hat{\alpha}_i^\ell > \alpha_s^\ell \land \alpha_i^\ell > \alpha_s^\ell) \) be the number incorrectly predicted ranking positions. Then, KT is computed by:

\[
KT = \frac{\sum_{\ell=1}^{m_{\text{test}}} (T_\ell - F_\ell) / \left( \binom{K}{2} \right)}{m_{\text{test}}} \in [-1, 1],
\]  

where \( \binom{K}{2} \) is the number of sample pairs.

5.5 Results

Training Time vs. Prediction Performance. Figure 2a and 2b show the training time of DSR-KL and siamese network vs. Top-1 Acc. and KT on test sets of Movehub-Cost-4, Movehub-Quality-4, and IMDb-4 datasets, partitioned with rank partitioning. For all datasets, DSR-KL results lie on the top left region of both Top-1 Acc. and KT plots: DSR-KL consistently leads to much faster training and better predictions than the siamese counterpart.

Table 2 shows the training time and test set prediction performance of DSR-KL, DSR-l2, siamese network, SR-l2, and SR-KL trained on all datasets (c.f. Table 1), partitioned with rank partitioning. DSR-KL and DSR-l2 are 1.5 – 142 times faster than siamese network over all datasets. Moreover, DSR-KL consistently attains equivalent or better prediction performance than both siamese network and DSR-l2 w.r.t. both Top-1 Acc. and KT. DSR-KL leads to particularly better performance in ranking predictions, by up to 6% higher KT than siamese and 25% higher KT than DSR-l2 on IMDb-4.

Deeper regression methods consistently outperform the predictions of shallow regression methods. Particularly, our deep spectral algorithms DSR-KL and DSR-l2 lead to significantly better predictions than the shallow versions SR-l2 and SR-KL, up to 38% Top-1 Acc. and 41% KT on IMDb-4. The training times of DSR-KL and DSR-l2 are also not noticeably larger than the ones of shallow versions; SR-KL converges even slower than DSR-KL, by up to 12 times on Movehub-Quality-5. Unlike SR-l2 that solves a least-squares problem with closed form solution, parameter update step of SR-KL (c.f. 15) requires an iterative optimization at each ADMM iteration.

Table 3 shows the training time and test set prediction performance of DSR-KL, DSR-l2, siamese network, SR-l2, and SR-KL trained on all datasets, partitioned
Figure 2: (a)-(b) Training time of DSR-KL and siamese network vs. Top-1 Acc. and KT on test sets of Movehub-Cost-4, Movehub-Quality-4, and IMDB-4 datasets, partitioned w.r.t. rank partitioning. (c) Training time and prediction performances of DSR-KL, DSR-I2, and siamese network vs. query size ($K$) on Movehub-Quality dataset partitioned w.r.t. rank partitioning. Top-1 accuracy for each $K$ is next to the corresponding marker.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Method</th>
<th>Time (s)</th>
<th>Performance on the Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>↓</td>
<td>Top-1 Acc. ↑</td>
</tr>
<tr>
<td>ICLR-3</td>
<td>DSR-KL</td>
<td>152.86 ± 29.98</td>
<td>0.9 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>DSR-I2</td>
<td>165.59 ± 22.63</td>
<td>0.79 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>Siamese</td>
<td>1445.77</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>SR-KL</td>
<td>529.02 ± 117.26</td>
<td>0.37 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>SR-I2</td>
<td>20.59 ± 3.02</td>
<td>0.87 ± 0.0</td>
</tr>
<tr>
<td>Movehub-Cost-4</td>
<td>DSR-KL</td>
<td>49.54 ± 34.2</td>
<td>0.88 ± 0.07</td>
</tr>
<tr>
<td></td>
<td>DSR-I2</td>
<td>73.38 ± 32.82</td>
<td>0.72 ± 0.02</td>
</tr>
<tr>
<td></td>
<td>Siamese</td>
<td>523.64</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>SR-KL</td>
<td>29.64 ± 3.59</td>
<td>0.47 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>SR-I2</td>
<td>19.7 ± 0.72</td>
<td>0.8 ± 0.0</td>
</tr>
<tr>
<td>Movehub-Quality-4</td>
<td>DSR-KL</td>
<td>63.05 ± 4.14</td>
<td>0.88 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>DSR-I2</td>
<td>551.86 ± 212.91</td>
<td>0.82 ± 0.07</td>
</tr>
<tr>
<td></td>
<td>Siamese</td>
<td>710.35</td>
<td>0.88</td>
</tr>
<tr>
<td></td>
<td>SR-KL</td>
<td>98.02 ± 3.73</td>
<td>0.17 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>SR-I2</td>
<td>18.98 ± 4.11</td>
<td>0.84 ± 0.0</td>
</tr>
<tr>
<td>IMDB-4</td>
<td>DSR-KL</td>
<td>23.93 ± 16.15</td>
<td>0.9 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>DSR-I2</td>
<td>54.16 ± 13.22</td>
<td>0.78 ± 0.03</td>
</tr>
<tr>
<td></td>
<td>Siamese</td>
<td>3409.03</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>SR-KL</td>
<td>57.93 ± 0.87</td>
<td>0.16 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>SR-I2</td>
<td>9.43 ± 1.02</td>
<td>0.52 ± 0.0</td>
</tr>
<tr>
<td>ICLR-4</td>
<td>DSR-KL</td>
<td>84.93 ± 1.76</td>
<td>0.88 ± 0.01</td>
</tr>
<tr>
<td></td>
<td>DSR-I2</td>
<td>84.78 ± 1.21</td>
<td>0.63 ± 0.01</td>
</tr>
<tr>
<td></td>
<td>Siamese</td>
<td>632.18</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>SR-KL</td>
<td>347.29 ± 39.6</td>
<td>0.29 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>SR-I2</td>
<td>563.11 ± 40.33</td>
<td>0.86 ± 0.0</td>
</tr>
<tr>
<td>ROP</td>
<td>DSR-KL</td>
<td>776.71 ± 136.74</td>
<td>0.89 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>DSR-I2</td>
<td>694.00 ± 431.12</td>
<td>0.84 ± 0.03</td>
</tr>
<tr>
<td></td>
<td>Siamese</td>
<td>1152.06</td>
<td>0.86</td>
</tr>
<tr>
<td></td>
<td>SR-KL</td>
<td>610.91 ± 20.69</td>
<td>0.5 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>SR-I2</td>
<td>3.08 ± 0.59</td>
<td>0.89 ± 0.00</td>
</tr>
<tr>
<td>Movehub-Cost-5</td>
<td>DSR-KL</td>
<td>183.6 ± 48.2</td>
<td>0.85 ± 0.047</td>
</tr>
<tr>
<td></td>
<td>DSR-I2</td>
<td>216.5 ± 64.34</td>
<td>0.81 ± 0.04</td>
</tr>
<tr>
<td></td>
<td>Siamese</td>
<td>7986.22</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>SR-KL</td>
<td>189.85 ± 0.72</td>
<td>0.45 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>SR-I2</td>
<td>280.95 ± 1.5</td>
<td>0.78 ± 0.0</td>
</tr>
<tr>
<td>Movehub-Quality-5</td>
<td>DSR-KL</td>
<td>79.85 ± 2.91</td>
<td>0.85 ± 0.05</td>
</tr>
<tr>
<td></td>
<td>DSR-I2</td>
<td>91.87 ± 6.31</td>
<td>0.62 ± 0.04</td>
</tr>
<tr>
<td></td>
<td>Siamese</td>
<td>2241.49</td>
<td>0.80</td>
</tr>
<tr>
<td></td>
<td>SR-KL</td>
<td>924.07 ± 0.2</td>
<td>0.13 ± 0.0</td>
</tr>
<tr>
<td></td>
<td>SR-I2</td>
<td>208.1 ± 0.1</td>
<td>0.84 ± 0.0</td>
</tr>
</tbody>
</table>

Table 2: Rank Partitioning

Training time vs. Top-1 Acc. and KT on test sets of all datasets (c.f. Table 1), partitioned w.r.t. rank and sample partitioning, respectively. We report averages and standard deviations over folds for all algorithms except for siamese. Our algorithms, as well as the algorithm that attains the best performance for each dataset are indicated in bold.

Table 3: Sample Partitioning
with sample partitioning. Note that this is the setting when regressing scores from features is essential, as training samples cannot participate in any observations in validation or test sets. Agreeing with the speed gain in rank partitioning, DSR-KL and DSR-l2 are 8 – 175 times faster than the siamese network over all datasets. Moreover, DSR-KL leads to particularly better performance in maximal-choice predictions, by up to 26% higher Top-1 Acc. than siamese on IMDB-4 and 56% higher Top-1 Acc. than DSR-l2 on Movehub-Cost-4. Finally, deeper regression methods DSR-KL, DSR-l2, and siamese network, outperform the predictions of shallow counterparts SR-l2 and SR-KL, by up to 39% Top-1 Acc. and 11% KT on Movehub-cost-5 and ICLR-3, respectively.

**Impact of K.** Figures 1 and 2c show training time and prediction performances of DSR-KL, DSR-l2, and siamese network vs. query size (K) on Movehub-Cost and Movehub-Quality datasets partitioned w.r.t. rank partitioning. The speed gain of DSR-KL and DSR-l2 over siamese reach up to 43 times as K increases, as each epoch of siamese grows exponentially with K. Moreover, agreeing with Table 2, DSR-KL consistently outperforms the predictions of DSR-l2, validating the extension of ADMM penalty to KL divergence.

**Details on Convergence.** Figure 3 shows the log-likelihood $-\mathcal{L}$ on training and validation sets of Movehub-Cost-4 and Movehub-Quality-4 datasets, partitioned w.r.t. rank partitioning. Each point for DSR-KL and DSR-l2 correspond to an overall iteration of ADMM (cf. (8)), while each point for siamese corresponds to a training epoch. DSR-KL consistently attains higher log-likelihood and better validation performance than both siamese network and DSR-l2, while converging faster.

**Comparison to Naïve Approach.** A naïve spectral algorithm can be constructed by a single iteration of the primal steps in (8): (i) solving (8a) to learn $\pi$ via repeated iterations of (13), and (ii) given $\pi$, solving (8b) by training the DNN regressor $\tilde{\pi}(X; W)$ via SGD over Eq. (9). Intuitively, this ignores/does not exploit the fact that samples with similar features ought to have similar scores. We denote this naïve approach as One-iter.-DSR-KL. Unlike One-iter.-DSR-KL, our algorithm DSR-KL has the capability of repeatedly adapting both $\pi$ and $W$ by solving (8a) via ADMM. This advantage is illustrated in Figures 3a and 3b, where not only $\tilde{\pi}(X; W)$, but also $\pi$ are adjusted until convergence. Moreover, Figure 4 shows the corresponding test set predictions of DSR-KL vs. One-iter.-DSR-KL on Movehub-Cost-4 and Movehub-Quality-4; DSR-KL outperforms One-iter.-DSR-KL by 13-21% Top-1 Acc. and 6-15% KT.

### 6 Conclusion

We model Plackett-Luce scores as deep neural network (DNN) functions of sample features. We solve the maximum likelihood estimation problem for the scores via ADMM and demonstrate that the scores are equivalent to the stationary distribution of a Markov Chain. Our method significantly outperforms standard siamese networks and state-of-the-art spectral algorithms for ranking regression. Given that the number of rankings grows exponentially in query size, designing active learning algorithms to identify which rankings to solicit from labelers is an interesting open problem. Generalizing existing active learning algorithms for shallow models, e.g. [Guo et al. 2018], to deeper models via our efficient algorithms is a promising direction.
Acknowledgments

Our work is supported by NIH (R01EY019474), NSF (SCH-1622542 at MGH, SCH-1622536 at Northeastern, SCH-1622679 at OHSU), Facebook Statistics Research Award, and by unrestricted departmental funding from Research to Prevent Blindness (OHSU).

Bibliography


