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.

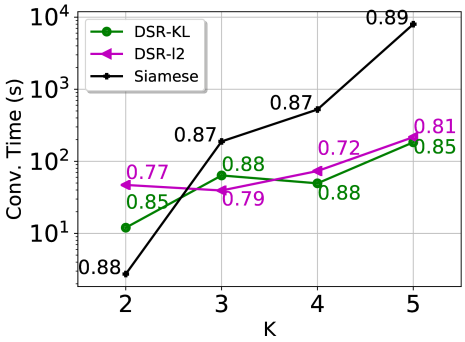


Figure 1: Training time and Top-1 prediction accuracy (indicated next to each marker) of DSR-KL, DSR-l2, and siamese network vs. query size (K) on the Movehub-Cost dataset partitioned w.r.t. rank partitioning. Siamese network training grows exponentially with K; both our algorithm, DSR-KL, and the one 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.

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.

A siamese network assumes that the score of each sample is determined by a base network: training over

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

ranking observations replicates this base network for every sample present in a ranking. However, siamese network training does not scale well with ranking size: the number of possible ranking observations is $O(n^K)$, where n is the number of samples and K is the ranking size. Datasets can span this regime, e.g., when multiple labelers generate rankings. For instance, consider a scenario where the n samples are papers submitted to a conference, and rankings are generated by a large pool of reviewers. In a conference with thousands of submissions, each reviewer is assigned $K \quad n$ papers to rank, and a global ranking is to be inferred by the (potentially $\Theta(n^K)$) dataset of partial rankings.

Consequently, stochastic gradient descent (SGD) epochs can grow exponentially with K. Moreover, the presence of K samples in each ranking observation increases the memory footprint of a batch; conversely, large values of K limit the size of batches that can be stored in memory. In particular, when $K = \Theta(n)$, SGD requires loading large–and potentially different–subsets of the dataset with each ranking; this further deteriorates performance. In practice, a single epoch over $K = 7$-way rankings takes 4.5 hours on an NVIDIA V100 GPU, even when samples have only 5 features.

An efficient algorithm for ranking regression over shallow models was recently proposed by Yıldız et al. (2020). Building on earlier work by Maystre and Grossglauser (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 ℓ_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 ℓ_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.

Cao et al. (2007); Xia et al. (2008) and Ma et al. (2020) 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 $\frac{n!}{(n-K)!}$ 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 ℓ_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., ℓ_1, ℓ_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 ℓ_0, ℓ_1, ℓ_2, or ℓ_∞ norm penalty.

We use ADMM with a Kullback-Leibler (KL) divergence proximal penalty as an alternative to ℓ_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 Açıkmeşe, 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 $i \in [n] \equiv \{1, \dots, n\}$. 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_ℓ, A_ℓ), $\ell \in [m] \equiv \{1, ..., m\}$, where $A_\ell \subseteq [n]$ is the ℓ-th *query* submitted to the labeler and $c_\ell \in A_\ell$ is her respective ℓ-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_ℓ, the probability that sample $c_\ell \in A_\ell$ is chosen is:

$$\mathbf{P}(c_\ell \mid A_\ell, \pi) = \pi_{c_\ell} / \textstyle\sum_{j\in A_\ell} \pi_j = \pi_\ell / \sum_{j\in A_\ell} \pi_j, \quad (1)$$

where, in the last equation, we abuse notation to write the score of the chosen sample as $\pi_\ell \equiv \pi_{c_\ell}$. Note that $\mathbf{P}(c_\ell \mid A_\ell, \pi) = \mathbf{P}(c_\ell \mid A_\ell, s\pi)$, for all $s > 0$; thus, w.l.o.g., we assume (or enforce via rescaling) that Plackett-Luce scores satisfy $\mathbf{1}^\top \pi = 1$, i.e., π 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_ℓ into an ordered sequence $\alpha_1^\ell \succ \alpha_2^\ell \succ \cdots \succ \alpha_K^\ell$. Under the Plackett-Luce model, this ranking is expressed as $K - 1$ maximal choice queries:

α_1^ℓ over A_ℓ, α_2^ℓ over $A_\ell \setminus \{\alpha_1^\ell\}$, etc., so that:

$$\mathbf{P}(\alpha_1^\ell \quad \alpha_2^\ell \qquad \alpha_K^\ell | A_\ell, \quad) = \prod_{t=1}^{K-1} \big(\pi_{\alpha_t^\ell} / \sum_{s=t}^{K} \pi_{\alpha_s^\ell}\big). \quad (2)$$

The product form of (2) implies that a ranking observation in response to a query A_ℓ can be converted to $K \quad 1$ *independent* maximal-choice observations (again, α_1^ℓ over A_ℓ, α_2^ℓ over $A_\ell \setminus \{\alpha_1^\ell\}$, 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 $\mathcal{D}$, Maximum Likelihood Estimation (MLE) of the Plackett-Luce scores $\quad \in \mathbb{R}_+^n$ amounts to minimizing the negative log-likelihood:

$$\mathcal{L}(\mathcal{D} | \quad) \quad \sum_{\ell=1}^{m} \big(\log \sum_{j \in A_\ell} \pi_j \quad \log \pi_\ell\big). \quad (3)$$

To regress scores $\quad$ from sample features $\mathbf{X} = [\mathbf{x}_1, .., \mathbf{x}_n]^T \in \mathbb{R}^{n \times d}$, we assume that there exists a function $\tilde{\pi} : \mathbb{R}^d \quad \mathbb{R}^{d'} \to [0, 1]$ (e.g., a DNN), parametrized by $\mathbf{W} \in \mathbb{R}^{d'}$, such that:

$$\pi_i = \tilde{\pi}(\mathbf{x}_i; \mathbf{W}), \quad \text{for all } i \in [n]. \quad (4)$$

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

$$\mathcal{L}\big(\mathcal{D} | \mathbf{W}\big) \quad \sum_{\ell=1}^{m} \big(\log \sum_{j \in A_\ell} \tilde{\pi}(\mathbf{x}_j; \mathbf{W}) \quad \log \tilde{\pi}(\mathbf{x}_\ell; \mathbf{W})\big). \quad (5)$$

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_ℓ, A_ℓ) in $\mathcal{D}$, the base network needs to be evaluated (and back-propagation needs to happen over) all samples in A_ℓ. Hence, if each query A_ℓ has size $K = |A_\ell|$, the siamese network contains K identical base networks, each receiving the feature vector $\mathbf{x}_i$ of a sample $i \in A_\ell$; this is further exacerbated in ranking queries, since each corresponds to $K \quad 1$ 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 $\mathcal{D}$ grows as $\binom{n}{K} = O(n^K)$ (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:

$$\underset{, \mathbf{W}}{\text{Minimize}} \quad \mathcal{L}(\mathcal{D} | \quad) \quad \sum_{\ell=1}^{m} \big(\log \sum_{j \in A_\ell} \pi_j \quad \log \pi_\ell\big) \quad (6a)$$

$$\text{subject to:} \qquad = \tilde{}(\mathbf{X}; \mathbf{W}), \qquad \mathbf{0}, \quad (6b)$$

where $\tilde{}(\mathbf{X}; \mathbf{W}) \in \mathbb{R}_+^n$ is the vector map whose coefficients $\tilde{\pi}_i \in \mathbb{R}_+$ are given by $\tilde{\pi}_i = \tilde{\pi}(\mathbf{x}_i; \mathbf{W})$. To solve (6) via ADMM, consider the following *augmented Lagrangian*:

$$L_\rho(\quad, \mathbf{W}, \mathbf{y}) = \mathcal{L}(\mathcal{D} | \quad) + \mathbf{y}^\top (\qquad \tilde{}(\mathbf{X}; \mathbf{W})) + \rho \quad D_p(\quad || \tilde{}(\mathbf{X}; \mathbf{W})), \quad (7)$$

where $\mathbf{y} \in \mathbb{R}^n$ is the Lagrangian dual variable corresponding to the equality constraint in (6b), $\rho \quad 0$ is a penalty parameter, and $D_p(\,||\,)$ is a proximal penalty term. This term satisfies: (i) $D_p(\,||\,\tilde{}) \quad 0$ for all $, \tilde{} \in \mathbb{R}_+^n$, and (ii) $D_p(\,||\,\tilde{}) = 0$ if and only if $\tilde{} = \quad$. Classic ADMM involves using the usual ℓ_2 proximal penalty, i.e., $D_p(\,||\,\tilde{}) = \|\tilde{} \qquad \|_2^2$. We depart from this by considering more generic D_p; as we discuss below, using KL-divergence instead, i.e., $D_p(\,||\,\tilde{}) = \sum_{i=1}^n \pi_i \log \frac{\pi_i}{\tilde{\pi}_i}$ has significant advantages in our setting.

Decoupling Optimization and a Spectral Algorithm. In its general form, ADMM alternates between optimizing $\quad$ and $\mathbf{W}$ until convergence via the following primal-dual algorithm on the augmented Lagrangian:

$$^{k+1} = \underset{\in \mathbb{R}_+^n}{\arg\min} L_\rho(\quad, \mathbf{W}^k, \mathbf{y}^k) \quad (8a)$$

$$\mathbf{W}^{k+1} = \underset{\mathbf{W} \in \mathbb{R}^{d'}}{\arg\min} L_\rho(\quad^{k+1}, \mathbf{W}, \mathbf{y}^k) \quad (8b)$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \rho(\quad^{k+1} \quad \tilde{}(\mathbf{X}; \mathbf{W}^{k+1})). \quad (8c)$$

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

$$\mathbf{W}^{k+1} = \underset{\mathbf{W} \in \mathbb{R}^{d'}}{\arg\min} \; \rho D_p\big(\quad^{k+1} || \tilde{}(\mathbf{X}; \mathbf{W})\big) \qquad \mathbf{y}^{k\top} \tilde{}(\mathbf{X}; \mathbf{W}). \quad (9)$$

Note that this operation *does not* depend on $\mathcal{D}$: the model $\tilde{}(\mathbf{X}; \mathbf{W})$ is regressed directly from the present score estimates $\quad^{k+1}$ 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 ℓ_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 $\boldsymbol{D}$–admits a highly efficient spectral implementation; ADMM (8) 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* $\tilde{}^k \quad \tilde{}(\boldsymbol{X};\boldsymbol{W}^k) \ 2 \ \mathsf{R}_+^n$ *and* $\boldsymbol{y}^k \ 2 \ \mathsf{R}^n$*, a stationary point* $\quad 2 \ \mathsf{R}_+^n$ *of the Augmented Lagrangian* (7) *satisfies the balance equations of a continuous-time Markov Chain (MC) with transition rates:*

$$\mu_{ji}(\ ) = \begin{cases} \lambda_{ji}(\ ) + \frac{2\pi_i\sigma_i(\ )\sigma_j(\ )}{\sum_{t\,2\,[n]} \pi_t\sigma_t(\ ) - \sum_{t\,2\,[n]_+} \pi_t\sigma_t(\ )} \\ \qquad \textit{if } j \ 2 \ [n]_+ \ \textit{ and } \ i \ 2 \ [n]_- \\ \lambda_{ji}(\ ) \quad \textit{otherwise,} \end{cases} \tag{10}$$

where

$$\sigma_i(\ ) = \rho\frac{\partial D_p(\ \ jj\tilde{}^k)}{\partial\pi_i} + y_i^k, \quad \textit{for } i \ 2 \ [n] \tag{11}$$

$$\lambda_{ji}(\ ) = \sum_{\ell\in W_i\cap L_j} \big(\sum_{t\in A_\ell} \pi_t\big)^{-1} \quad 0, \quad \textit{for } i,j \ 2 \ [n]\,, \tag{12}$$

and $([n]_+, [n]_-)$ *is a partition of* $[n]$ *such that* $\sigma_i(\ ) \quad 0$ *for all* $i \ 2 \ [n]_+$ *and* $\sigma_i(\ ) < 0$ *for all* $i \ 2 \ [n]_-$.

Theorem 4.1 generalizes Theorem 4.2 of Yıldız et al. (2020), which holds only for an ℓ_2 penalty for D_p; Maystre and Grossglauser (2015) had established a similar result for stationary points of $\boldsymbol{L}(\boldsymbol{D}j\ )$. Our theorem implies that a stationary point can be obtained through the following iterative algorithm, called ILSRX by Yıldız et al. (2020). Let $\mathtt{ssd}(\boldsymbol{M})$ be the stationary distribution of an MC with transition matrix $\boldsymbol{M} = [\mu_{ji}(\ )]_{i,j\in[n]}$, where $\mu_{ji}(\ )$ are given by (10). When matrix $\boldsymbol{M}$ is fixed (i.e., the transition rates are known), the vector $\mathtt{ssd}(\boldsymbol{M})$ is a solution to the linear system defined by the balance equations $= \boldsymbol{M}$ and $\mathbf{1}^\top = 1$, as it is a distribution. However, the transition matrix $\boldsymbol{M} = \boldsymbol{M}(\ )$ in Theorem 4.1 is itself a function of , and is therefore a priori unknown. Thus, following Yıldız et al. (2020); Maystre and Grossglauser (2015), we can find through:

$$^{q+1} = \mathtt{ssd}\,(\boldsymbol{M}(\ ^q))\,, \quad \text{for } q = 0,1,2,\ldots. \tag{13}$$

We note that the dataset $\boldsymbol{D}$ appears in the computation of the rate matrix $\boldsymbol{M}$, 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 $\boldsymbol{M}$, 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).

Yıldız et al. (2020) focus on the case where (i) Plackett-Luce scores are affine functions of sample features, i.e., $\tilde{}(\boldsymbol{X};\boldsymbol{W}) = \boldsymbol{X}\,\boldsymbol{W}$, and (ii) $D_p(\ \ jj\tilde{}) = k\tilde{} \quad k_2^2$. 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 ℓ_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 $\tilde{}(\boldsymbol{X};\boldsymbol{W})$ with a max-entropy loss and an additional linear term at each ADMM iteration. Compared to the ℓ_2-proximal penalty by Yıldız 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., $= \tilde{} = \frac{1}{n}\mathbf{1}$. We iteratively update , $\boldsymbol{W}$, and $\boldsymbol{y}$ via Eq. (8) until convergence. At each ADMM iteration k, we initialize with $^{k-1}$, and update via ILSRX given by Eq. (13). Then, we initialize the DNN parameters $\boldsymbol{W}$ with $\boldsymbol{W}^{k-1}$, and fine tune the DNN $\tilde{}(\boldsymbol{X};\boldsymbol{W})$ via SGD over Eq. (9). Each iteration of DSR involves the three steps in Eq. (8). 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 via, e.g., a power method, respectively. The update of $\boldsymbol{y}$ given $\tilde{}$ and is $O(n)$. 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 $D_p(\ \ jj\tilde{}\,)$ mentioned above: Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) and ℓ_2 norm. We describe implementation specifics for each of these cases below.

KL proximal penalty. For $D_p(\ \ jj\tilde{}\,) = \sum_{i=1}^n \pi_i \log\frac{\pi_i}{\pi_i}$,

Algorithm 1 DSR

procedure ADMM($X, \mathcal{D}, \rho$)
  Initialize $\equiv \tilde{} \leftarrow \frac{1}{n}\mathbf{1}$; $y \leftarrow \mathbf{0}$
  repeat
    $\leftarrow$ ILSRX(ρ, , $\tilde{}$, y)
    $W \leftarrow$ SGD over Eq. (9)
    $\tilde{} \leftarrow \tilde{}(X; W)$
    $y \leftarrow y + \rho(- \tilde{})$
  until convergence
return W,
end procedure

procedure ILSRX(ρ, , $\tilde{}$, y)
  repeat
    Calculate [$_i( )]_{i \in [n]}$ via Eq. (11)
    Calculate $M( ) = [\mu_{ji}( )]_{i,j \in [n]}$ via Eq. (10)
    $\leftarrow$ ssd($M( )$)
  until convergence
return
end procedure

Spec.	Dataset				
	ROP	ICLR-3/4	Movehub-Cost-4/5	Movehub-Quality-4/5	IMDB-4
K	2	3/4	4/5	4/5	4
n	100		50		
d	224 224	768	6	5	36
m	29,705	120,324/ 2,248,524	230,298/ 2,118,756	230,298/ 2,118,756	85,583
$\boldsymbol{X}$	image	numerical			

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 ($\boldsymbol{X}$)

we have:

$$\sigma_i( ) = \rho \frac{\partial D_p(\ jj\tilde{}^k)}{\partial \pi_i} + y_i^k = \rho(1 + \log \frac{\pi_i}{\tilde{\pi}_i^k}) + y_i^k, \quad (14)$$

for all i 2 $[n]$ in the transition rates (10). Moreover, the optimization (9) for training $\tilde{}(X; W)$ becomes:

$$\arg\min_{W \in \mathsf{R}^{d^0}} \sum_{i=1}^{n} (\quad \frac{y_i^k}{\rho} \tilde{\pi}_i \quad \pi_i^{k+1} \log \tilde{\pi}_i), \quad (15)$$

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

ℓ_2 *proximal penalty.* For $D_p(\ jj\tilde{}) = k\tilde{} \quad k_2^2$:

$$\sigma_i( ) = \rho \frac{\partial D_p(\ jj\tilde{}^k)}{\partial \pi_i} + y_i^k = 2\rho(\pi_i \quad \tilde{\pi}_i^k) + y_i^k, \quad (16)$$

for all i 2 $[n]$ in the transition rates (10). Moreover, (9) becomes a least squares minimization of the form:

$$\arg\min_{W \in \mathsf{R}^{d^0}} \ k\tilde{}(X; W) \quad (\quad ^{k+1} + \frac{1}{\rho} y^k) k_2^2 \, . \quad (17)$$

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[1] five competing algorithms. DSR with KL penalty (DSR-KL), DSR with ℓ_2 norm penalty (DSR-l2), and siamese network competitor regress scores via a DNN $\tilde{}(X; W)$ with parameters W. SR-l2 (Yıldız et al., 2020) and SR-KL use an affine regressor $\tilde{} = X \quad + b\mathbf{1}$ with parameters and b, and solve (6) via ADMM with ℓ_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) (Kendall, 1938) 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 GoogLeNet (Szegedy et al., 2015), 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 (Deng et al., 2009). 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 ℓ_2 regularizers to all layers. For each configuration of (i) ℓ_2 regularization parameter varying in $[2 \quad 10^{-5}, 2 \quad 10^{-2}]$, (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.

[1] Our code is publicly available at 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 $\tilde{\pi}(\boldsymbol{X}; \boldsymbol{W})$ via Adam optimization (Kingma and Ba, 2015) over Eq. (9). We check the convergence of $\boldsymbol{L}$ 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 $\tilde{\pi}(\boldsymbol{X}; \boldsymbol{W})$ 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 $r_{\texttt{tol}} = 10^{-4}$. 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:

$$\rho^{k+1} = \begin{cases} \tau\rho^k, & \text{if } \boldsymbol{k} \tilde{} \, {}^k \boldsymbol{k}_2^2 > \beta \, \boldsymbol{k} \tilde{} \, {}^k \quad {}^{\tilde{}\, k-1} \boldsymbol{k}_2^2 \\ \frac{\rho^k}{\tau}, & \text{if } \boldsymbol{k} \tilde{} \, {}^k \quad {}^k \boldsymbol{k}_2^2 < \beta \, \boldsymbol{k} \tilde{} \, {}^k \quad {}^{\tilde{}\, k-1} \boldsymbol{k}_2^2 \\ \rho^k, & \text{otherwise,} \end{cases}$$

where $\tau = 2$ and $\beta = 10$.

The siamese network competitor minimizes Eq. (5) w.r.t. $\boldsymbol{W}$ via SGD. We train a siamese network architecture with base network $\tilde{\pi}$ on observations $\boldsymbol{D}$ via Adam optimization (Kingma and Ba, 2015) over Eq. (5). 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., $\tilde{} = \boldsymbol{X} \quad + b\mathbf{1}$: SR-l2 proposed by Yıldız et al. (2020), and its variant SR-KL that uses KL proximal penalty instead of ℓ_2 norm penalty for ADMM. As the stopping criterion for both algorithms, we use $\boldsymbol{k} \quad {}^k \quad {}^{k-1} \boldsymbol{k}_2 < r_{\texttt{tol}} \, \boldsymbol{k} \quad {}^k \boldsymbol{k}_2$ and $\boldsymbol{k}(\boldsymbol{X} \quad {}^k + b^k \mathbf{1}) \quad (\boldsymbol{X} \quad {}^{k-1} + b^{k-1} \mathbf{1} \boldsymbol{k}_2 < r_{\texttt{tol}} \, \boldsymbol{k} \boldsymbol{X} \quad {}^k + b^k \mathbf{1} \boldsymbol{k}_2$. Following Yıldız et al. (2020), we set the ADMM penalty parameter as $\rho = 1$.

5.4 Evaluation Metrics

Let the test set be $\boldsymbol{D}_{\texttt{rank}} = \boldsymbol{f}(\alpha^\ell, A_\ell) \boldsymbol{j} \, \ell \quad \boldsymbol{2} \; \boldsymbol{f}1, ..., m_{\texttt{test}} \boldsymbol{g}\boldsymbol{g}$, where $\alpha^\ell = \alpha_1^\ell \quad \alpha_2^\ell \quad \alpha_K^\ell$ is an ordered sequence of the samples in A_ℓ. Given A_ℓ, we predict the ℓ-th choice as $\hat{c}_\ell = \arg\max_{i \in A_\ell} \tilde{\pi}_i$. We calculate the Top-1 accuracy (Top-1 Acc.) as:

$$\text{Top-1 Acc.} = \frac{\sum_{\ell=1}^{m_{\texttt{test}}} \mathbb{1}(\hat{c}_\ell = \alpha_1^\ell)}{m_{\texttt{test}}} \; \boldsymbol{2} \; [0, 1]. \qquad (18)$$

We also predict the ranking as $\hat{\alpha}^\ell = \arg\text{sort}[\tilde{\pi}_i]_{i \in A_\ell}$, i.e. sequence of the samples in A_ℓ 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 α^ℓ and predicted ranking $\hat{\alpha}^\ell$, $\ell \; \boldsymbol{2} \; \boldsymbol{f}1, ..., m_{\texttt{test}} \boldsymbol{g}$. For observation ℓ, let $T_\ell = \sum_{t=1}^K \sum_{s=1}^K \mathbb{1}(\hat{\alpha}_t^\ell \quad \hat{\alpha}_s^\ell \wedge \alpha_t^\ell \quad \alpha_s^\ell)$ be the number correctly predicted ranking positions, and $F_\ell = \sum_{t=1}^K \sum_{s=1}^K \mathbb{1}(\hat{\alpha}_t^\ell \quad \hat{\alpha}_s^\ell \wedge \alpha_s^\ell \quad \alpha_t^\ell)$ be the number incorrectly predicted ranking positions. Then, KT is computed by:

$$\text{KT} = \frac{\sum_{\ell=1}^{m_{\texttt{test}}} (T_\ell \quad F_\ell) / \binom{K}{2}}{m_{\texttt{test}}} \; \boldsymbol{2} \; [\quad 1, 1], \qquad (19)$$

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

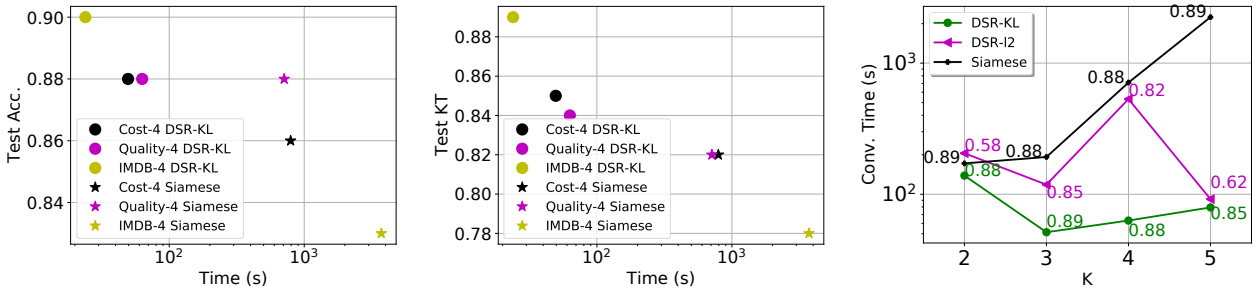
5.5 Results

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



(a) Training time vs. Top-1 Acc. (b) Training time vs. KT (c) Query size (K) vs. performance

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-l2, 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.

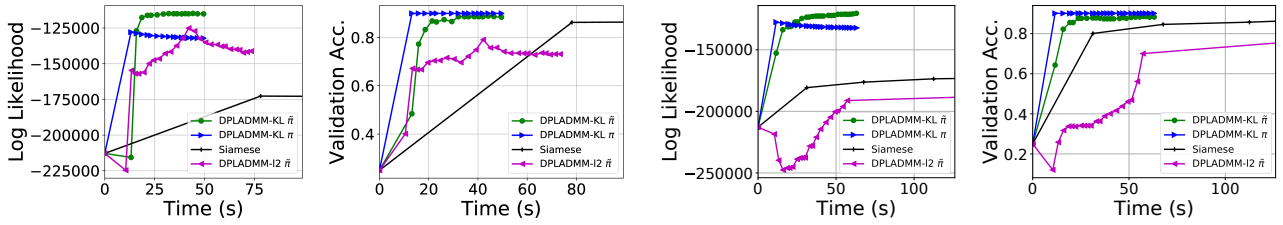
Dataset	Method	Time (s) #		Performance on the Test Set			
				Top-1 Acc. ″		KT ″	
ICLR-3	DSR-KL	152.86	29.98	0.9	0.0	0.86	0.0
	DSR-l2	165.59	22.63	0.79	0.0	0.5	0.0
	Siamese	1445.77		0.88		0.8	
	SR-KL	529.02	117.26	0.37	0.0	0.02	0.0
	SR-l2	20.59	3.02	0.87	0.0	0.82	0.0
Movehub-Cost-4	DSR-KL	49.54	34.2	0.88	0.07	0.85	0.09
	DSR-l2	73.38	32.82	0.72	0.02	0.58	0.03
	Siamese	523.64		0.87		0.82	
	SR-KL	29.64	3.59	0.47	0.0	0.27	0.0
	SR-l2	19.7	0.72	0.8	0.0	0.54	0.0
Movehub-Quality-4	DSR-KL	63.05	4.14	0.88	0.0	0.84	0.0
	DSR-l2	531.86	212.91	0.82	0.07	0.75	0.08
	Siamese	710.35		0.88		0.82	
	SR-KL	98.02	3.73	0.17	0.0	-0.1	0.0
	SR-l2	18.98	4.11	0.84	0.0	0.62	0.0
IMDB-4	DSR-KL	23.93	16.15	0.9	0.0	0.9	0.05
	DSR-l2	54.16	13.22	0.78	0.035	0.38	0.07
	Siamese	3409.03		0.87		0.78	
	SR-KL	57.93	0.87	0.16	0.0	-0.04	0.0
	SR-l2	9.43	1.02	0.52	0.0	0.08	0.0
ICLR-4	DSR-KL	84.93	1.76	0.88	0.01	0.62	0.06
	DSR-l2	84.78	1.21	0.63	0.01	0.19	0.01
	Siamese	623.18		0.84		0.76	
	SR-KL	347.29	39.6	0.29		0.0	0.0
	SR-l2	503.11	40.35	0.86		0.82	0.0
ROP	DSR-KL	776.71	136.74	0.89	0.0	0.79	0.0
	DSR-l2	694.99	431.12	0.84	0.03	0.68	0.064
	Siamese	1152.06		0.86		0.73	
	SR-KL	610.91	20.69	0.5	0.0	0.0	0.0
	SR-l2	3.08	0.59	0.89	0.0	0.79	0.0
Movehub-Cost-5	DSR-KL	183.6	48.2	0.85	0.047	0.84	0.08
	DSR-l2	216.5	64.34	0.81	0.04	0.69	0.02
	Siamese	7986.22		0.89		0.83	
	SR-KL	189.85	0.72	0.45	0.0	0.28	0.0
	SR-l2	209.5	1.5	0.78	0.0	0.55	0.0
Movehub-Quality-5	DSR-KL	79.35	2.91	0.85	0.05	0.79	0.08
	DSR-l2	91.87	6.31	0.62	0.04	0.5	0.05
	Siamese	2241.49		0.89		0.83	
	SR-KL	924.07	0.2	0.13	0.0	-0.1	0.0
	SR-l2	208.1	0.1	0.84	0.0	0.65	0.0

Table 2: Rank Partitioning

Dataset	Method	Time (s) #		Performance on the Test Set			
				Top-1 Acc. ″		KT ″	
ICLR-3	DSR-KL	145.76	9.78	0.48	0.06	0.28	0.09
	DSR-l2	122.92	75.43	0.51	0.02	0.07	0.08
	Siamese	827.05		0.48		0.05	
	SR-KL	96.49	90.02	0.48	0.0	0.0	0.0
	SR-l2	4.91	4.6	0.47	0.0	0.06	0.0
Movehub-Cost-4	DSR-KL	17.65	7.68	0.61	0.07	0.45	0.12
	DSR-l2	19.85	3.7	0.05	0.08	-0.3	0.22
	Siamese	216.22		0.6		0.66	
	SR-KL	7.45	0.3	0.31	0.0	0.44	0.0
	SR-l2	4.13	0.2	0.5	0.0	0.71	0.0
Movehub-Quality-4	DSR-KL	31.41	5.02	0.88	0.0	0.74	0.05
	DSR-l2	34.72	8.68	0.67	0.11	0.81	0.31
	Siamese	258.87		0.88		0.88	
	SR-KL	7.68	0.2	0.48	0.0	0.05	0.0
	SR-l2	4.69	0.1	0.51	0.0	0.55	0.0
IMDB-4	DSR-KL	526.01	11.57	0.73	0.29	-0.14	0.25
	DSR-l2	27.73	26.63	0.21	0.21	-0.02	0.08
	Siamese	1240.98		0.47		0.02	
	SR-KL	31.33	13.26	0.04	0.0	0.05	0.0
	SR-l2	6.97	2.71	0.6	0.06	0.04	0.06
ICLR-4	DSR-KL	288.45	3.84	0.48	0.13	0.06	0.18
	DSR-l2	280.72	270.75	0.47	0.02	0.032	0.1
	Siamese	10142.55		0.32		-0.07	
	SR-KL	131.8	120.01	0.45	0.0	-0.07	0.0
	SR-l2	198.49	6.87	0.37	0.03	-0.01	0.0
ROP	DSR-KL	25.3	15.4	0.8	0.01	0.6	0.02
	DSR-l2	210.45	47.16	0.79	0.01	0.59	0.02
	Siamese	4438.61		0.82		0.65	
	SR-KL	527.76	163.76	0.61	0.0	0.23	0.0
	SR-l2	1.96	1.1	0.45	0.0	-0.08	0.0
Movehub-Cost-5	DSR-KL	111.13	31.61	0.76	0.29	0.67	0.15
	DSR-l2	137.14	65.55	0.19	0.05	0.52	0.34
	Siamese	2842.12		0.62		0.74	
	SR-KL	30.51	0.3	0.16	0.0	0.39	0.0
	SR-l2	30.14	0.2	0.37	0.0	0.73	0.0
Movehub-Quality-5	DSR-KL	114.03	30.18	0.92	0.06	0.35	0.25
	DSR-l2	46.99	13.23	0.57	0.34	0.73	0.46
	Siamese	752.06		0.76		0.08	
	SR-KL	29.68	1.2	0.62	0.0	0.08	0.0
	SR-l2	30.03	1.1	0.39	0.0	0.59	0.0

Table 3: Sample 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.



(a) Convergence on Movehub-Cost-4

(b) Convergence on Movehub-Quality-4

Figure 3: Log-likelihood $-\mathcal{L}$ on training and Top-1 Acc. on 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 iteration of ADMM, while each point for siamese corresponds to a training epoch.

Dataset	Method	Performance on the Test Set	
		Top-1 Acc. "	KT "
Movehub-Cost-4	DSR-KL	0.88	0.85
	One-iter.-DSR-KL	0.67	0.53
Movehub-Quality-4	DSR-KL	0.88	0.84
	One-iter.-DSR-KL	0.75	0.73

Figure 4: Test set predictions of DSR-KL vs. One-iter.-DSR-KL on Movehub-Cost-4 and Movehub-Quality-4.

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 L on training and Top-1 Acc. on 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 (c.f. (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 via repeated iterations of (13), and (ii) given , solving (8b) by training the DNN regressor $\tilde{}(X;W)$ 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 naive approach as One-iter.-DSR-KL. Unlike One-iter.-DSR-KL, our algorithm DSR-KL has the capability of repeatedly adapting both and W by solving (6) via ADMM. This advantage is illustrated in Figures 3a and 3b, where not only $\tilde{}(X;W)$, but also 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

Ataer-Cansızoğlu, E. (2015). *Retinal image analytics: A complete framework from segmentation to diagnosis*. Northeastern University.

Blitzer (2017). Movehub city rankings. `https://www.kaggle.com/blitzr/movehub-city-rankings?select=movehubqualityoflife.csv`.

Bosman, A. S., Engelbrecht, A., and Helbig, M. (2020). Visualising basins of attraction for the cross-entropy and the squared error neural network loss functions. *Neurocomputing*.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the Alternating Direction Method of Multipliers. *Foundations and Trends R in Machine Learning*, 3(1):1–122.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press.

Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1994). Signature verification using a "siamese" time delay neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 737–744.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 89–96. ACM.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine learning*, pages 129–136.

Caruana, R. and Niculescu-Mizil, A. (2004). Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 69–78.

Chang, H., Yu, F., Wang, J., Ashley, D., and Finkelstein, A. (2016). Automatic triage for a photo series. *ACM Transactions on Graphics (TOG)*, 35(4):148.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Doughty, H., Damen, D., and Mayol-Cuevas, W. (2018). Who's better? who's best? pairwise deep ranking for skill determination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6057–6066.

Dubey, A., Naik, N., Parikh, D., Raskar, R., and Hidalgo, C. A. (2016). Deep learning the city: Quantifying urban perception at a global scale. In *European Conference on Computer Vision (ECCV)*, pages 196–212. Springer.

Elo, A. E. (1978). *The rating of chessplayers, past and present*. Arco Pub.

Golik, P., Doetsch, P., and Ney, H. (2013). Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Interspeech*, volume 13, pages 1756–1760.

Guo, Y., Tian, P., Kalpathy-Cramer, J., Ostmo, S., Campbell, J. P., Chiang, M. F., Erdoğmuş, D., Dy, J. G., and Ioannidis, S. (2018). Experimental design under the Bradley-Terry model. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2198–2204.

Han, B. (2018). DATELINE: Deep Plackett-Luce model with uncertainty measurements. *arXiv preprint arXiv:1812.05877*.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM.

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Lei, Q., Zhong, K., and Dhillon, I. S. (2016). Coordinate-wise power method. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2064–2072.

Leka, O. (2016). IMDB movies dataset. `https://www.kaggle.com/orgesleka/imdbmovies`.

Li, Y., Cheng, X., and Gui, G. (2018). Co-robust-ADMM-net: Joint ADMM framework and DNN for robust sparse composite regularization. *IEEE Access*, 6:47943–47952.

Liu, R., Jiang, Z., Fan, X., Li, H., and Luo, Z. (2018). Single image layer separation via deep ADMM unrolling. In *2018 IEEE International Conference on Multimedia and Expo*, pages 1–6.

Ma, J., Liu, X.-Y., Shou, Z., and Yuan, X. (2019). Deep tensor ADMM-net for snapshot compressive imaging. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 10223–10232.

Ma, J., Yi, X., Tang, W., Zhao, Z., Hong, L., Chi, E. H., and Mei, Q. (2020). Learning-to-rank with partitioned preference: Fast estimation for the plackett-luce model. *arXiv preprint arXiv:2006.05067*.

Maystre, L. and Grossglauser, M. (2015). Fast and accurate inference of Plackett-Luce models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 172–180.

McFadden, D. (1973). Conditional logit analysis of qualitative choice behavior.

Pahikkala, T., Tsivtsivadze, E., Airola, A., Järvinen, J., and Boberg, J. (2009). An efficient algorithm for learning to rank from preference graphs. *Machine Learning*, 75(1):129–165.

Plackett, R. L. (1975). The analysis of permutations. *Applied Statistics*, pages 193–202.

Ryzin, G. v. and Mahajan, S. (1999). On the relationship between inventory costs and variety benefits in retail assortments. *Management Science*, 45(11):1496–1509.

Shi, Z., Zhang, X., and Yu, Y. (2017). Bregman divergence for stochastic variance reduction: saddle-point and adversarial prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6031–6041.

Sun, J., Li, H., Xu, Z., et al. (2016). Deep ADMM-Net for compressive sensing MRI. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10–18.

Sun, S.-H. (2020). Crawl and visualize ICLR 2020 openreview data. `https://github.com/shaohua0116/ICLR2020-OpenReviewData`.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR), 2015*.

Thurstone, L. L. (1927). The method of paired comparisons for social values. *The Journal of Abnormal and Social Psychology*, 21(4):384.

Tian, P., Guo, Y., Kalpathy-Cramer, J., Ostmo, S., Campbell, J. P., Chiang, M. F., Dy, J., Erdoğmuş, D., and Ioannidis, S. (2019). A severity score for Retinopathy of Prematurity. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1809–1819.

Wang, F., Cao, W., and Xu, Z. (2018). Convergence of multi-block Bregman ADMM for nonconvex composite problems. *Science China Information Sciences*, 61(12):122101.

Wang, H. and Banerjee, A. (2014). Bregman Alternating Direction Method of Multipliers. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2816–2824.

Xia, F., Liu, T.-Y., Wang, J., Zhang, W., and Li, H. (2008). Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199.

Yang, Y., Sun, J., Li, H., and Xu, Z. (2020). ADMM-CSNet: A deep learning approach for image compressive sensing. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI*, 42(3):521–538.

Ye, S., Feng, X., Zhang, T., Ma, X., Lin, S., Li, Z., Xu, K., Wen, W., Liu, S., Tang, J., Fardad, M., Lin, X., Liu, Y., and Wang, Y. (2019). Progressive DNN compression: A key to achieve ultra-high weight pruning and quantization rates using ADMM.

Ye, S., Zhang, T., Zhang, K., Li, J., Xie, J., Liang, Y., Liu, S., Lin, X., and Wang, Y. (2018). A unified framework of DNN weight pruning and weight clustering/quantization using ADMM.

Yıldız, İ., Dy, J., Erdoğmuş, D., Kalpathy-Cramer, J., Ostmo, S., Campbell, J. P., Chiang, M. F., and Ioannidis, S. (2020). Fast and accurate ranking regression. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Yıldız, İ., Tian, P., Dy, J., Erdoğmuş, D., Brown, J., Kalpathy-Cramer, J., Ostmo, S., Campbell, J. P., Chiang, M. F., and Ioannidis, S. (2019). Classification and comparison via neural networks. *Neural Networks*.

Yu, Y. and Açıkmeşe, B. (2019). Stochastic bregman parallel direction method of multipliers for distributed optimization. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 5550–5555. IEEE.

Yu, Y., Açıkmeşe, B., and Mesbahi, M. (2018). Bregman parallel direction method of multipliers for distributed optimization via mirror averaging. *IEEE Control Systems Letters*, 2(2):302–306.

Zhao, H. and Liao, P. (2019). CAE-ADMM: Implicit bitrate optimization via ADMM-based pruning in compressive autoencoders.

Zhao, P., Liu, S., Wang, Y., and Lin, X. (2018). An ADMM-based universal framework for adversarial attacks on deep neural networks. In *Proceedings of the 26th ACM International Conference on Multimedia*, page 1065–1073, New York, NY, USA. Association for Computing Machinery.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 19–27.

A Proof of Theorem 4.1

Let a stationary point $\in \mathbb{R}^n_+$ of the Augmented Lagrangian (7) be such that:

$$\frac{\partial L_\rho(\ , \mathbf{W}^k, \mathbf{y}^k)}{\partial \pi_i} = 0 \ \forall i \in [n] \tag{20a}$$

$$, \ \frac{\partial L(\mathcal{D}| \ )}{\partial \pi_i} + y_i^k + \rho \frac{\partial D_p(\ || \tilde{\ }^k)}{\partial \pi_i} = 0. \ \forall i \in [n] \tag{20b}$$

Let $\sigma_i(\ ) = \rho \frac{\partial D_p(\ || \tilde{\ }^k)}{\partial \pi_i} + y_i^k$, for all $i \in [n]$. Then, Eq.(20a) is equivalent to:

$$\frac{\partial L(\mathcal{D}| \ )}{\partial \pi_i} + \sigma_i(\ ) = 0 \ \forall i \in [n]\,. \tag{21}$$

Partial derivatives of the negative log-likelihood $L(\mathcal{D}| \ )$ are given by:

$$\frac{\partial L(\mathcal{D}| \ )}{\partial \pi_i} = \sum_{\ell \in W_i} \left(\frac{1}{\sum_{t \in A_\ell} \pi_t} \quad \frac{1}{\pi_i} \right) + \sum_{\ell \in L_i} \frac{1}{\sum_{t \in A_\ell} \pi_t}, \tag{22}$$

for all $i \in [n]$, where $W_i = \{\ell | i \in A_\ell, c_\ell = i\}$ is the set of observations where sample $i \in [n]$ is chosen and $L_i = \{\ell | i \in A_\ell, c_\ell \neq i\}$ is the set of observations where sample $i \in [n]$ is not chosen. Setting $\frac{\partial \mathcal{L}(\mathcal{D}| \ )}{\partial \pi_i}$ from Eq. (22) to Eq. (21), we have:

$$\frac{\partial L_\rho(\ , \mathbf{W}^k, \mathbf{y}^k)}{\partial \pi_i} = \sum_{\ell \in W_i} \left(\frac{1}{\sum_{t \in A_\ell} \pi_t} \quad \frac{1}{\pi_i} \right) + \sum_{\ell \in L_i} \frac{1}{\sum_{t \in A_\ell} \pi_t} + \sigma_i(\ ) = 0, \tag{23}$$

for all $i \in [n]$. Multiplying both sides of Eq. (23) with π_i, $i \in [n]$, we have:

$$\sum_{\ell \in W_i} \left(\frac{\sum_{j \neq i \in A_\ell} \pi_j}{\sum_{t \in A_\ell} \pi_t} \right) \quad \sum_{\ell \in L_i} \left(\frac{\pi_i}{\sum_{t \in A_\ell} \pi_t} \right) \quad \pi_i \sigma_i(\ ) = 0, \tag{24}$$

for all $i \in [n]$. Note that $\sum_{\ell \in W_i} \sum_{j \neq i \in A_\ell} = \sum_{j \neq i} \sum_{\ell \in W_i \cap L_j}$ and $\sum_{\ell \in L_i} = \sum_{j \neq i} \sum_{\ell \in W_j \cap L_i}$. Accordingly, we rewrite Eq. (24) as:

$$\sum_{j \neq i} \sum_{\ell \in W_i \cap L_j} \left(\frac{\pi_j}{\sum_{t \in A_\ell} \pi_t} \right) \quad \sum_{j \neq i} \sum_{\ell \in W_j \cap L_i} \left(\frac{\pi_i}{\sum_{t \in A_\ell} \pi_t} \right) \quad \pi_i \sigma_i(\ ) = 0, \tag{25}$$

for all $i \in [n]$. Then, the stationarity condition given by Eq.(20a) is equivalent to:

$$\sum_{j \neq i} \pi_j \lambda_{ji}(\ ) \quad \sum_{j \neq i} \pi_i \lambda_{ij}(\ ) = \pi_i \sigma_i(\ ) \ \forall i \in [n]\,, \tag{26}$$

where $\lambda_{ji}(\ )$, $i, j \in [n]\,, i \neq j$ are given by Eq. (12).

It is not evident that Eq.(26) corresponds to the balance equations of an MC as, in general, $(\ ) = [\sigma_i(\ )]_{i \in [n]} \neq \mathbf{0}$. Nevertheless, for $\sigma_i(\ ) = \rho \frac{\partial D_p(\ || \tilde{\ }^k)}{\partial \pi_i} + y_i^k$, $i \in [n]$, Eq.(26) has the same form as the balance equations in Theorem 4. 2 established by Yıldız et al. (2020). By this theorem, a stationary $\in \mathbb{R}^n_+$ satisfying (20a) is also the stationary distribution of the continuous-time MC with transition rates given by Eq. (10). □

B Datasets

Retinopathy of Prematurity (ROP). The Retinopathy of Prematurity (ROP) dataset contains $n = 100$ vessel-segmented retina images with dimensions $d = 224 \quad 224$ (Ataer-Cansızoğlu, 2015). Experts are provided

with two images and are asked to choose the image with higher severity of the ROP disease. Five experts independently label 5941 image pairs; the resulting dataset contains $m = 29705$ pairwise comparisons. Note that some pairs are labelled more than once by different experts.

International Conference on Learning Representations (ICLR). The ICLR Dataset contains abstracts and reviewer ratings of 2561 papers that are submitted to ICLR 2020 conference and are available on OpenReview website (Sun, 2020). We choose the top $n = 100$ papers, and extract $d = 768$ numerical features from each abstract using the Deep Bidirectional Transformers (BERT) (Devlin et al., 2019) architecture, pre-trained on the Books Corpus dataset (Zhu et al., 2015) and English Wikipedia. We normalize $\boldsymbol{X}$ to have 0 mean and unit variance over samples $[n]$. We generate all possible $m = 120,324(2,248,524)$ $K = 3(4)$-way rankings w.r.t. the relative order of the average reviewer ratings. We add noise to the resulting rankings following the same process as Movehub-Cost.

Movehub-Cost. The Movehub-Cost dataset contains the total ranking of 216 cities w.r.t. cost of living (Blitzer, 2017). Each city is associated with $d = 6$ numerical features, which are average costs for cappuccino, cinema, wine, gasoline, rent, and disposable income. We normalize $\boldsymbol{X}$ to have 0 mean and unit variance over samples $[n]$. We select $n = 50$ cities and generate all $m = 230,298(2,118,756)$ $K = 4(5)$-way rankings w.r.t. the relative order of the queried cities in the total ranking. To mimic the real-life noise introduced by human labelling, we apply the following post-processing to the resulting rankings: For each ranking, we sample a value uniformly at random in $[0, 1]$. If the value is less than 0.1, we add noise to the ranking by a cyclic permutation of the ranked samples.

Movehub-Quality. The Movehub-Quality dataset contains total ranking of the same 216 cities as Movehub-Cost, this time w.r.t. quality of life. Each city is associated with $d = 5$ numerical features, including overall scores for purchase power, healthcare, pollution, quality of life, and crime. We normalize $\boldsymbol{X}$ to have 0 mean and unit variance over samples $[n]$. We select $n = 50$ cities and generate all $m = 230,298(2,118,756)$ $K = 4(5)$-way rankings w.r.t. the relative order of the queried cities in the total ranking. We add noise to the rankings following the same process as Movehub-Cost.

IMDB. The IMDB Movies Dataset contains IMDB ratings of 14,762 movies, each of which is associated with $d = 36$ numerical features (Leka, 2016). We normalize $\boldsymbol{X}$ to have 0 mean and unit variance over samples $[n]$. We select $n = 50$ movies and generate all possible $m = 85,583$ $K = 4$-way rankings w.r.t. the relative order of the ratings of queried movies. We add noise to the resulting rankings following the same process as Movehub-Cost.