Group Match Prediction via Neural Networks

Sunghyun Kim[1], Minje Jang[2] and Changho Suh[3]

[1]Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA, 02139, USA
[2]Krafton, 231 Teheran-ro, Gangnam-gu, Seoul, 06142, South Korea
[3]Korea Advanced Institute of Science and Technology, 291 Daehak-ro, Yuseong-gu, Daejeon, 34141, South Korea

Abstract

We consider the group match prediction problem where the goal is to estimate the probability of one group of items preferred over another, based on partially observed group comparison data. Most prior algorithms, each tailored to a specific statistical model, suffer from inconsistent performances across different scenarios. Motivated by a key common structure of the state-of-the-art, which we call the *reward-penalty* structure, we develop a *unified* framework that achieves consistently high performances across a wide range of scenarios. Our distinction lies in introducing neural networks embedded with the reward-penalty structure. Extensive experiments on synthetic and real-world datasets show that our framework consistently leads to the best performance, while the state-of-the-art perform inconsistently.

Keywords

group match prediction, group recommendation, neural network

1. Introduction

We often compare a pair of items (or groups of items) and judge which one is of higher utility. We also become interested in predicting comparison outcomes for unobserved pairs, based on partial comparison data for previously observed pairs, and in recommending select options in preference to the others. We investigate the group match prediction (group recommendation) problem where we aim to estimate the probability of one group of M items preferred over another, based on partially observed group comparison data.

Most prior algorithms postulate ground-truth utilities for items, and assume underlying models to exist. These models are considered to describe the utility of a group as a function of the utilities of its items, and govern statistical patterns of comparison data based on the group utilities [1, 2, 3, 4]. Some algorithms have been developed for prominent yet specific statistical models, and shown to attain optimal performances (see Section 4.1). Yet, major challenges arise when we attempt to employ them in practice.

First, prior algorithms lead to *inconsistent* performances. Tailored to specific models, they achieve decent performances in some scenarios that are well-represented by their models, but perform poorly in others. This inconsistency limits the application of each algorithm to a narrow range of scenarios.

Second, no prior algorithm can offer a satisfactory performance when a given scenario cannot be represented by well-studied models. For instance, underlying models for some scenarios can be inherently too complex to be approximated by existing models. In such cases, all prior algorithms are ill-suited.

We address all these challenges by developing a *unified* algorithmic framework that can (*i*) infer and adapt to any underlying models for given scenarios; and (*ii*) achieve consistently high performances.

Main Contributions. *First*, we identify a key common structure (which we call the *reward-penalty* structure, to be detailed in Section 4.1) shared among state-of-the-art algorithms, and incorporate the structure into our framework. This emphasis on structural aspects enables our framework to attain high performances (Section 5). *Second*, we introduce neural networks so as to enable our framework to infer and adapt to any latent models. We design the neural networks to well-respect the reward-penalty structure in order to retain high performances across vairous scenarios. (Sections 4.2 and 4.3) *Third*, we construct the neural networks in such a way that our framework can be robust against prohibitive scalability issues. This robustness makes our framework applicable to a broader range of scenarios which involve large numbers of items.

Insight from the state of the arts: Looking into state-of-the-art algorithms in the tasks of match prediction and rank aggregation (a related and long-studied task) [5, 6, 1, 2], we find that they share a key element: so-called the *reward-penalty* mechanism in estimating the utilities of items. The mechanism rewards an item greatly for winning (or being more preferred) in a disadvantageous comparison where its group is weaker than the counterpart. Likewise, it penalizes an item greatly for

3rd Edition of Knowledge-aware and Conversational Recommender Systems (KaRS) & 5th Edition of Recommendation in Complex Environments (ComplexRec) Joint Workshop @ RecSys 2021, September 27–1 October 2021, Amsterdam, Netherlands
sunghyun@mit.edu (S. Kim); jangminje427@krafton.com (M. Jang); chsuh@kaist.ac.kr (C. Suh)

losing (or being less preferred) in an advantageous one. Also, the magnitudes of such rewards and penalties are proportional to its contribution to its group. It turns out that across different state-of-the-art algorithms, only the terms corresponding to rewards and penalties vary (Section 4.1 for details), but the common mechanism remains unchanged. We take it as the main structure in our framework ((5) in Section 4.2 for details).

Neural networks: Motivated by the above observations, we incorporate three separate neural network modules into our framework (Sections 4.2 and 4.3 for architectural details): two modules represent reward and penalty mechanisms, and the other combines them to produce final comparison outcome predictions. These modules are trained according to the given dataset so that they can adapt to any hidden models that underlie the dataset, making our framework attain universal applicability. The overall architecture that puts all three modules together is specifically designed to maintain the reward-penalty structure, in order to retain high performances across various scenarios.

Scalability: A prior work employed a single-layer neural network as an initial effort to predict winning probabilities of group matches [7]. This attempt showed a promising result, demonstrating improved prediction accuracy on a real-world online game dataset. However, it exhibits scalability issues. It requires one input node per item, thus is too prohibitive to be extended for scenarios with large numbers of items (scaling with n). In contrast, we design our neural network modules carefully to avoid such scalability issues (Figures 1 and 2: the input dimensions are fixed to $2M$ regardless of n). It also provides higher performances compared to the prior work.

Extensive experiments: We compare our framework with the single-layer baseline and other algorithms by extensive experiments on synthetic and real-world datasets. Using synthetic datasets (Section 5.1), we show that our framework achieves the best performance across four prominent models in the literature, while the other algorithms suffer from inconsistent performances across different models. Three models consider extensions of the Bradley-Terry-Luce model [8] to the group comparison scenario. The other considers a generalized version of the Thurstone model [4] widely used in skill rating systems of online games. Using real-world datasets (Section 5.2), we show that our framework performs consistently well across diverse real-world scenarios. We consider five real-world datasets (sources in Footnote 7). One is a crowd-sourced image classification dataset, another is a collection of movie ratings, and the other three are online game match records. In our performance evaluations, we use the cross entropy loss (see (1)) and the prediction accuracy (see (8)).

Ablation studies: To verify that the reward-penalty structure in our framework plays a key role in bringing performance improvements, we conduct ablation studies (Table 2 in Section 5.2). As our baseline, we consider our framework where the reward and penalty modules are arbitrarily made defunct (Section 5.2 for details). Evaluating the performances of our framework and the baseline on all five real-world datasets in terms of both cross entropy loss and prediction accuracy, we demonstrate that the reward and penalty modules consistently improve performances across all datasets and metrics, confirming their effectiveness.

2. Related Work

The most relevant prior works are [1, 2, 3, 4]. These works assume prominent models (some long-established and some widely used in practice) for in-group interactions and group comparisons, and carry out statistical analysis to a great extent. We compare our framework with the algorithms developed therein.

The problem of estimating individual item utilities from group comparison data has been investigated in [1, 2]. They considered extensions of the BTL model where the group utility is either the sum or the product of its items' utilities, and group comparisons follow the BTL model. We call the two models the BTL-sum and BTL-product models respectively.

A more advanced in-group interaction model has been developed in [3]. They considered a scenario where a pair of individual items in a group leads to a synergy. The group utility is represented as the sum of two quantities, which we call the HOI model: (1) the sum of individual item utilities and (2) the sum of the products of all pairs of individual item utilities. The work of [9] has considered a scenario where any k-tuple of items leads to a synergy.

In [4], they assumed individual item utilities to be centered around a mean following a Gaussian distribution and viewed the group utility as their sum, which we call the Thurstone model. Their algorithm is widely used in skill rating systems of online games where groups of users compete.

The work of [7] has made an initial effort in employing a neural network to predict winning probabilities of group matches. It has been shown that a single-layer neural network can fit some variants of the BTL model [2] and improve prediction accuracy through experiments on a real-world online game dataset. As noted, one clear distinction compared to our work lies in scalability. The neural network developed in the prior work requires one input node per item, thus is prohibitive for scenarios with large numbers of items. In contrast, we develop a neural network that requires *only a fixed* number of input nodes regardless of the number of items (Section 4.2 for details).

3. Problem Setup

The goal is to predict comparison outcome likelihoods for unobserved pairs of groups, given partial group comparison data. Each data point consists of (A, B, y_{AB}), (A, B, y_{AB}). A and B are groups of M individual items. y_{AB} indicates which group is preferred: $y_{AB} = \mathbb{I}(A \succ B)$ where $A \succ B$ indicates A preferred over B. We denote the set of observed comparisons by D_{obs} and that of unobserved comparisons by D_{unobs}. Our training dataset available is $\{(A, B, y_{AB})\}_{(A,B)\in D_{\text{obs}}}$. We use the cross entropy loss as our metric. Let us define $\hat{y}_{AB}$ as the estimate of $\Pr[y_{AB} = 1]$ produced by our algorithmic framework. Then formally, our goal is to minimize:

$$\frac{-1}{|D_{\text{obs}}|} \sum_{(A,B)\in D_{\text{obs}}} y_{AB} \log \hat{y}_{AB} + (1 - y_{AB}) \log(1 - \hat{y}_{AB}). \tag{1}$$

Notation. $[n] = \{1, 2, \dots, n\}$ is the set of all items. We use lowercase letters (e.g., i) for individual items, and uppercase letters (e.g., A) for sets of items. $\{w_i\}_{i\in A}$ is the set of w_i's for all $i \in A$. $[w_i]_{i\in A}$ is a *vector* of w_i's for all $i \in A$ and its ordering is provided in the context. $\hat{y}$ is an estimate of y. We use subscripts as in y_{AB} when y concerns a comparison between A and B. We use superscripts as in $w^{(\ell)}$ when w is updated iteratively. We use bold symbols as in $\boldsymbol{w}$ to indicate the vector of w_i's for all $i \in [n]$.

4. Proposed Algorithm

4.1. Motivation

Our framework design is inspired by optimal state-of-the-art algorithms (achieving minimal sample complexity or global minima of cross entropy loss) in extensions of the Bradley-Terry-Luce model (BTL) model [8]. We make the following key observations regarding the structural similarities shared by the state-of-the-art:

(a) They exhibit "reward" and "penalty" terms in the estimation process (details in (2) and (3)). In updating an item's utility estimate, they reward the item for contributing to its group's winning, and penalize it for contributing to its group's losing.

(b) The expressions of these reward/penalty terms vary as the underlying models change.

(c) The magnitudes of rewards and penalties depend on the power dynamics between groups. A greater reward is given to an item when its group is relatively weaker compared to the opponent group, and likewise a greater penalty is given when its group is relatively stronger.

(d) Their magnitudes depend on the portion of an item's contribution (or blame) within its group. Suppose an item's group wins (or loses) in a comparison. The item is given a greater reward (or penalty) when its share of contribution (or blame) within the group is relatively greater.

Reward-Penalty Structure. Let us examine the algorithms developed in extensions of the BTL model.

- *Rank Centrality* [5] has been developed under the standard BTL model where individual items are compared in pairs. It achieves the minimal sample complexity for top-K rank aggregation, whose task is to estimate the set of top-K items, in certain regimes [10, 11]. As in Section 3, we define $y_{ij} = \mathbb{I}(i \succ j)$, given a pair of items i and j. By some manipulation, the update rule at step ℓ for individual item utility $w_i^{(\ell)}$ of item i can be shown as[1]:

$$w_i^{(\ell+1)} \leftarrow w_i^{(\ell)} + \alpha \sum_{j:(i,j)\in D_{\text{obs}}} \left(y_{ij} w_j^{(\ell)} - (1 - y_{ij}) w_i^{(\ell)} \right). \tag{2}$$

  For item i, one can interpret $w_j^{(\ell)}$ as the reward since it increases $w_i^{(\ell+1)}$ when $i \succ j$ ($y_{ij} = 1$), and $w_i^{(\ell)}$ (next to $(1 - y_{ij})$) as the penalty since it decreases $w_i^{(\ell+1)}$ when $i \prec j$ ($y_{ij} = 0$). α is a step size in the update. Note that the reward is large when the opponent's utility estimate is large; we give a large reward for a win against a strong opponent (observation (a) above). Likewise, the penalty is large when its own utility estimate is large; we give a large penalty for a loss against a weak opponent (observation (c) above).

- Majorization-Minimization (MM) for the BTL-sum model has been developed in [6, 1]. We define $w_A^{(\ell)} := \sum_{i\in A} w_i^{(\ell)}$. By some manipulation, the update rule at step ℓ for individual item utility $w_i^{(\ell)}$ of item i can be shown as[2]:

$$w_i^{(\ell+1)} \leftarrow w_i^{(\ell)} + \alpha_i \sum_{\substack{(A,B)\in D_{\text{obs}} \\ i\in A}} \left(y_{AB} \cdot \mathsf{R}_{AB,i}^{(\ell)} - (1 - y_{AB}) \cdot \mathsf{P}_{AB,i}^{(\ell)} \right) \tag{3}$$

  where

$$\mathsf{R}_{AB,i}^{(\ell)} = \frac{w_B^{(\ell)}}{w_A^{(\ell)} + w_B^{(\ell)}} \cdot \frac{w_i^{(\ell)}}{w_A^{(\ell)}}, \quad \mathsf{P}_{AB,i}^{(\ell)} = \frac{w_A^{(\ell)}}{w_A^{(\ell)} + w_B^{(\ell)}} \cdot \frac{w_i^{(\ell)}}{w_A^{(\ell)}}. \tag{4}$$

[1] As in [5], $\alpha = \frac{1}{\max_i d_i}$ where d_i is the number of distinct items to which item i is compared. Also, we describe *Rank Centrality* as an iterative algorithm (one way to obtain the stationary distribution of the empirical pairwise preference matrix) to highlight its inherent reward-and-penalty structure.

[2] $\alpha_i = \left(\sum_{(A,B)\in D_{\text{obs}}: i\in A} \left(w_A^{(\ell)} + w_B^{(\ell)} \right)^{-1} \right)^{-1}$.

Note that the update rule (3) is similar to (2) of *Rank Centrality*, but the reward and penalty terms in (4) are different (observation (b) above). The interpretation is similar. The reward for item i is large when the opponent group's utility estimate is large (see $w_B^{(\ell)}$ in the numerator of $\mathsf{R}_{AB,i}^{(\ell)}$). Note also that the larger the contribution of item i within its own group (see $w_i^{(\ell)}/w_A^{(\ell)}$ of $\mathsf{R}_{AB,i}^{(\ell)}$), the greater the reward (observation (d) above). The same holds for the penalty.

- MM for the BTL-product model has been developed in [2] and shown to achieve global minima in terms of cross entropy loss. Its individual item utility update rule is described as in (3) but the reward and penalty terms are different.[3] A similar interpretation applies, and all observations (a)–(d) can be found.

We introduce two separate modules to represent rewards and penalties respectively. We employ *neural networks* for the modules so they can adapt to the underlying models for *any* given scenario and dataset.

4.2. Modules R and P

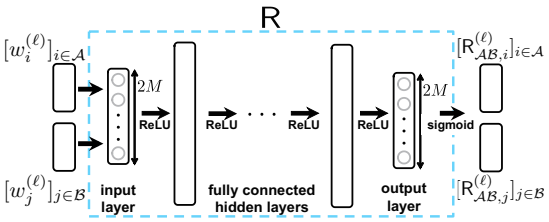

Figure 1: Architecture of module R (and P, which is omitted as it has the same structure with different weights). It takes as input utility estimates for the $2M$ items in a pair of groups (A, B) and produces as output reward and penalty estimates for the items.

Structure: Figure 1 depicts the detailed architecture of the two modules which reflect rewards (R) and penalties (P) discussed in Section 4.1. The input and output of module R and P are of dimension $2M$. M can be set arbitrarily according to the scenario of interest. R (P respectively) takes as input the current utility estimates of the individual items in a given group comparison (A, B), and produces as output the current reward (penalty respectively) estimates for the items. All layers are fully connected. The activations between layers are rectified linear units [12]. The final activation is the sigmoid function [13].

[3] $\alpha_i = \left(\sum \frac{w_A^{(\ell)}}{w_A^{(\ell)}+w_B^{(\ell)}}\right)^{-1}$ where $w_A^{(\ell)} = \prod_i w_i^{(\ell)}$, $\mathsf{R}_{AB,i}^{(\ell)} = \frac{w_B^{(\ell)}}{w_A^{(\ell)}+w_B^{(\ell)}} \cdot w_i^{(\ell)}$, $\mathsf{P}_{AB,i}^{(\ell)} = \frac{w_A^{(\ell)}}{w_A^{(\ell)}+w_B^{(\ell)}} \cdot w_i^{(\ell)}$.

Operation: Recall that our comparison data samples do not include utility estimate information. Each sample only specifies the pair of groups compared (A, B) and the comparison outcome $\hat{y}_{AB}$. Thus, we begin with a randomly initialized utility estimate vector $\boldsymbol{w}^{(0)} \in \mathbb{R}^n$ (initialization details in Section 4.4). Starting from $\boldsymbol{w}^{(0)}$, we obtain $\boldsymbol{w}^{(L)} \in \mathbb{R}^n$ by applying modules R and P repeatedly L times to update $\boldsymbol{w}^{(\ell)}$ as follows:

$$w_i^{(\ell+1)} \leftarrow w_i^{(\ell)} + \alpha \sum_{\substack{(A,B)\in D_{\text{obs}} \\ i\in A}} \left(y_{AB} \cdot \mathsf{R}_{AB,i}^{(\ell)} - (1-y_{AB}) \cdot \mathsf{P}_{AB,i}^{(\ell)}\right), \tag{5}$$

where[4] $\alpha = \frac{c}{\max_i d_i}$ and $d_i = |\{(A,B) : i \in A \cup B\}|$.

At step ℓ, given $\boldsymbol{w}^{(\ell)}$, R and P produce positive real values in $[0, 1]$. They are reward and penalty values respectively, and are used to obtain $\boldsymbol{w}^{(\ell+1)}$ as per (5). They are given as follows:

$$\begin{aligned} \mathsf{R}_{AB,i}^{(\ell)} &= \mathsf{R}\left(i, [w_j^{(\ell)}]_{j\in A}, [w_k^{(\ell)}]_{k\in B}\right), \\ \mathsf{P}_{AB,i}^{(\ell)} &= \mathsf{P}\left(i, [w_j^{(\ell)}]_{j\in A}, [w_k^{(\ell)}]_{k\in B}\right). \end{aligned} \tag{6}$$

At the end of step ℓ, we normalize $\boldsymbol{w}^{(\ell+1)}$ to be zero-mean $w_i^{(\ell+1)} \leftarrow w_i^{(\ell+1)} - \sum_{i=1}^n \frac{w_i^{(\ell+1)}}{n}$, and unity-norm $w_i^{(\ell+1)} \leftarrow \frac{w_i^{(\ell+1)}}{\|\boldsymbol{w}^{(\ell+1)}\|_2}$. We repeat until we obtain $\boldsymbol{w}^{(L)}$. Figure 3 illustrates the process. See the first stage therein.

Scalability: The input and output dimensions ($2M$) are *independent* of the total number of items (n). Thus, our framework does not suffer from scalability issues in contrast to a prior neural network based approach [7] in which the input dimension of the neural network therein scales in proportion to n. This scalability issue for the prior approach manifests in our real-world datasets with large numbers of items. See our real-world data experiment results for *GIFGIF* and *IMDb 5000* datasets in Table 1 in Section 5.2.

Refinement: We apply R and P multiple times ($L > 1$) to obtain a final utility estimate vector $\boldsymbol{w}^{(L)}$. The best value of L is set via hyper-parameter tuning. This series of applications is to "refine" utility estimates.

Data augmentation: R and P take as input a concatenation of two vectors $[w_j^{(\ell)}]_{j\in A}$ and $[w_k^{(\ell)}]_{k\in B}$. As they are vectors, not sets, ordering matters. We apply data augmentation to make our framework robust against arbitrary orderings of the items within a group. Specifically, given a sample, we create extra samples which represent the same outcome but have different item orderings. For example, given

[4] Prior work (see Footnote 1) has motivated the choice of α. The numerator c is determined by hyper-parameter tuning.

a sample $(A = (1, 2), B = (3, 4), y_{AB} = 1)$, we create extra samples such as $(A' = (2, 1), B' = (4, 3), y_{A'B'} = 1)$. We also seek robustness against arbitrary orderings of the groups. Specifically, we create extra samples by changing the order of two sets A and B as in $(A = (3, 4), B = (1, 2), y_{AB} = 0)$ and A' and B' as in $(A' = (4, 3), B' = (2, 1), y_{A'B'} = 0)$. These techniques help train R and P so as to be robust against item orderings within a group and group orderings.

Ablation study: R and P are the most distinctive features of our framework. To evaluate their effectiveness, we conduct ablation studies. See Table 2 in Section 5.2.

4.3. Module G

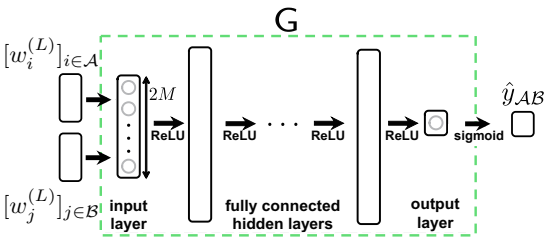

Figure 2: Architecture of module G. It takes as input utility estimates for the $2M$ items in (A, B) (obtained by modules R and P) and produces as output a group comparison prediction.

To perform our main task, we need more than rewards and penalties. As seen in Section 4.1, latent models can be assumed to exist. In-group interaction models (among items within a group) lead to group utilities. Group comparison models (between a pair of groups) govern statistical patterns of comparison outcomes. The role of module G is to fit these models. Modules R and P help quantify and provide item utility estimates. Then module G takes as input the item utility estimates for a pair of groups, and produces as output the probability of one group preferred over the other. The three modules interact closely to perform the task. The overall architecture that ties them all will soon be depicted at the end of this section.

Structure: Figure 2 depicts the detailed architecture of module G. The input and output of module G are of dimension $2M$ and a scalar respectively. As in modules R and P, we set the value of M as per the given scenario of interest. As the dimensions are independent of the total number of items (n), the module does not suffer from scalability issues. All layers are fully connected. The activations between layers are rectified linear units [12]. The final activation is the sigmoid function [13].

Operation: Given a group comparison (A, B), the module takes as input the utility estimates of the items in groups A and B, and produces as output the winning probability estimate of A preferred over B:

$$\hat{y}_{AB} = \mathrm{G}\left([w_i^{(\ell)}]_{i\in A}, [w_j^{(\ell)}]_{j\in B}\right). \tag{7}$$

As in (6), module G takes as input a concatenation of two vectors $[w_i^{(\ell)}]_{i\in A}$ and $[w_j^{(\ell)}]_{j\in B}$. The item and group orderings for the input to G are the same as those for the input to R and P. As mentioned, the item utilities used as input to module G are obtained from the operation of modules R and P.

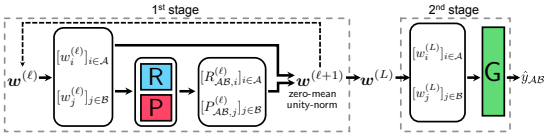

Figure 3: Overall architecture of our framework.

Figure 3 illustrates the overall architecture where modules R, P and G are put together. The first stage depicts the operation of R and P (initialization of $\boldsymbol{w}^{(0)}$ omitted). The second stage depicts the operation of G. Note that the reward-penalty structure is present. This structure is unaffected during the training process where only the parameters inside the modules are updated.

4.4. Training Procedure

We split D_{obs} randomly into D_{train} and D_{val}. We let D_{val} be a fraction (1%–2%) of D_{obs} and use it for validation purposes. We divide D_{train} into B equal-size batches. We denote by D^b_{train} the data in batch $b \in [B]$.

(*a*) *Initialization:* Initialize $\boldsymbol{w}^{(0)}$ using a Gaussian distribution whose mean is zero and variance is the normalized identity matrix, and the parameters of R, P and G using the Xavier initialization [14].

(*b*) *Refinement:* For all samples $(A, B) \in D^b_{\text{train}}$, start with $\{([w_i^{(0)}]_{i\in A}, [w_j^{(0)}]_{j\in B})\}_{(A,B)\in D^b_{\text{train}}}$ initialized in (*a*), apply R and P modules L times, and obtain $\{([w_i^{(L)}]_{i\in A}, [w_j^{(L)}]_{j\in B})\}_{(A,B)\in D^b_{\text{train}}}$.

(*c*) *Prediction:* For all samples $(A, B) \in D^b_{\text{train}}$, start with $\{([w_i^{(L)}]_{i\in A}, [w_j^{(L)}]_{j\in B})\}_{(A,B)\in D^b_{\text{train}}}$ obtained in (*b*), apply G module, and obtain $\{\hat{y}_{AB}\}_{(A,B)\in D^b_{\text{train}}}$.

(*d*) *Model parameter update:* Update the parameters of R, P, and G via the Adam optimizer [15] to minimize the loss. To compute the loss, replace D_{obs} in (1) by D^b_{train}. Apply weight decay regularization with a factor of 0.01. Repeat (*b*)–(*d*) for all batches $b \in [B]$.

The parameters are updated B times using D_{train} once in one epoch. We use 500 epochs. In each, we compute a validation loss using D_{val}. We apply early stopping and choose the parameters with the lowest loss.

5. Experimental Results

To verify the broad applicability of our framework, we conduct extensive experiments using synthetic and real-world datasets. We compare it with six other algorithms: MM-sum [1], MM-prod [2], SGD-HOI [3], *TrueSkill* [4], *Rank Centrality* [5], and an algorithm based on a single-layer neural network [7]. SGD-HOI has been developed for the factorization HOI model in [3] and *TrueSkill* for the Thurstone model in [4]. As *Rank Centrality* has been developed for the case of comparing two individual items, we consider its natural extensions depending on the dataset. For example, in the sum model, we replace the summation in (2) by $\sum_{(A,B)\in D_{\text{obs}}:i\in A}\left(y_{AB}\sum_{k\in B}w_k^{(\ell)}+(1-y_{AB})\sum_{k\in A}w_k^{(\ell)}\right)$. The single-layer neural network algorithm developed in [7] has a scalability issue. It requires at least one input node per item, thus becomes bloated with large n. This drawback prevents us from measuring its performance in our setup[5] for some real-world datasets such as *GIFGIF* and *IMDb 5000* (to be presented in Table 1 in Section 5.2).

5.1. Synthetic Data Experiments

We use four synthetic datasets: BTL-sum, BTL-product, HOI and a generalized Thurstone. We set $n = 300$ and $M = 5$. In the HOI model, we generate the ground-truth utilities and dimension-7 features using Gaussian distributions. In the others, we generate the ground-truth utilities uniformly at random. We generate $5n\log n$ distinct paired groups and each pair is compared 10 times.[6]

We split generated datasets randomly into D_{obs} (90%) and D_{unobs} (10%). All algorithms use D_{obs} to predict unobserved group comparisons in D_{unobs}. They use a fraction (1%–2%) of D_{obs} for validation purposes. We use $L = 20$ and the learning rate of 10^{-2}. We use four hidden layers for all modules. For R and P, each layer consists of $7M$ nodes, and for G, each layer consists of $9M$ nodes.

Figure 4 shows our result. The performance curves of the algorithms that underperform by large gaps are not presented. Let us discuss each sub-figure from left to right. (1) *BTL-sum model:* In most settings where we have sufficient data, our framework achieves the performance promised by MM-sum, which has been shown in [1] to achieve local minima in terms of cross entropy loss. (2) *BTL-product model:* Our framework achieves the optimal performance promised by MM-prod, which has been shown in [2] to achieve global minima in terms of cross entropy loss. (3) *HOI model:* SGD-HOI performs best in most settings. Our framework is second-best with a slight gap to the best performance in those settings, but performs best when we have sufficient data. This is because our framework using neural networks is affected by overfitting with insufficient data. (4) *Thurstone model:* MM-prod and our framework perform best. *TrueSkill* comes next with a gap, but it clearly outperforms the others. Interestingly, *TrueSkill*, developed specifically for the Thurstone model, does not lead to the best performance. But this result does not run counter to theory, as its optimality has not been shown in the literature.

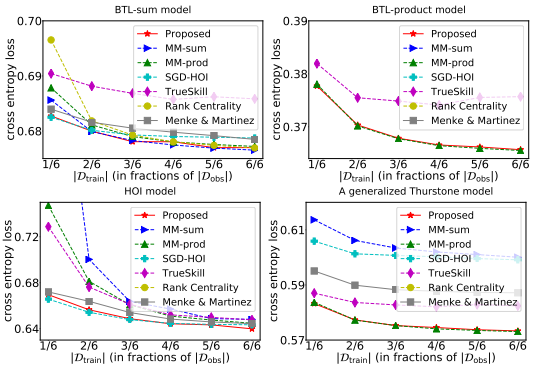

Figure 4: Synthetic data experiments. From left to right, top to bottom: BTL-sum model, BTL-product model, HOI model, and a generalized Thurstone model. Off-the-scale curves due to low performance algorithms are not shown. Amount of data used for training is described in fractions of D_{obs}, 90% of the dataset for training (x-axis). Performance is obtained by using D_{unobs}, 10% of the dataset for testing (y-axis).

Our framework performs consistently best (or near-best with a marginal gap) across all datasets, while the others perform inconsistently across them. Some perform well in one, but poorly in the others (for example, MM-sum performs best only in the BTL-sum model but poorly in all others). It is important to note that our framework outperforms the single-layer neural network baseline developed in [7] across all datasets.

5.2. Real-World Data Experiments

As in Section 5.1, we split real-world datasets randomly into D_{obs} (90%) and D_{unobs} (10%), and use a fraction (1%–2%) of D_{obs} for validation purposes if necessary. We

[5] Intel Core i7-6850K @ 3.6GHz (CPU) and GeForce GTX 1080 Ti (Single GPU).

[6] The $5n\log n$ is chosen in order to guarantee connectedness between any pair of nodes within a comparison (hyper-)graph where nodes represent items and an hyper-edge represents a group comparison [16]. Without connectedness guarantees, some algorithms may fail to produce meaningful estimates.

Table 1
Match prediction results in terms of cross entropy loss and prediction accuracy in real-world dataset experiments (N/A: not available due to scalability issues). The best performances are **boldfaced** and the second-best are underlined. The numbers in parentheses indicate the ranks in each dataset and metric.

	GIFGIF		HOTS		DOTA 2		LoL		IMDb 5000	
	CE-Loss	Acc	CE-Loss	Acc	CE-Loss	Acc	CE-Loss	Acc	CE-Loss	Acc
Proposed	**.3053** (1)	.8729 (3)	**.6790** (1)	**.5673** (1)	**.6599** (1)	.6090 (2)	**.6936** (1)	**.5411** (1)	**.8107** (1)	**.6016** (1)
Menke & Martınez	N/A	N/A	.6830 (6)	.5458 (7)	.6601 (2)	**.6127** (1)	.6937 (2)	.5236 (5)	N/A	N/A
MM-sum	.3114 (3)	**.8758** (1)	.6796 (4)	.5645 (4)	.6610 (4)	.6049 (5)	.7036 (3)	.5166 (7)	1.0719 (4)	.5812 (2)
MM-prod	.3126 (4)	**.8758** (1)	.6793 (3)	.5660 (2)	.6603 (3)	.6083 (3)	.7070 (5)	.5249 (3)	3.7455 (5)	.5428 (5)
SGD-HOI	.4056 (6)	.8614 (5)	.6798 (5)	.5562 (5)	.6673 (6)	.5955 (6)	.7125 (6)	.5171 (6)	.9471 (3)	.5780 (3)
TrueSkıll	.3063 (2)	.8728 (4)	.6857 (7)	.5499 (6)	.6683 (7)	.5863 (7)	.7057 (4)	.5245 (4)	.8409 (2)	.5736 (4)
Rank Centralıty	.3761 (5)	.8555 (6)	.6792 (2)	.5660 (2)	.6667 (5)	.6051 (4)	.7230 (7)	.5276 (2)	3.7455 (5)	.5096 (6)

Table 2
Ablation study results that evaluate the effectiveness of modules R & P in real-world dataset experiments. "Not Trained" indicates R & P made defunct; to produce coarse item utility estimates, they are active once ($L = 1$) without refinements, and data augmentation and model updates are not applied. "Trained" indicates our complete framework.

	GIFGIF		HOTS		DOTA 2		LoL		IMDb 5000	
	CE-Loss	Acc	CE-Loss	Acc	CE-Loss	Acc	CE-Loss	Acc	CE-Loss	Acc
R & P Not Traıned	0.338	0.864	0.693	0.515	0.689	0.535	0.696	0.520	0.743	0.538
R & P Traıned	0.296	0.875	0.682	0.560	0.660	0.614	0.690	0.539	0.716	0.576
Improvements (CE-Loss decreases, Accuracy ıncreases)	**12.291%**	1.338%	1.514%	8.736%	4.255%	**14.764%**	0.880%	3.704%	3.616%	7.058%

use five real-world datasets[7]: *GIFGIF*, *HOTS*, *DOTA 2*, *LoL*, *IMDb 5000*. We use $L = 30, 15, 15, 20, 20$ and the learning rates of $10^{-3}, 10^{-3}, 10^{-2}, 10^{-2}, 10^{-2}$ respectively. We use four hidden layers for all modules. Each layer consists of $7M$ nodes for R and P, and $9M$ nodes for G.

Let us discuss each dataset. (1) *GIFGIF:* A crowdsourcing project.[7] We use the dataset pre-processed in [17]. A participant is presented with two images and asked to choose one which better describes a given emotion.[8] This dataset belongs to a special case of our interest as individual comparisons are concerned. We consider the emotion of happiness. We have 6,120 images and 106,886 samples. (2) *HOTS:* A collection of *HOTS* match records from 10/26/17 to 11/26/17 collected by *HOTS* logs.[7] Each match consists of two groups with five players each. The players choose heroes for each match out of a pool of 84. We choose high-quality matches only where all players are highly-skilled according to some available statistics. There are 26,486 records. (3) *DOTA 2:* A collection of *DOTA 2* match records.[7] Each match consists of two groups with five players each, and they choose heroes out of a pool of 113. There are 50,000 records. (4) *LoL:* A collection of *LoL* professional match records.[7] Two groups with five players each compete. The players choose heroes out of a pool of 140. There are 7,610 records. (5) *IMDb 5000:* A collection of meta-data for 5,000 movies.[7] Each movie has a score and is associated with keywords. To fit our purpose, we generate match records for movie pairs. We consider each movie as a group and its five keywords as its items. Given a pair, we declare a win for the one with a higher score. We have 8,021 keywords and 123,420 samples.

In addition to the cross entropy loss, we also consider another metric highly-relevant in practice: prediction accuracy. We declare it a win for a group if the estimate of its winning probability is above a certain threshold, which we set as 0.5 [18]. Thus, the prediction accuracy is expressed as follows[9]:

$$\frac{1}{|D_{\text{unobs}}|} \sum_{(A,B)\in D_{\text{unobs}}} y_{AB}\mathbb{I}_{\geq \frac{1}{2}}(\hat{y}_{AB}) + (1 - y_{AB})\mathbb{I}_{< \frac{1}{2}}(\hat{y}_{AB}). \tag{8}$$

Table 1 shows our result. We boldface the best performances and underline the second-best. The numbers in parentheses indicate the ranks among the algorithms being compared in a given setup of dataset and metric. Our framework consistently yields the top performances

[7] gifgif.media.mit.edu (*GIFGIF*); hotslogs.com/Info/API (*HOTS*); kaggle.com/devinanzelmo/dota-2-matches (*DOTA 2*); kaggle.com/chuckephron/leagueoflegends (*LoL*); kaggle.com/carolzhangdc/imdb-5000-movie-dataset (*IMDb 5000*).

[8] "neither" is also allowed, but we exclude such data.

[9] We define $\mathbb{I}_{\geq \frac{1}{2}}(x)$ as 1 if $x \geq \frac{1}{2}$ and as 0 otherwise. Similarly, $\mathbb{I}_{< \frac{1}{2}}(x)$ equals to 1 if $x < \frac{1}{2}$ and to 0 otherwise.

in all cases, while the others suffer from significantly inconsistent performances across datasets and/or metrics.

To verify that our design of modules R and P plays a critical role in achieving consistently high performances across various scenarios, we conduct ablation studies. As our baseline, we consider the case where we refine initialized estimates $\boldsymbol{w}^{(0)}$ using R and P once ($L = 1$), data augmentation is not applied, and the model parameters of R and P are not updated. Thus, module G takes as input coarse utility estimates and produces as output a group comparison outcome prediction. This baseline can be viewed as a naive neural network based approach without the key reward-penalty mechanisms.

Table 2 shows our result. The best performance improvements (boldfaced) reach up to 12.291% in cross entropy loss and 14.764% in prediction accuracy. The average performance improvements are 4.511% in cross entropy loss and 7.12% in prediction accuracy. Note that across all datasets and metrics, it is empirically demonstrated that the reward-penalty structure by modules R and P brings consistent improvements.

6. Conclusion

We explore the group match prediction problem where the goal is to predict the probability of one group preferred over the other given an unseen pair of groups, based on partially observed group comparison data. We develop a unified algorithmic framework that employs neural networks and show that it can yield consistently best performances compared to other state-of-the-art algorithms on multiple datasets across various domains.

Acknowledgments

This work was supported by the ICT R&D program of Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2020-0-00626).

References

[1] T. K. Huang, C. J. Lin, R. C. Weng, Generalized Bradley-Terry models and multi-class probability estimates, Journal of Machine Learning Research 7 (2006) 85–115.

[2] T. K. Huang, C. J. Lin, R. C. Weng, Ranking individuals by group comparisons, Journal of Machine Learning Research 9 (2008) 2187–2216.

[3] Y. Li, M. Cheng, K. Fujii, F. Hsieh, C.-J. Hsieh, Learning from group comparisons: Exploiting higher order interactions, Advances in Neural Information Processing Systems (2018) 4981–4990.

[4] R. Herbrich, T. Minka, T. Graepel, Trueskill™: A bayesian skill rating system, Advances in neural information processing systems (2007) 569–576.

[5] S. Negahban, S. Oh, D. Shah, Rank centrality: Ranking from pair-wise comparisons, Operations Research 65 (2016) 266–287.

[6] D. R. Hunter, MM algorithms for generalized Bradley-Terry models, Annals of Statistics (2004) 384–406.

[7] J. E. Menke, T. R. Martinez, A Bradley–Terry artificial neural network model for individual ratings in group competitions, Neural Computing and Applications 17 (2008) 175–186.

[8] R. A. Bradley, M. E. Terry, Rank analysis of incomplete block designs: I. the method of paired comparisons, Biometrika 39 (1952) 324–345.

[9] C. DeLong, N. Pathak, K. Erickson, E. Perrino, K. Shim, J. Srivastava, Teamskill: Modeling team chemistry in online multi-player games, Pacific-Asia Conference on Knowledge Discovery and Data Mining (2011) 519–531.

[10] M. Jang, S. Kim, C. Suh, S. Oh, Optimal sample complexity of M-wise data for top-K ranking, Advances in Neural Information Processing Systems (2017) 1686–1696.

[11] Y. Chen, J. Fan, C. Ma, K. Wang, Spectral method and regularized MLE are both optimal for top-K ranking, Annals of statistics 47 (2019) 2204–2235.

[12] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, Proceedings of the fourteenth international conference on artificial intelligence and statistics (2011) 315–323.

[13] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural networks 2 (1989) 359–366.

[14] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, International Conference on Machine Learning (2010) 249–256.

[15] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[16] O. Cooley, M. Kang, C. Koch, Threshold and hitting time for high-order connectedness in random hypergraphs, Electronic Journal of Combinatorics (2016) 2–48.

[17] L. Maystre, M. Grossglauser, Just sort it! A simple and effective approach to active preference learning, International Conference on Machine Learning (2017) 2344–2353.

[18] O. Delalleau, E. Contal, E. Thibodeau-Laufer, R. C. Ferrari, Y. Bengio, F. Zhang, Beyond skill rating: Advanced matchmaking in ghost recon online, IEEE Transactions on Computational Intelligence and AI in Games 4 (2012) 167–177.