Aggregating Quantitative Relative Judgments: From Social Choice to Interpretable Ranking Prediction

Anonymous submission

Abstract

Quantitative Relative Judgment Aggregation (QRJA) is a new research topic in (computational) social choice. In this model, agents judge the relative quality of different candidates, and we aim to obtain an aggregate assessment of the candidates' qualities. In QRJA, the judges do not necessarily have to be people with subjective opinions. For example, we can view a race as a "judgment" of the contestants' relative abilities, and we may wish to aggregate the results of multiple races. In this work, we explore the intriguing interplay between QRJA in a social choice context and its application to ranking prediction. We introduce new aggregation rules based on QRJA and study their structural and computational properties. We evaluate the proposed methods on data from various real races and show that QRJA-based methods offer effective and interpretable contest ranking predictions.

1 Introduction

In *voting theory*, each voter *ranks* a set of candidates, and a *voting rule* maps the vector of rankings to either a winning candidate or an aggregate ranking of all the candidates. There has been significant interaction between computer scientists interested in voting theory and the *learning-to-rank* community. This latter community is interested in problems such as learning how to rank webpages in response to a search query, or how to rank recommendations to a user (see, e.g., Liu (2009)). One may also wish to aggregate multiple rankings into a single one, for example aggregating the ranking results from different algorithms ("voters") into a single meta-ranking. The interests of the communities differ somewhat: e.g., the learning-to-rank community is less concerned about strategic voting and more about learning how much weight to assign to each voter. Still, a natural intersection point for these two communities is a model where there is a latent "true" ranking of the candidates, of which all the votes are just noisy observations. Given that, it is natural to try to estimate the true ranking given the votes. A procedure for this corresponds to a voting rule. (See, e.g., Young (1995); Conitzer and Sandholm (2005); Meila et al. (2007); Conitzer, Rognlie, and Xia (2009); Caragiannis, Procaccia, and Shah (2013); Soufiani, Parkes, and Xia (2014); Xia (2016), and Elkind and Slinko (2015) for an overview.)

Voting rules are but one type of mechanism in the broader field of *social choice*, which concerns itself with the broader problem of making decisions based on the preferences and opinions of multiple agents. Such opinions are not necessarily represented as rankings. For example, in *judgment aggregation* (for an overview, see Endriss (2015)), judges assess whether certain propositions are true or false. The observation that there are other types of input that are aggregated in social choice leads to the natural question of whether these also have analogous problems in statistics and machine learning (as is the case for ranking).

In this paper, we study this question for a relatively new model in social choice, which is the *quantitative* judgment aggregation problem (Conitzer, Brill, and Freeman 2015; Conitzer et al. 2016). Specifically, the goal is to aggregate *relative quantitative judgments*. Such judgments compare two candidates quantitatively, like "I believe that using 1 unit of gasoline is as bad as creating 2.7 units of landfill trash." As in the case of ranking, relative "judgments" can be produced by a process other than an agent reporting them. Consider, for example, a race in which contestant A finishes in 20:00 and contestant B in 30:00. Here, the "judgment" is that A is 10:00 faster than B. In a different race, their relative performance may be different. We may wish to aggregate these performances, in order to evaluate the contestants as well as to predict their relative performance in a future race. Given the different motivation, some of the aspects that are important in the social choice setting are less important to us in this context. For example, in a social choice setting, we may worry about agents strategically misreporting their judgments. In the contexts we are considering, this is not relevant because races are not strategic entities.

Specifically, we make the following contributions in this work. **(1).** We generalize the societal trade-offs problem in past work (Conitzer, Brill, and Freeman 2015; Conitzer et al. 2016) and propose quantitative relative judgment aggregation (QRJA). (Section 3) **(2).** We thoroughly characterize the computational aspects of ℓ_p QRJA (a subclass of QRJA) (Section 4). **(3).** We conduct extensive experiments to evaluate the performance of QRJA on the problem of predicting contest results using various real-world datasets (Section 5), demonstrating its effectiveness.

2 Motivating Examples

To better motivate our study and help readers understand the problem, we first consider simple mean/median approaches

for aggregating quantitative judgments and illustrate their limitations through three examples.

Example 1. When each race has some common "difficulty" (e.g. how hilly a marathon route is), if a contestant only participates in the "easy" races (or only the "hard" races), simply taking the median or mean of history performance will return biased estimates, as shown in Figure 1.

Contestant \ Race	Boston	New York	Chicago
Alice	4:00:00	4:10:00	3:50:00
Bob	4:11:00	4:18:00	4:01:00
Charlie			4:09:00

Figure 1: Bob finished 8 minutes faster than Charlie in the same race (Chicago), which suggests that Bob runs marathons faster than Charlie. However, if we look at the results of all three races and simply take the mean or median, Charlie's mean/median finishing time will be faster than Bob's. This is because, Charlie only participated in the Chicago race, where conditions were more favorable.

Example 2. If our data shows that Alice has beaten Bob before in some race, and Bob has beaten Charlie in another race, but we have never seen Alice and Charlie competing in the same race, we may want to predict that Alice is a faster runner than Charlie (see Figure 2). When comparing Alice and Charlie, the simple median/mean effectively ignores all the data on Bob. However, as indicated above, some of Bob's information can be useful for their comparison.

Contestant \ Race	Boston	New York	Chicago
Alice		4:10:00	
Bob	4:11:00	4:18:00	4:01:00
Charlie			4:09:00

Figure 2: The same set of results as in Figure 1, but with some data missing. If we only look at the data on Alice and Charlie, it is difficult to judge who is the faster runner. If anything, Charlie appears slightly faster. However, if we know Bob's results in these races, then we can use transitivity to infer that Alice is faster than Charlie.

Example 3. When the variance of the difficulty of races is much higher than that in the contestants' performance, taking the median will essentially focus on the result of a single race (with median difficulty) as illustrated in Figure 3.

QRJA takes care of all the issues above because it considers *relative* performance instead of absolute performance.

3 Problem Formulation

In this section, we formally define the Quantitative Relative Judgment Aggregation (QRJA) problem. We start with the definition of its input, quantitative relative judgments.

Definition 3.1 (Quantitative Relative Judgment)**.** *For a set of n candidates $N = \{1, 2, \ldots, n\}$, a **quantitative relative judgment** is a tuple $J = (a, b, y)$, where $a, b \in N$ and $y \in \mathbb{R}$, indicating a judgment that candidate a is better than candidate b by y units quantitatively.*

Contestant \ Race	Boston	New York	Chicago
Alice	4:00:00	4:10:00	3:50:00
Bob	4:11:00	4:18:00	4:01:00
Charlie	4:10:00	4:32:00	4:09:00

Figure 3: Simply taking the median throws away the useful information in the other two races if the common noise has high variance. Specifically, everyone's median race time is in Boston, so based on this, we would predict Charlie to be faster than Bob. However, if we consider the other two races as well it certainly seems that Bob is faster than Charlie.

The input of QRJA is a set of quantitative relative judgments to be aggregated. We model the aggregation result as a vector $\mathbf{x} \in \mathbb{R}^n$, where x_i is the single-dimensional evaluation of candidate i. The aggregation result should be consistent with the input judgments as much as possible, i.e., for a quantitative relative judgment (a, b, y), we want $|x_a - x_b - y|$ to be small. We use a loss function $f(|x_a - x_b - y|)$ to measure the inconsistency between the aggregation result and the input judgments. The function f should be convex and monotone. The aggregation result should minimize the weighted total loss. Formally, we have Definition 3.2.

Definition 3.2 (Quantitative Relative Judgment Aggregation (QRJA))**.** *Consider n candidates $N = \{1, \ldots, n\}$ and m quantitative relative judgments $\mathbf{J} = (J_1, \ldots, J_m)$ with weights $\mathbf{w} = (w_1, \ldots, w_m)$ where $J_i = (a_i, b_i, y_i)$. The **quantitative relative judgment aggregation** problem with convex and monotone loss function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ asks for a vector $\mathbf{x} \in \mathbb{R}^n$ that minimizes $\sum_{i=1}^{m} w_i f(|x_{a_i} - x_{b_i} - y_i|)$.*

Previous work (Conitzer, Brill, and Freeman 2015; Conitzer et al. 2016; Zhang, Cheng, and Conitzer 2019) studied a special case of the QRJA problem where $f(x) = x$. In this work, we broaden the scope and consider QRJA with convex and monotone loss functions.

To analyze QRJA, we first note that the convexity of the function f guarantees that the optimization program is convex. Therefore, we can use standard convex optimization methods like gradient descent or ellipsoid method to solve the program in polynomial time. However, general-purpose convex optimization methods are usually not efficient enough in practice when the numbers of candidates n and judgments m are both large. Therefore, for the rest of the paper, we mainly focus on ℓ_p QRJA, i.e., QRJA with $f(x) = x^p$ where $p \geq 1$, which allows for a more thorough study and more efficient algorithms. We also provide an axiomatic characterization of ℓ_p QRJA in Appendix C.

4 Theoretical Aspects of ℓ_p QRJA

In this section, we study the theoretical aspects of ℓ_p QRJA. We first show that when $p > 1$, ℓ_p QRJA can be solved with ℓ_p-regression techniques more efficiently than ellipsoid algorithm and gradient descent with explicit polynomial runtime that depends on p (Section 4.1). Next we present faster algorithms for ℓ_p QRJA tailored for the special cases of

$p = 1$ and $p = 2$ (Section 4.2). Moreover, we show that if $1 \le p \le 2$ and the number of judgements m is much larger than the number of candidates n, with a small approximation error we can reduce the number of judgments to $\widetilde{O}(n)$ by subsampling (Section 4.3). Finally, in contrast to the above positive results, we show that when $p < 1$, ℓ_p QRJA is NP-hard by reducing from Max-Cut (Section 4.4).

4.1 Polynomial-Time Algorithms When $p > 1$

We show that ℓ_p QRJA can be solved with ℓ_p-regression techniques in polynomial time when $p > 1$ by reducing ℓ_p QRJA to ℓ_p-regression in this subsection. We start with the definition of the ℓ_p-regression problem below.

Definition 4.1 (ℓ_p-Regression)**.** *For a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, a vector $\mathbf{z} \in \mathbb{R}^m$ and a real number $p > 0$, the ℓ_p-**regression** problem asks to find a vector $\mathbf{x}$ that minimizes $\|\mathbf{Ax} - \mathbf{z}\|_p^p$.*

Reduction from ℓ_p QRJA to ℓ_p-regression. Given an ℓ_p QRJA instance with n candidates and m quantitative relative judgments $\mathbf{J} = \{J_1, \ldots, J_m\}$ ($J_i = (a_i, b_i, y_i)$) with weights $\mathbf{w} = \{w_1, \ldots, w_m\}$, we construct a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ and a vector $\mathbf{z} \in \mathbb{R}^m$ as follows:

$$A_{i,j} = \begin{cases} \sqrt[p]{w_i} & \text{if } j = a_i \\ -\sqrt[p]{w_i} & \text{if } j = b_i \\ 0 & \text{otherwise} \end{cases}, \quad z_i = \sqrt[p]{w_i} y_i.$$

The correctness and the runtime of this reduction are stated in Lemma 4.2. Lemma 4.2 is proved in Appendix A.1.

Lemma 4.2. *For any $p > 0$, there is a reduction from ℓ_p QRJA to ℓ_p-regression that runs in $O(m)$.*

Lemma 4.2 allows us to use ℓ_p-regression techniques to solve ℓ_p QRJA. ℓ_p-regression is a fundamental problem that has been studied extensively, with many applications in data analysis and machine learning. For example, see (Nesterov and Nemirovskii 1994; Xue and Ye 2000; Drineas, Mahoney, and Muthukrishnan 2006; Dasgupta et al. 2009; Shalev-Shwartz and Tewari 2011; Clarkson and Woodruff 2013; Meng and Mahoney 2013; Cohen and Peng 2015; Yang et al. 2016). In particular, we invoke the following algorithmic result from (Bubeck et al. 2018).

Lemma 4.3 (Bubeck et al. 2018)**.** *For any $p > 1$, given $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{z} \in \mathbb{R}^n$, we can compute a vector $\mathbf{x}$ such that $\|\mathbf{Ax} - \mathbf{z}\|_p \le \min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{z}\|_p + \varepsilon$ in time*

$$\widetilde{O}_p\left(\left(\mathrm{nnz}(\mathbf{A})\left(1 + \sqrt{\frac{n}{m^{1-2\gamma}}}\right) + m^\gamma n^2 + n^\omega\right)\log\frac{\|\mathbf{z}\|_2^p}{\varepsilon}\right),$$

where $\mathrm{nnz}(\mathbf{A})$ is the number of non-zero entries in $\mathbf{A}$, $\gamma = |\frac{1}{2} - \frac{1}{p}|$, and ω is the matrix multiplication exponent.[1]

Since the dependence on ε is $\log(1/\varepsilon)$ in Lemma 4.3, ℓ_p-regression can be solved up to machine precision in polynomial time. We immediately have the following theorem.

[1]Throughout the paper, we write $\widetilde{O}(f)$ for $O(f \operatorname{poly}\log f)$, and we use $O_p(f)$ to hide factors related to p.

Theorem 4.4. *For any $p > 1$, ℓ_p QRJA can be solved in*

$$\widetilde{O}_p\left(\left(m + n^{\frac{1}{2}} m^{\gamma+\frac{1}{2}} + m^\gamma n^2 + n^\omega\right)\log\frac{mW}{\varepsilon}\right),$$

where ε is the machine precision, $W = \max_i\{y_i, w_i\}$, $\gamma = |\frac{1}{2} - \frac{1}{p}|$, and ω is the matrix multiplication exponent.

Despite the ongoing effort of improving the running time for ℓ_p-regression for general values of p, in practice it is common to choose $p = 1$ or $p = 2$, because ℓ_2 regression admits faster computation and ℓ_1 regression is often robust to outliers (Wang and Scott 1994). Therefore, in the next section, we study the special cases of ℓ_p QRJA where $p \in \{1, 2\}$, and present faster algorithms for them.

4.2 Faster Algorithms When $p \in \{1, 2\}$

In this subsection, we show that ℓ_p QRJA admits faster algorithms when $p = \{1, 2\}$.

Algorithm for ℓ_1 QRJA. When $p = 1$, the overall loss function of QRJA is a sum of absolute values of some linear terms. We can therefore formulate ℓ_1 QRJA as a linear program (LP) as below (Zhang, Cheng, and Conitzer 2019).

$$\begin{array}{lll} \text{minimize} & \sum_{i=1}^m w_i\left(z_i^+ + z_i^-\right) & \\ \text{subject to} & z_i^+ \ge x_{a_i} - x_{b_i} - y_i & \forall i \in [m] \\ & z_i^- \ge y_i + x_{b_i} - x_{a_i} & \forall i \in [m] \\ & z_i^+ \ge 0, z_i^- \ge 0 & \forall i \in [m] \\ & x_i \in \mathbb{R} & \forall i \in [n] \end{array}$$

For this LP, (Zhang, Cheng, and Conitzer 2019) gave a faster algorithm than using general-purpose LP solvers.

Lemma 4.5 (Zhang, Cheng, and Conitzer 2019)**.** *There is a reduction from ℓ_1 QRJA to Minimum Cost Flow with $O(n)$ vertices and $O(m)$ edges in $O(T_{\mathrm{SSSP}}(n, m, W))$ time, where $T_{\mathrm{SSSP}}(n, m, W)$ is the time required to solve Single-Source Shortest Path with negative weights on a graph with n vertices, m edges, and maximum absolute distance W.*

Using the SSSP algorithm in (Bernstein, Nanongkai, and Wulff-Nilsen 2022) and the minimum cost flow algorithm in (Chen et al. 2022), we then have Corollary 4.6.

Corollary 4.6. *ℓ_1 QRJA can be solved in $\widetilde{O}(m\log^8 n \log W + m^{1+o(1)}\log^2 W)$ time, where $W = \max_{i\in N}\{w_i, y_i\}$.*

Algorithm for ℓ_2 QRJA. Recall Lemma 4.2, ℓ_2 QRJA is reducible to ℓ_2-regression. The ℓ_2-regression problem is commonly known as the (linear) Least-Square Regression problem. We exploit the special structure of ℓ_2 QRJA, and relate it to the problem of solving Laplacian linear systems (Spielman and Teng 2004; Koutis, Miller, and Peng 2011; Cohen et al. 2014). Formally, we will prove the following Theorem 4.7 in Appendix A.2.

Theorem 4.7. *ℓ_2 QRJA is equivalent to solving a Laplacian linear system. In particular, ℓ_2 QRJA can be solved in time $\widetilde{O}(m\sqrt{\log n})$ using the Lapalcian solver in (Cohen et al. 2014).*

4.3 Subsampling Judgments When $p \in [1, 2]$

In this subsection, we show that for $p \in [1, 2]$, we can reduce the number of judgments by subsampling using Lewis weights with a small approximation error. Specifically, we consider Algorithm 1 for subsampling.

Algorithm 1: Subsampling Judgments

Input: ℓ_p QRJA instance $(n, m, \mathbf{J}, \mathbf{w})$, subsample count $M \in \mathbb{N}$ and subsampling weights $\mathbf{s} \in \mathbb{R}^m$
Output: ℓ_p QRJA instance $(n', m', \mathbf{J}', \mathbf{w}')$

1: Let $q_i \leftarrow \frac{s_i}{\sum_{j=1}^m s_j}$ for each $i \in \{1, 2, \ldots, m\}$.
2: **for** $i \in \{1, 2, \ldots, M\}$ **do**
3: Sample $x \in \{1, 2, \ldots, m\}$ with probability q_x.
4: Let $J'_i \leftarrow J_x$ and $w'_i \leftarrow \frac{w_x}{M \cdot q_x}$.
5: Let $n' \leftarrow n$ and $m' \leftarrow M$.
6: **return** $(n', m', \mathbf{J}', \mathbf{w}')$.

At a high level, Algorithm 1 takes an ℓ_p QRJA instance, a parameter M and a vector $\mathbf{s} \in \mathbb{R}^m$ as input. It then samples M judgments from the input instance with probability proportional to $\mathbf{s}$, and outputs a new ℓ_p QRJA instance with the sampled judgments. The weight of any judgment in the output instance is divided by its expected number of occurrences in the output instance, so that the expected total weight of any judgment is preserved after subsampling.

To obtain theoretical guarantees of the subsampled instance, we need to choose the vector $\mathbf{s}$ carefully. We use the Lewis weights mentioned in (Cohen and Peng 2015) for this purpose. The Lewis weights are defined as follows.

Definition 4.8 (Lewis weights for ℓ_p QRJA)**.** *For an ℓ_p QRJA instance $(n, m, \mathbf{J}, \mathbf{w})$, define matrix $\mathbf{A} \in \mathbb{R}^{m \times (n+1)}$*

$$A_{i,j} = \begin{cases} \sqrt[p]{w_i} & \text{if } j = a_i \\ -\sqrt[p]{w_i} & \text{if } j = b_i \\ -\sqrt[p]{w_i} y_i & \text{if } j = n+1 \\ 0 & \text{otherwise.} \end{cases}$$

*The **Lewis weights** for this ℓ_p QRJA instance is defined as the unique vector $\mathbf{s} \in \mathbb{R}^m$ such that for each $i \in \{1, 2, \ldots, m\}$,*

$$\mathbf{a}_i \left(\mathbf{A}^\top \mathbf{S}^{1-2/p} \mathbf{A} \right)^{-1} \mathbf{a}_i^\top = s_i^{2/p},$$

where $\mathbf{S} = \operatorname{diag}(\mathbf{s})$ and $\mathbf{a}_i$ is the i-th row of $\mathbf{A}$.

The existence and uniqueness of such weights are first shown in (Lewis 1978). In (Cohen and Peng 2015), the authors show that for $p \in [1, 2]$, the Lewis weights can be computed in $O(\text{nnz}(\mathbf{A}) + n^{\omega+\theta}) = O(m + n^{\omega+\theta})$ time with a small error, where ω is the matrix multiplication exponent and $\theta > 0$ is any constant. We will invoke the Matrix Concentration Bound established in (Cohen and Peng 2015) to show the following theorem in Appendix A.4.

Theorem 4.9. *Fix $p \in [1, 2]$ and error ε, there exists a constant C such that for any ℓ_p QRJA instance $(n, m, \mathbf{J}, \mathbf{w})$, if we run Algorithm 1 with $M = Cn \log n \log\log^2 n$ and $\mathbf{s}$ being the Lewis weights for $(n, m, \mathbf{J}, \mathbf{w})$, then the output instance has an optimal solution that is at most $(1 + \varepsilon)$-approximate in the original instance.*

Theorem 4.9 shows when $m = \omega(n \log n \log\log^2 n)$, we can reduce it to $O(n \log n \log\log^2 n)$ with a small error in $O(m + n^{\omega+\theta})$ time for $p \in [1, 2]$.

4.4 NP-Hardness When $p < 1$

In this subsection, we study ℓ_p QRJA when $p < 1$. Note that in this case, the function $f(x) = x^p$ is no longer convex. We show that ℓ_p QRJA is NP-hard by reducing from Max-Cut.

Definition 4.10 (Max-Cut)**.** *For an undirected graph $G = (V, E)$, Max-Cut asks for a partition of V into two sets S and T that the number of edges between S and T is maximized.*

Reduction from Max-Cut to ℓ_p QRJA. Given a Max-Cut instance on an undirected graph $G = (V, E)$, let $n = |V|, m = |E|$, $w_2 = \frac{2n}{1-p} + 1$, and $w_1 = nw_2 + 1$. We will construct an ℓ_p QRJA instance with $n + 2$ candidates $V \cup \{v^{(s)}, v^{(t)}\}$ and $O(n + m)$ quantitative relative judgments. Specifically, we add the following judgments:

- $(v^{(t)}, v^{(s)}, 1)$ with weight w_1.
- $(v^{(s)}, u, 0)$ with weight w_2 for each $u \in V$.
- $(v^{(t)}, u, 0)$ with weight w_2 for each $u \in V$.
- $(u, v, 1), (v, u, 1)$ with weight 1 for each $(u, v) \in E$.

In Appendix A.3, we will prove that the Max-Cut instance has a cut of size at least k if and only if the constructed ℓ_p QRJA instance has a solution with loss at most $nw_2 + 2(m - k) + k2^p$, which implies the following hardness result.

Theorem 4.11. *For any $p < 1$, there exists a constant $c > 0$ such that it is NP-hard to approximate ℓ_p QRJA within a multiplicative factor of $\left(1 + \frac{c}{n^2}\right)$.*

5 Experiments

We conduct experiments on real-world datasets to compare the performance of QRJA with existing methods. We focus on a specific application of QRJA, namely making predictions of contest results, where we treat the contestants as candidates and the results of the contests as judgments. All experiments are done on a server with 56 cores and 504G RAM. All source code required for conducting experiments is included in the supplementary material.

5.1 Performance Experiments Setup

Datasets. We consider types of contests where events are reasonably frequent (so it makes sense to make predictions for future events based on past ones), and contest results contain numerical scores in addition to rankings. Specifically, we use the four datasets listed below. In Appendix B, we include additional experiments on three more datasets.

- **Chess.** This dataset contains the results of the Tata Steel Chess Tournament (https://tatasteelchess.com/, also historically known as the Hoogovens Tournament or the Corus Chess Tournament) from 1983 to 2023[2]. Each contest is typically a round-robin tournament among 10 to 14

[2]We choose the time frame of our datasets to be longer than the active period of most contestants to emphasize that contestants come and go, but their past performance could help the prediction.

contestants. A contestant's numerical score is that contestant's number of wins in the tournament. There are in total 80 contests and 408 contestants in this dataset.

- **F1.** This dataset contains the results of Formula 1 races (https://www.formula1.com/) from 1950 to 2023. In each contest, we take all contestants who complete the whole race. There are around 7 such contestants in each contest. A contestant's numerical score is the negative of his/her finishing time (in seconds). There are 878 contests and 261 contestants in total in this dataset.
- **Marathon.** This dataset contains the results of the Boston and New York Marathons from 2000 to 2023. We use data from https://www.marathonguide.com/, which publishes results of all major marathon events. Each contest usually involves more than 20000 contestants. We take the 100 top-ranked contestants in each contest as our dataset. A contestant's numerical score is the negative of that contestant's finishing time (in seconds). There are 44 contests and 2984 contestants in total in this dataset.
- **Codeforces.** This dataset contains the results of Codeforces (https://codeforces.com), a website hosting frequent online programming contests, from 2010 to 2023 (Codeforces Round 875). We consider only Division 1 contests, where only more skilled contestants can participate. Each contest involves around 700 contestants. We take the 100 top-ranked contestants in each contest as our dataset. A contestant's numerical score is that contestant's points in that contest. There are 327 contests and 5338 contestants in total in this dataset.

Evaluation Metrics. For all the datasets we use, contests are naturally ordered chronologically. We use the results of the first $i-1$ contests to predict the results of the i-th contest. We apply the following two metrics to evaluate the prediction performance of different algorithms.

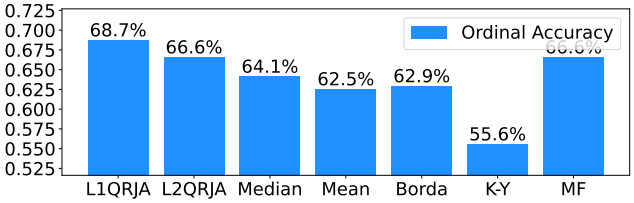
- **Ordinal Accuracy.** This metric measures the percentage of correct relative ordinal predictions. For each contest, we predict the ordinal results of all pairs of contestants that (i) have both appeared before and (ii) have different numerical scores in the current contest. We compute the percentage of correct predictions among all such pairs.
- **Quantitative Loss.** This metric measures the average absolute error of relative quantitative predictions. For each contest, we predict the difference in numerical scores of all pairs of contestants that have both appeared before. We then compute the quantitative loss as the average absolute error of the predictions among all such pairs. We normalize this number by the quantitative loss of the trivial prediction that always predicts 0 for all pairs.

Implementation. We have implemented both ℓ_1 and ℓ_2 QRJA in Python. We use Gurobi (Gurobi Optimization, LLC 2023) and NetworkX (Hagberg, Swart, and S Chult 2008) to implement ℓ_1 QRJA and SciPy (Jones, Oliphant, and Peterson 2014) to implement ℓ_2 QRJA. To transform the contest standings into a QRJA instance, we construct a quantitative relative judgment $J = (a, b, y)$ for each contest and each pair of contestants (a, b) with y being the score difference between a and b in that contest. We set all weights to 1 to ensure fair comparison with benchmarks.

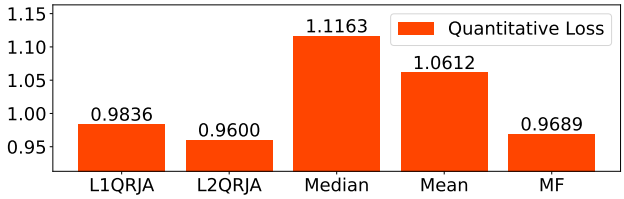
Benchmarks. We evaluate ℓ_1 and ℓ_2 QRJA against several benchmark algorithms. Specifically, we consider the natural one-dimensional aggregation methods Mean and Median, social choice methods Borda and Kemeny-Young, and a common method for prediction, matrix factorization. We describe how we apply these methods to our setting below.

- **Mean** and **Median.** For every contestant in the training set, we take the mean or median of that contestant's scores in training contests. We then make predictions based on differences between these mean or median scores. In one-dimensional environments like ours, means and medians are considered to be among the best imputation methods for various tasks (see, e.g., Engels and Diehr, 2003, Shrive et al., 2006).
- **The Borda rule.** The Borda rule is a voting rule that only uses rankings as input and only produces a ranking as output. We use a normalized version of the Borda rule. The i-th ranked contestant in contest j receives $1-\frac{2(i-1)}{n_j-1}$ points, where n_j is the number of contestants in the contest. The aggregated ranking result is obtained by sorting the contestants by their total numbers of points.
- **The Kemeny-Young rule.** (Kemeny 1959; Young and Levenglick 1978; Young 1988)**.** The Kemeny-Young rule is also a voting method that takes multiple (partial) rankings of the contestants as input and only produces a ranking as output. Specifically, it outputs a ranking that minimizes the number of *disagreements* on pairs of contestants with the input rankings. Finding the optimal Kemeny-Young ranking is known to be NP-hard (Bartholdi, Tovey, and Trick 1989). In our experiments, we use Gurobi to solve the mixed-integer program formulation of the Kemeny-Young rule given in (Conitzer, Davenport, and Kalagnanam 2006). As this method is still computationally expensive and can only scale to hundreds of contestants, for each contest we predict, we only keep the contestants within that specific contest and discard all other contestants to run Kemeny-Young.
- **Matrix Factorization (MF).** Matrix factorization is a class of algorithms that take in a matrix with missing entries and output a prediction of the whole matrix. In the context of our experiments, we can view each contestant as a row and each contest as a column. The score of a contestant in a contest is the entry in the corresponding row and column. We implement several variants of MF and include the results of one of them (Koren, Bell, and Volinsky 2009) in this section as other variants have comparable or worse performance. For implementation details and results of other variants, see Appendix B.3.

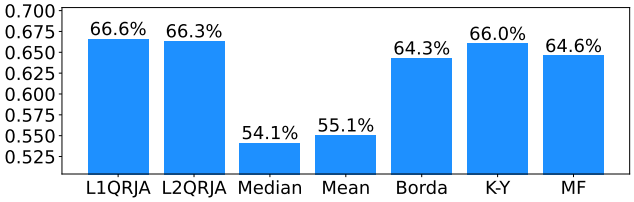
Many other, related approaches deserve mention in this context. But we do not include them in the benchmarks because they do not exactly fit our setting or motivation. For example, the seminal Elo rating system (Elo 1978) as well as many other methods (Maher 1982; Karlis and Ntzoufras 2008; Guo et al. 2012; Hunter et al. 2004) can all predict the results of pairwise matches in, e.g., chess and football.



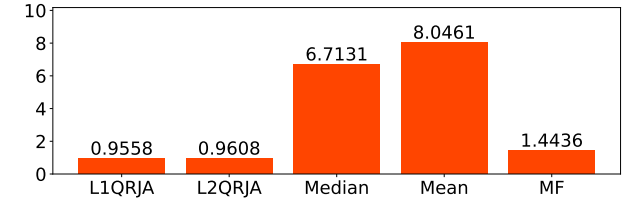
(a) Ordinal accuracy on Chess



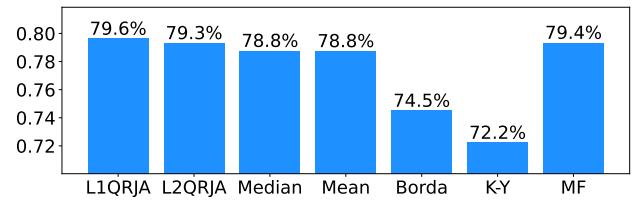
(b) Quantitative loss on Chess



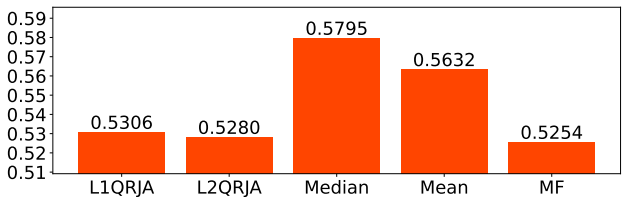
(c) Ordinal accuracy on F1



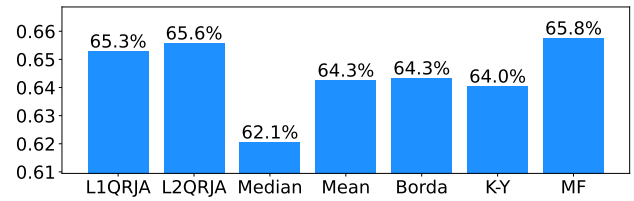
(d) Quantitative loss on F1



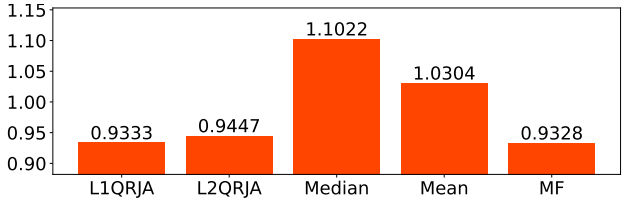
(e) Ordinal accuracy on Marathon



(f) Quantitative loss on Marathon

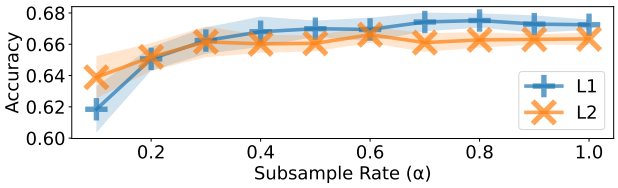


(g) Ordinal accuracy on Codeforces

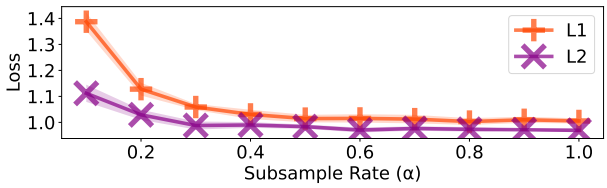


(h) Quantitative loss on Codeforces

Figure 4: Ordinal accuracy and quantitative loss of the algorithms on all four datasets. Error bars are not shown here as the algorithms are deterministic. The results show that both versions of QRJA perform consistently well across the tested datasets.



(a) ℓ_1 and ℓ_2 QRJA's ordinal accuracy on Chess



(b) ℓ_1 and ℓ_2 QRJA's quantitative loss on Chess

Figure 5: The performance of ℓ_1 and ℓ_2 QRJA on Chess after subsampling judgments using Algorithm 1 with equal weights for all judgments. The subsample rate α means $M = \lfloor \alpha m \rfloor$ in Algorithm 1. Error bars indicate the standard deviation. The results show that Algorithm 1 can reduce the number of judgments to a factor of 0.4 with a minor performance loss on Chess.

However, they are not originally designed for predicting the results of contests with more than two contestants.

5.2 Performance Experiment Results

The complete experimental results of all algorithms on the four datasets are shown in Fig. 4. Note that Borda and Kemeny-Young do not make quantitative predictions, so they are not included in Figs. 4b, 4d, 4f and 4h.

The performance of QRJA. As shown in Fig. 4, both versions of QRJA perform consistently well across the tested datasets. They are always among the best algorithms in terms of both ordinal accuracy and quantitative loss.

The performance of Mean and Median. In terms of ordinal accuracy, Mean and Median do well on Marathon, but are not among the best algorithms on other datasets, especially on F1 (for both) and Codeforces (for Median).

Moreover, in terms of quantitative loss, Mean and Median are never among the best algorithms. This shows that although Mean and Median occasionally perform well (e.g., on Marathon; we also find Median performs the best on another dataset in Appendix B.1), they fail in other cases.

The performance of Borda and Kemeny-Young. Borda and Kemeny-Young do not make quantitative predictions, so we only compare them with other algorithms in terms of ordinal accuracy. As shown in Fig. 4, Borda and Kemeny-Young perform very well on F1, but are not among the best algorithms on other datasets. By only using rankings as input, Borda and Kemeny-Young are more robust on datasets where contestants' performance varies a lot. However, they fail to utilize the quantitative information on other datasets.

The performance of Matrix Factorization (MF). MF works well across the tested datasets in terms of both metrics. In all of our four datasets, it has performance comparable to QRJA. The advantage of QRJA over MF is the simplicity and interpretability of its model. Additionally, we observe in Appendix B.1 that ℓ_1 QRJA is more robust to large variance in contestants' performance than MF.

Summary of experimental results. In summary, both MF and QRJA are never significantly worse than the best-performing algorithm on any of the tested datasets, unlike the other benchmark methods. QRJA additionally offers a simple and interpretable model. This shows that QRJA is an effective and interpretable method for making predictions on contest results.

5.3 Subsampling Experiments

We also conduct experiments to test the performance of our subsampling algorithm (Algorithm 1), which speeds up the (approximate) computation of QRJA on large datasets. In the experiments, we specify the subsample rate α and let $M = \lfloor \alpha m \rfloor$ in Algorithm 1. We do not choose to use Lewis weights in the experiments because the theoretical algorithm is not easy to implement and the asymptotic guarantee requires a large number of judgments to hold, and therefore it is less meaningful for practical purposes. Instead, We use equal weights for all judgments, which is less costly and much simpler, and thus more likely to work well in practice.

Experiment setup. We run ℓ_1 and ℓ_2 QRJA with instances subsampled by Algorithm 1 on the datasets. For each $\alpha = \{0.1, 0.2, \ldots, 1.0\}$, we run ℓ_1 and ℓ_2 QRJA 10 times and report their average performance on both metrics with error bars. Due to the space constraints, we only show the results on Chess in Fig. 5 in this section. The results on other datasets are deferred to Appendix B.2.

Experiment results. As is shown in Fig. 5, with equal weights for all judgments, Algorithm 1 can reduce the number of judgments without significantly hurting the performance of ℓ_1 and ℓ_2 QRJA as long as the sampling rate α is not too small (≥ 0.4 for Chess). This shows that Algorithm 1 is a practical algorithm for subsampling judgments in QRJA. We also note that as the experiments show, ℓ_2 QRJA is more robust to subsampling than ℓ_1 QRJA.

6 Related Work

Random utility models. Random utility models (Fahandar, Hüllermeier, and Couso (2017); Zhao, Villamil, and Xia (2018)) explicitly reason about the contestants being numerically different from each other, e.g., one contestant is generally 1.1 times as fast as another. However, they are still designed for settings in which the only input data we have is ranking data, rather than numerical data such as finishing times. Moreover, random utility models generally do not model common factors, such as a given race being tough and therefore resulting in higher finishing times for *everyone*.

Matrix completion. Richer models considered in recommendation systems appear too general for the scenarios we have in mind. Matrix completion (Rennie and Srebro 2005; Candès and Recht 2009) is a popular approach in collaborative filtering, where the goal is to recover missing entries given a partially-observed low-rank matrix. While using higher ranks may lead to better predictions, we want to model contestants in a single-dimensional way, which is necessary for interpretability purposes (the single parameter being interpreted as the "quality" of the contestant).

Preference learning. In preference learning, we train on a subset of items that have preferences toward labels and predict the preferences for all items (see, e.g., Pahikkala et al. (2009)). One high-level difference is that preference learning tends to use existing methodologies in machine learning to learn rankings. In contrast, our methods (as well as those in previous work Conitzer, Brill, and Freeman (2015); Conitzer et al. (2016)) are social-choice-theoretically well motivated. In addition, our methods are designed for quantitative predictions, while the main objective of preference learning is to learn ordinal predictions (e.g., rankings).

Elo and TrueSkill. Empirical methods, such as the Elo rating system (Elo 1978) and Microsoft's TrueSkill (Herbrich, Minka, and Graepel 2006), have been developed to maintain rankings of players in various forms of games. Unlike QRJA, these methods focus more on the online aspects of the problem, i.e., how to properly update scores after each game. While under specific statistical assumptions, these methods can in principle predict the outcome of a future game, they are not designed for making ordinal or quantitative predictions in their nature.

7 Conclusion

In this paper, we conduct a thorough investigation of the problem of aggregating relative quantitative judgments. We formulate the QRJA problem and focus on a subclass, ℓ_p QRJA. We show that ℓ_p QRJA is NP-hard for $p < 1$, and give polynomial-time algorithms for $p > 1$ and faster algorithms for $p \in \{1, 2\}$. We also give an algorithm for subsampling judgments to speed up its approximate computation when $p \in \{1, 2\}$. Finally, we conduct experiments on real-world datasets to show that QRJA is an effective and interpretable method for the application of predicting contest results. We hope our work can inspire future research on interpretable and transparent methods for judgment aggregation.

References

Bartholdi, J.; Tovey, C. A.; and Trick, M. A. 1989. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and welfare*, 6: 157–165.

Bernstein, A.; Nanongkai, D.; and Wulff-Nilsen, C. 2022. Negative-weight single-source shortest paths in near-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, 600–611. IEEE.

Bubeck, S.; Cohen, M. B.; Lee, Y. T.; and Li, Y. 2018. An homotopy method for ℓ_p regression provably beyond self-concordance and in input-sparsity time. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing (STOC)*, 1130–1137. ACM.

Candès, E. J.; and Recht, B. 2009. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics*, 9(6): 717–772.

Caragiannis, I.; Procaccia, A. D.; and Shah, N. 2013. When do noisy votes reveal the truth? In *Proceedings of the fourteenth ACM conference on Electronic commerce*, 143–160. ACM.

Chen, L.; Kyng, R.; Liu, Y. P.; Peng, R.; Gutenberg, M. P.; and Sachdeva, S. 2022. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, 612–623. IEEE.

Clarkson, K. L.; and Woodruff, D. P. 2013. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing (STOC)*, 81–90. ACM.

Cohen, M. B.; Kyng, R.; Miller, G. L.; Pachocki, J. W.; Peng, R.; Rao, A. B.; and Xu, S. C. 2014. Solving SDD linear systems in nearly $m \log^{1/2} n$ time. In *Proc. 46th Annual ACM Symposium on Theory of Computing (STOC)*, 343–352.

Cohen, M. B.; and Peng, R. 2015. ℓ_p Row Sampling by Lewis Weights. In *Proceedings of the 47th annual ACM Symposium on Theory of Computing (STOC)*, 183–192. ACM.

Conitzer, V.; Brill, M.; and Freeman, R. 2015. Crowdsourcing Societal Tradeoffs. In *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 1213–1217. Istanbul, Turkey.

Conitzer, V.; Davenport, A.; and Kalagnanam, J. 2006. Improved bounds for computing Kemeny rankings. In *AAAI*, volume 6, 620–626.

Conitzer, V.; Freeman, R.; Brill, M.; and Li, Y. 2016. Rules for Choosing Societal Tradeoffs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 460–467. Phoenix, AZ, USA.

Conitzer, V.; Rognlie, M.; and Xia, L. 2009. Preference Functions That Score Rankings and Maximum Likelihood Estimation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, 109–115. Pasadena, CA, USA.

Conitzer, V.; and Sandholm, T. 2005. Common Voting Rules as Maximum Likelihood Estimators. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 145–152. Edinburgh, UK.

Dasgupta, A.; Drineas, P.; Harb, B.; Kumar, R.; and Mahoney, M. W. 2009. Sampling algorithms and coresets for ℓ_p regression. *SIAM Journal on Computing*, 38(5): 2060–2078.

Drineas, P.; Mahoney, M. W.; and Muthukrishnan, S. 2006. Sampling algorithms for ℓ_2 regression and applications. In *Proceedings of the 17th annual ACM-SIAM Symposium on Discrete Algorithm (SODA)*, 1127–1136. SIAM.

Elkind, E.; and Slinko, A. 2015. Rationalizations of Voting Rules. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*, chapter 8. Cambridge University Press.

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

Endriss, U. 2015. Judgment Aggregation. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*, chapter 17. Cambridge University Press.

Engels, J. M.; and Diehr, P. 2003. Imputation of missing longitudinal data: a comparison of methods. *Journal of clinical epidemiology*, 56(10): 968–976.

Fahandar, M. A.; Hüllermeier, E.; and Couso, I. 2017. Statistical inference for incomplete ranking data: the case of rank-dependent coarsening. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1078–1087. JMLR. org.

Guo, S.; Sanner, S.; Graepel, T.; and Buntine, W. 2012. Score-based bayesian skill learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 106–121. Springer.

Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual.

Hagberg, A.; Swart, P.; and S Chult, D. 2008. Exploring network structure, dynamics, and function using NetworkX. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States).

Herbrich, R.; Minka, T.; and Graepel, T. 2006. TrueSkill$^{\text{TM}}$: A Bayesian Skill Rating System. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, 569–576.

Hunter, D. R.; et al. 2004. MM algorithms for generalized Bradley-Terry models. *The annals of statistics*, 32(1): 384–406.

Jones, E.; Oliphant, T.; and Peterson, P. 2014. SciPy: open source scientific tools for Python.

Karlis, D.; and Ntzoufras, I. 2008. Bayesian modelling of football outcomes: using the Skellam's distribution for the goal difference. *IMA Journal of Management Mathematics*, 20(2): 133–145.

Kemeny, J. 1959. Mathematics without Numbers. *Daedalus*, 88: 575–591.

Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37.

Koutis, I.; Miller, G. L.; and Peng, R. 2011. A Nearly-$m \log n$ Time Solver for SDD Linear Systems. In *Proc. 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, 590–598.

Lewis, D. 1978. Finite dimensional subspaces of L_p. *Studia Mathematica*, 63(2): 207–212.

Liu, T.-Y. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval*, 3(3): 225–231.

Maher, M. J. 1982. Modelling association football scores. *Statistica Neerlandica*, 36(3): 109–118.

Meila, M.; Phadnis, K.; Patterson, A.; and Bilmes, J. 2007. Consensus Ranking Under the Exponential Model. In *Proceedings of the 23rd Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, 285–294. Vancouver, BC, Canada.

Meng, X.; and Mahoney, M. W. 2013. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing (STOC)*, 91–100. ACM.

Nesterov, Y.; and Nemirovskii, A. 1994. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM.

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.

Rennie, J. D. M.; and Srebro, N. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning*, 713–719.

Shalev-Shwartz, S.; and Tewari, A. 2011. Stochastic methods for ℓ_1-regularized loss minimization. *Journal of Machine Learning Research*, 12(Jun): 1865–1892.

Shrive, F. M.; Stuart, H.; Quan, H.; and Ghali, W. A. 2006. Dealing with missing data in a multi-question depression scale: a comparison of imputation methods. *BMC medical research methodology*, 6(1): 57.

Soufiani, H. A.; Parkes, D. C.; and Xia, L. 2014. A statistical decision-theoretic framework for social choice. In *Advances in Neural Information Processing Systems*, 3185–3193.

Spielman, D. A.; and Teng, S. 2004. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proc. 36th Annual ACM Symposium on Theory of Computing (STOC)*, 81–90.

Wang, F. T.; and Scott, D. W. 1994. The L 1 method for robust nonparametric regression. *Journal of the American Statistical Association*, 89(425): 65–76.

Xia, L. 2016. Quantitative Extensions of the Condorcet Jury Theorem with Strategic Agents. In *AAAI*, 644–650.

Xue, G.; and Ye, Y. 2000. An efficient algorithm for minimizing a sum of p-norms. *SIAM Journal on Optimization*, 10(2): 551–579.

Yang, J.; Chow, Y.-L.; Ré, C.; and Mahoney, M. W. 2016. Weighted SGD for ℓ_p regression with randomized preconditioning. In *Proceedings of the 27th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 558–569. SIAM.

Young, H. P. 1988. Condorcet's Theory of Voting. *American Political Science Review*, 82: 1231–1244.

Young, H. P. 1995. Optimal Voting Rules. *Journal of Economic Perspectives*, 9(1): 51–64.

Young, H. P.; and Levenglick, A. 1978. A Consistent Extension of Condorcet's Election Principle. *SIAM Journal of Applied Mathematics*, 35(2): 285–300.

Zhang, H.; Cheng, Y.; and Conitzer, V. 2019. A better algorithm for societal tradeoffs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2229–2236.

Zhao, Z.; Villamil, T.; and Xia, L. 2018. Learning mixtures of random utility models. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

A Missing Proofs in Section 4

A.1 Proof of Lemma 4.2

Lemma 4.2. *For any $p > 0$, there is a reduction from ℓ_p QRJA to ℓ_p-regression that runs in $O(m)$.*

Proof of Lemma 4.2: Recall the reduction from an ℓ_p QRJA instance $(\mathbf{J}, \mathbf{w})$ to ℓ_p-regression $(\mathbf{A}, \mathbf{x})$:

$$A_{i,j} = \begin{cases} \sqrt[p]{w_i} & \text{if } j = a_i \\ -\sqrt[p]{w_i} & \text{if } j = b_i \\ 0 & \text{otherwise} \end{cases}, \quad z_i = \sqrt[p]{w_i} y_i.$$

Now suppose that $\mathbf{x}$ is an optimal solution to the constructed ℓ_p-regression. Then, $\mathbf{x}$ minimizes $\|\mathbf{A}\mathbf{x} - \mathbf{z}\|_p^p = \sum_{i=1}^m ((\mathbf{A}\mathbf{x})_i - z_i)^p = \sum_{i=1}^m w_i |x_{a_i} - x_{b_i} - y_i|^p$. Thus, any optimal solution $\mathbf{x}$ to the ℓ_p-regression instance $(\mathbf{A}, \mathbf{z})$ is an optimal solution to the original ℓ_p QRJA instance. ■

A.2 Proof of Theorem 4.7

Theorem 4.7. *ℓ_2 QRJA is equivalent to solving a Laplacian linear system. In particular, ℓ_2 QRJA can be solved in time $\widetilde{O}(m\sqrt{\log n})$ using the Lapalcian solver in (Cohen et al. 2014).*

Proof of Theorem 4.7: We first show that ℓ_2-regression corresponds to solving a linear system. Let $f(\mathbf{x})$ denote the objective value of ℓ_2-regression at x. The goal is to minimize

$$\tfrac{1}{2} f(\mathbf{x}) = \tfrac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{z}\|_2^2 = \tfrac{1}{2}\mathbf{x}^\top \mathbf{A}^\top \mathbf{A}\mathbf{x} - \mathbf{z}^\top \mathbf{A}\mathbf{x} + \tfrac{1}{2}\mathbf{z}^\top \mathbf{z}.$$

The derivative is $f'(\mathbf{x}) = \mathbf{A}^\top \mathbf{A}\mathbf{x} - \mathbf{A}^\top \mathbf{z}$. Therefore, $f(\mathbf{x})$ is minimized when $\mathbf{x} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{z}$.

Let $\mathbf{L} = \mathbf{A}^\top \mathbf{A}$. Recall the reduction from ℓ_2 QRJA to ℓ_2-regression, $\mathbf{A}$ has only two non-zero entries in each row and they sum up to 0, we can view $\mathbf{A}$ as an edge-vertex incidence matrix and thus $\mathbf{L}$ is a graph Laplacian. [3]

Finally, we invoke the main result of (Cohen et al. 2014): given an $n \times n$ Laplacian matrix $\mathbf{L}$ with m non-zero entries, one can compute an ε-approximate solution to a linear system in $\mathbf{L}$ in time

$$O\left(m\sqrt{\log n}\log(1/\varepsilon)\right).$$

This completes the proof of Theorem 4.7. ■

A.3 Proof of Theorem 4.11

Theorem 4.11. *For any $p < 1$, there exists a constant $c > 0$ such that it is NP-hard to approximate ℓ_p QRJA within a multiplicative factor of $\left(1 + \frac{c}{n^2}\right)$.*

Recall the reduction from Max-Cut to ℓ_p QRJA: Given an instance of Max-Cut with an undirected graph $G = (V, E)$, let $n = |V|, m = |E|$ and let $w_2 = \frac{2n}{1-p} + 1, w_1 = nw_2 + 1$. We construct an instance of ℓ_p QRJA with $n + 2$ candidates $V \cup \{v^{(s)}, v^{(t)}\}$ and $O(n + m)$ quantitative relative judgments. Specifically, we construct the followings:

- Add $(v^{(t)}, v^{(s)}, 1)$ with weight w_1.
- Add $(v^{(s)}, u, 0)$ with weight w_2 for each $u \in V$.
- Add $(v^{(t)}, u, 0)$ with weight w_2 for each $u \in V$.
- Add $(u, v, 1), (v, u, 1)$ with weight 1 for each $(u, v) \in E$.

To show validity of the reduction above, we will first establish integrality of any optimal solution.

Lemma A.1. *Any optimal solution of the ℓ_p QRJA instance described in the above reduction is integral. Moreover, all variables must be either 0 or 1 up to a global constant shift.*

We need an inequality for the proof of Lemma A.1.

Lemma A.2. *For any $d \in (0, \frac{1}{2}]$, $p \in (0, 1)$,*

$$1 - (1 - d)^p \le pd^p.$$

Proof of Lemma A.2: Fix $p \in (0, 1)$. Let $f(d) = pd^p - 1 + (1 - d)^p$. We have

$$f'(d) = p(pd^{p-1} - (1 - d)^{p-1}).$$

Note that f' is decreasing for $d \in (0, 1)$. In other words, f is single peaked on $(0, \frac{1}{2}]$ and continuous at 0. Now we only have to check that $f(0) \ge 0$, which is trivial, and $f\left(\frac{1}{2}\right) \ge 0$. For the latter, let

$$g(p) = (p + 1)0.5^p - 1.$$

$g(p) \ge 0$ for $p \in [0, 1]$ since $g(p)$ is concave on $[0, 1]$ and $g(0) = g(1) = 0$. The lemma then follows. ■

We then proceed to prove Lemma A.1.

Proof of Lemma A.1: Let x_a be the potential of candidate a in ℓ_p QRJA. W.l.o.g. assume that in any solution, $x_{v^{(s)}} = 0$. We first show that if $x_{v^{(t)}} \ne 1$, then moving it to 1 strictly improves the solution. Suppose $|x_{v^{(t)}} - 1| = d$. By moving $x_{v^{(t)}}$ to 1, we decrease the loss on the judgment $(v^{(t)}, v^{(s)}, 1)$ by $w_1 d^p$. For other judgments $(v^{(t)}, u)$ incident on $v^{(t)}$, the loss increase by no more than $w_2 d^p$, since

$$|(x_{v^{(t)}} \pm d) - x_u|^p \le |x_{v^{(t)}} - x_u|^p + d^p.$$

Overall, the cost decreases by at least

$$w_1 d^p - n w_2 d^p = d^p > 0.$$

Now we show moving any fractional x_u to the closest value in $\{0, 1\}$ strictly improves the solution. There are two cases:

- $x_u \in (0, 1)$. W.l.o.g. $x_u \in (1, \frac{1}{2}]$ and we try to move it to 0 by a displacement of $d = x_u$. The total loss on $(v^{(s)}, u, 0)$ and $(v^{(t)}, u, 0)$ decreases by $w_2(d^p + (1 - d)^p - 1)$, while the total cost on judgments of form $(u, v, 1)$ and $(v, u, 1)$ can increase by no more than $n(d^p + (2 + d)^p - 2^p)$. With Lemma A.2, we see that
$$\begin{aligned} w_2(d^p + (1 - d)^p - 1) &\ge w_2(d^p - pd^p) \\ &> 2nd^p \\ &\ge n(d^p + (2 + d)^p - 2^p). \end{aligned}$$
So, there is a positive improvement from rounding x_u.
- $x_u \notin [0, 1]$. W.l.o.g. $x_u < 0$ and we try to move it to 0 by a displacement of $d = -x_u$. The total loss on $(v^{(s)}, u, 0)$ and $(v^{(t)}, u, 0)$ decreases by $w_2(d^p + (1 + d)^p - 1)$, while the total cost on edges of form $(u, v, 1)$ and $(v, u, 1)$ can increase by no more than $n(d^p + (2 + d)^p - 2^p)$. And
$$\begin{aligned} w_2(d^p + (1 + d)^p - 1) &\ge w_2 d^p \\ &> 2nd^p \\ &\ge n(d^p + (2 + d)^p - 2^p). \end{aligned}$$

[3] A graph Laplacian matrix $\mathbf{L}$ satisfies $L_{i,j} < 0$ for any $i \ne j$, and $L_{i,i} = \sum_{j \ne i} |L_{i,j}|$. $\mathbf{L}$ is always positive semidefinite (PSD).

We conclude that in any optimal solution, $x_{v^{(s)}} = 0$, $x_{v^{(t)}} = 1$, and for any $u \in V$, $x_u \in \{0, 1\}$. ■

Next, we present a lemma that shows the connection between solutions in the Max-Cut instance and those in the constructed ℓ_p QRJA instance.

Lemma A.3. *A Max-Cut instance has a solution of size at least k iff its corresponding ℓ_p QRJA instance has a solution of loss at most $nw_2+2(m-k)+k2^p$. Moreover, with such a solution to the ℓ_p QRJA instance, one can construct a Max-Cut solution of the claimed size.*

Proof of Lemma A.3: Given a Max-Cut solution (S, T) of size at least k, setting the potentials of the vertices in S and T to be 0 and 1 respectively gives an ℓ_p QRJA solution with loss at most $nw_2 + 2(m - k) + k2^p$.

Given a ℓ_p QRJA solution of loss at most $nw_1 + 2(m - k) + k2^p$, we first round the solution to the form stated in Lemma A.1. This improves the solution. The two vertex sets $U = \{u \in V \mid x(u) = 0\}$ and $V = \{v \in V \mid x(v) = 1\}$ then form a Max-Cut solution of size at least k. ■

We are now ready to prove Theorem 4.11.

Proof of Theorem 4.11: According to Lemma A.3, any approximation with an additive error less than $2 - 2^p$ of the constructed ℓ_p QRJA instance can be rounded to produce an optimal solution to Max-Cut. Since Max-Cut is NP-Hard and the constructed ℓ_p QRJA instance's optimal solution has loss $\Theta(n^2 + m)$, the theorem follows. ■

A.4 Proof of Theorem 4.9

Theorem 4.9. *Fix $p \in [1, 2]$ and error ε, there exists a constant C such that for any ℓ_p QRJA instance $(n, m, \mathbf{J}, \mathbf{w})$, if we run Algorithm 1 with $M = Cn \log n \log \log^2 n$ and $\mathbf{s}$ being the Lewis weights for $(n, m, \mathbf{J}, \mathbf{w})$, then the output instance has an optimal solution that is at most $(1 + \varepsilon)$-approximate in the original instance.*

Proof of Theorem 4.9: Recall in the definition of the Lewis weights, we define matrix $\mathbf{A} \in \mathbb{R}^{m\times(n+1)}$ as the following

$$A_{i,j} = \begin{cases} \sqrt[p]{w_i} & \text{if } j = a_i \\ -\sqrt[p]{w_i} & \text{if } j = b_i \\ -\sqrt[p]{w_i}y_i & \text{if } j = n + 1 \\ 0 & \text{otherwise.} \end{cases}$$

Then for $\mathbf{x} \in \mathbb{R}^n$, we have

$$\left\| \mathbf{A} \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \right\|_p^p = \sum_{i=1}^{m} w_i |x_{a_i} - x_{b_i} - y_i|^p.$$

Thus the ℓ_p QRJA loss is always equal to $\|\mathbf{Ax}\|_p^p$ for some $\mathbf{x} \in \mathbb{R}^{n+1}$. The theorem then follows from the ℓ_p Matrix Concentration Bounds in (Cohen and Peng 2015). ■

B Additional Experiments

B.1 Performance Experiments on More Datasets

We include in this subsection the performance experiments on three more datasets. The new datasets are listed below.

- **Cross-Tables.** This dataset contains the results of cross-tables (a crossword-style word game) tournaments (https://www.cross-tables.com/) from 2000 to 2023. Each contest is a round-robin tournament involving around 8 contestants. A contestant's numerical score is his/her number of wins in the tournament. There are 1215 contests and 1912 contestants in total in this dataset.
- **F1-Full.** This dataset is an alternative version of F1. In F1-Full, we choose to additionally include contestants who do not complete the whole race. Now the contestants are ranked first by the number of laps they finish, and then their finishing time. A contestant's numerical score is the negative of the contestant's finishing time (in seconds). If the contestant does not finish all laps, we add a large penalty (1000 seconds) for each lap the contestant fails to finish. There are 878 contests and 606 contestants.
- **Codeforces-Core.** This dataset is a modified version of Codeforces. We only keep contestants who have participated in at least half of the contests in this dataset. We test on this modified dataset because all other datasets we use in the experiments are sparse datasets (i.e., contestants participate in a small fraction of the contests on average), so we want to see what happens on dense ones. There are 327 contests and 17 contestants in total.

We evaluate ℓ_1 and ℓ_2 QRJA using the same metrics against the same set of benchmarks as in Section 5 on these three datasets. The results are shown in Fig. 6. We highlight a few extra observations below.

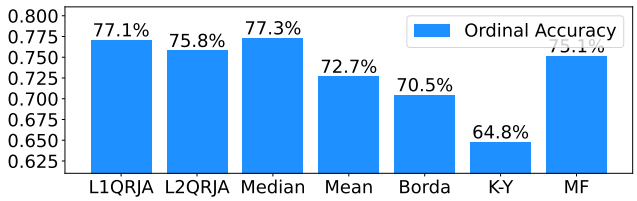
Extra observations on Cross-Tables. In terms of ordinal accuracy, Median performs the best among the tested algorithms on Cross-Tables. However, in terms of quantitative loss, Median is the worst algorithm among the tested ones. Moreover, it mostly performs suboptimally on other datasets as shown in Figs. 4 and 6. This shows that although Median is occasionally good in performance, it fails in other cases.

Extra observations on F1-Full. On F1-Full, both MF and ℓ_2 QRJA and perform considerably worse than ℓ_1 QRJA. This is not seen in other datasets. We believe this is because our score calculation results in a large variance in contestants' scores on F1-Full, which makes it harder for these methods to make good predictions. This also shows that ℓ_1 QRJA is more robust to datasets with large variances in contestants' performance than these methods. We also notice that Borda and Kemeny-Young perform well on F1-Full, which is consistent with their good performance on F1.

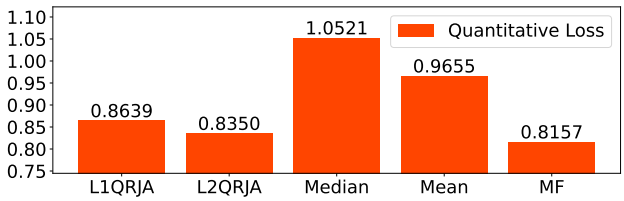
Extra observations on Codeforces-Core. In terms of ordinal accuracy, all tested algorithms except Borda perform well. In terms of quantitative loss, MF and Median are worse than the other ones. This shows that on a dense dataset like Codeforces-Core, most algorithms can make good predictions. Moreover, MF does not have a clear advantage over other algorithms in our problem even if the dataset is dense.

B.2 Subsampling Experiments on More Datasets

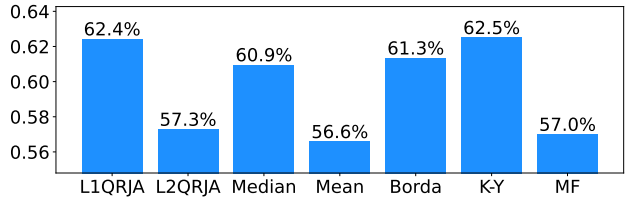
We also conduct the subsampling experiments in Section 5.3 on all other 5 datasets. The results are shown in Fig. 7.



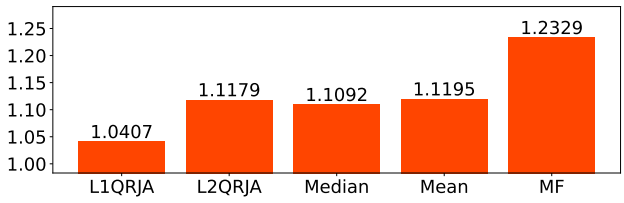
(a) Ordinal accuracy on Cross-Tables



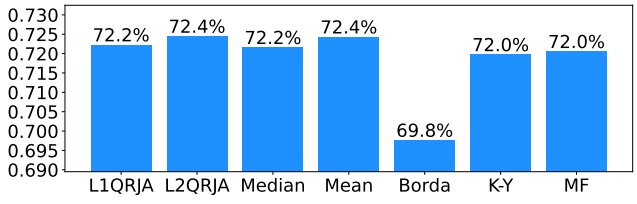
(b) Quantitative loss on Cross-Tables



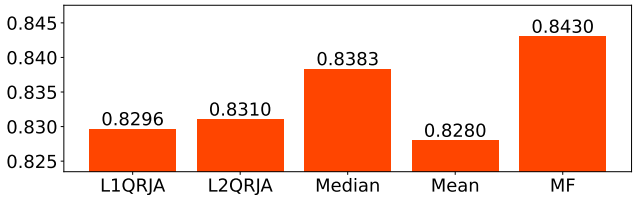
(c) Ordinal accuracy on F1 Full



(d) Quantitative loss on F1 Full



(e) Ordinal accuracy on Codeforces-Core



(f) Quantitative loss on Codeforces-Core

Figure 6: The performance of the algorithms on Cross-Tables, F1-Full, and Codeforces-Core. Error bars are not shown as the algorithms are deterministic. The results show that ℓ_1 QRJA still performs consistently well across the tested datasets. However, ℓ_2 QRJA performs considerably worse than ℓ_1 QRJA on F1-Full. This is not seen in other datasets.

Experiment results. The message here is the same as that in Section 5.3. In particular, Algorithm 1 can reduce the number of judgments with only a minor loss in performance as long as the subsample rate α is not too small. Note that in some of the figures, like Fig. 7c, the errors seem to be large visually. This is because of the small scale of the y-axis (only 0.6% for Fig. 7c). The actual errors are small. Moreover, we observe that the performance of ℓ_2 QRJA is slightly more robust to subsampling than that of ℓ_1 QRJA. This is consistent with the results in Section 5.3.

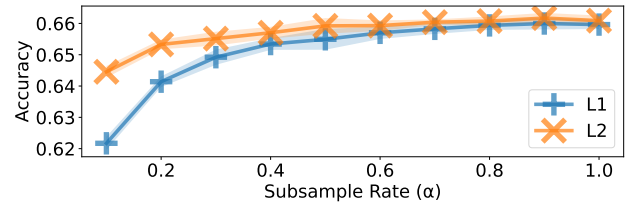
B.3 Experiments about Matrix Factorization

Recall that in Section 5, we only show results of one version of Matrix Factorization (MF). We include in this subsection the experiments involving different variants of Matrix Factorization as well as their implementation details.

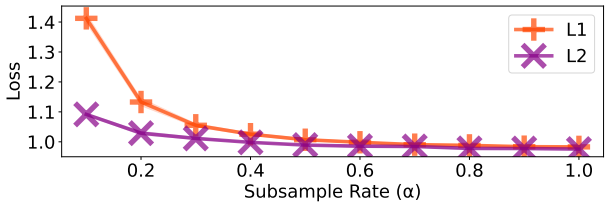
Implementation details. We have implemented two variants of MF: Low-Rank MF and Additive MF. The MF algorithm used in Section 5 is Low-Rank MF with rank $r = 1$. We describe the implementation details below.

- **Low-Rank MF.** Recall that in the context of our experiments, we can view each contestant as a row and each contest as a column. The score of a contestant in a contest is the entry in the corresponding row and column. A classical model of MF (Koren, Bell, and Volinsky 2009) is factorizing $\mathbf{A} \in \mathbb{R}^{n\times m}$ as the product of two low-rank matrices $\mathbf{U}\mathbf{V}^\top$, where $\mathbf{U} \in \mathbb{R}^{n\times r}$, $\mathbf{V} \in \mathbb{R}^{m\times r}$ for some small r. Note that in our experiments, the algorithm is required to predict a new column of $\mathbf{A}$ with no known entries. Therefore, we cannot directly apply this method since the corresponding row of $\mathbf{V}$ will remain unchanged after initialization. To solve this problem, we instead predict every column with known entries in $\mathbf{A}$ and then take the average of the predictions as the prediction for the new column. We use the standard loss function that sums up the squared errors of all observed entries. We implement this method with SciPy (Jones, Oliphant, and Peterson 2014) and use gradient descent for a fixed number of epochs on a deterministic initialization to keep the results deterministic. We test $r = 1, 2, 5$ in this subsection.
- **Additive MF.** We also consider an additive variant of MF. For $\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m$, this method predicts $A_{i,j} = x_i + y_j$. Here, x_i can be viewed as contestant i's skill level, and y_j can be interpreted as the (inversed) difficulty of contest j. We then use the vector $\mathbf{x}$ to make predictions. Note that this version of MF resembles QRJA in that for each of these two methods, the loss function is 0 if $A_{i,j} = x_i + y_j$ holds for the known entries. We also use the standard sum of the squared loss function and use gradient descent for a fixed number of epochs on a deterministic initialization to keep it deterministic.

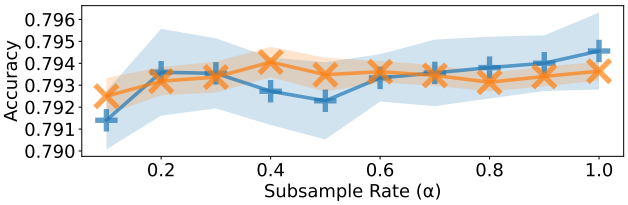
Performance experiments. We first evaluate these variants of MF using the same metrics as in Section 5 on all datasets. The results are shown in Fig. 8. We can see that R1 MF and Additive MF generally have similar performance. In contrast, R2 and R5 MF perform worse than the former. We



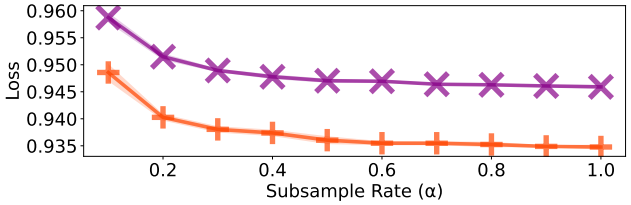
(a) QRJA's ordinal accuracy on F1



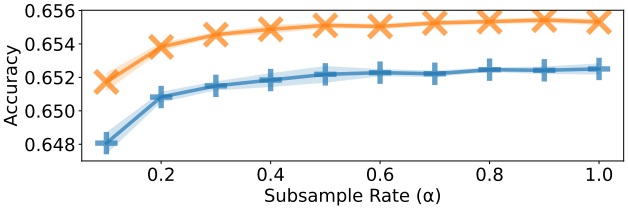
(b) QRJA's quantitative loss on F1



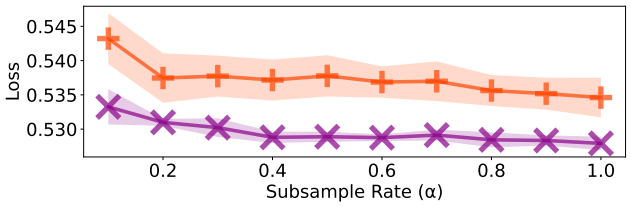
(c) QRJA's ordinal accuracy on Marathon



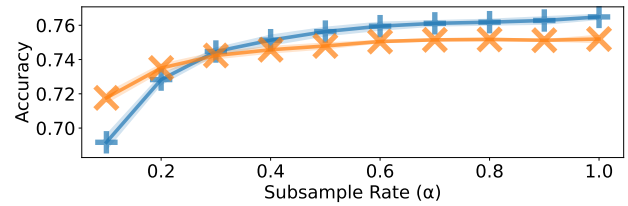
(d) QRJA's quantitative loss on Marathon



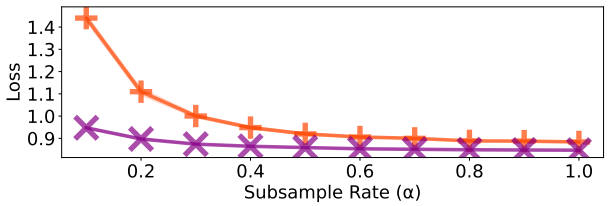
(e) QRJA's ordinal accuracy on Codeforces



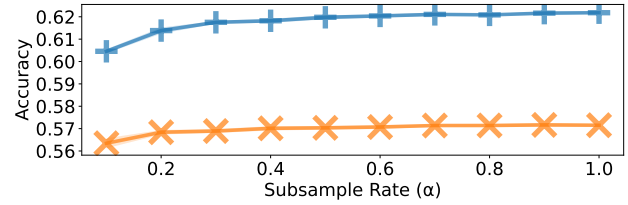
(f) QRJA's quantitative loss on Codeforces



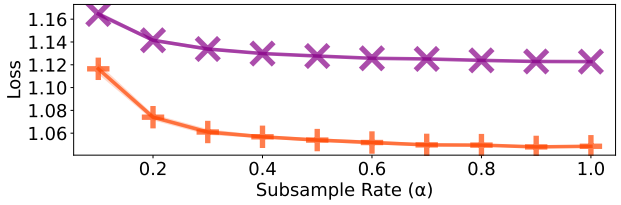
(g) QRJA's ordinal accuracy on Cross-Tables



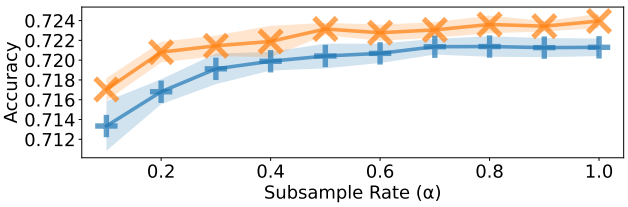
(h) QRJA's quantitative loss on Cross-Tables



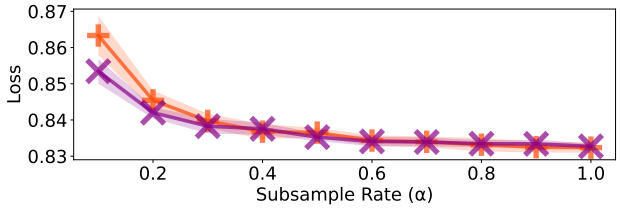
(i) QRJA's ordinal accuracy on F1-Full



(j) QRJA's quantitative loss on F1-Full

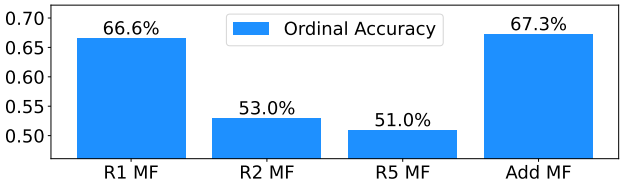


(k) QRJA's ordinal accuracy on Codeforces-Core

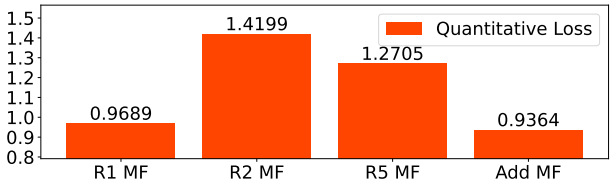


(l) QRJA's quantitative loss on Codeforces-Core

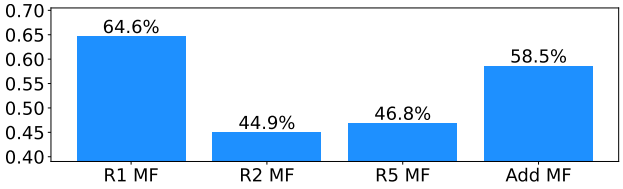
Figure 7: The performance of ℓ_1 and ℓ_2 QRJA after subsampling judgments using Algorithm 1 with equal weights for all judgments. The subsample rate α means $M = \lfloor \alpha m \rfloor$ in Algorithm 1. Error bars indicate the standard deviation. The results show that Algorithm 1 can reduce the number of judgments to a factor less than 1.0 with a minor loss in performance in the used datasets. Note that errors in some figures appear large because of the small scale of the y-axis. The actual errors are small.



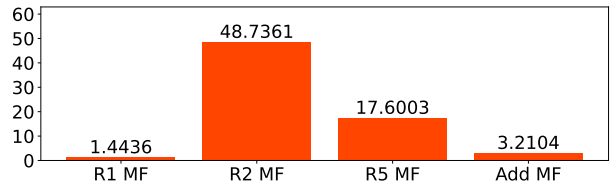
(a) MF's ordinal accuracy on Chess



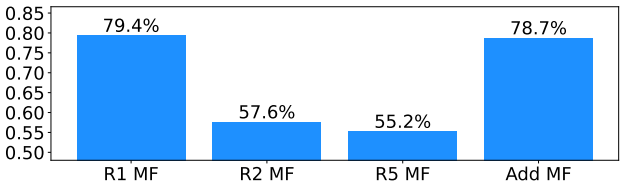
(b) MF's quantitative loss on Chess



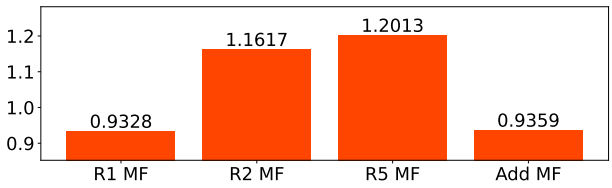
(c) MF's ordinal accuracy on F1



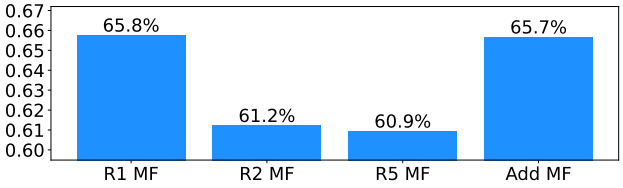
(d) MF's quantitative loss on F1



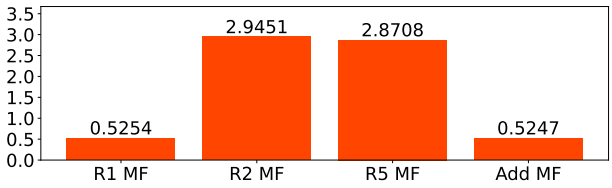
(e) MF's ordinal accuracy on Marathon



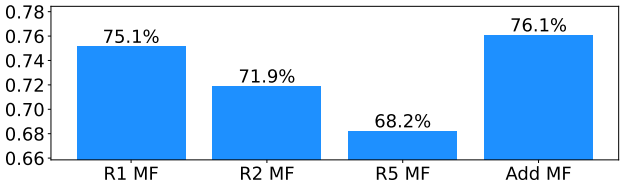
(f) MF's quantitative loss on Marathon



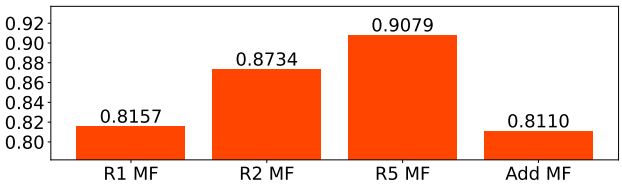
(g) MF's ordinal accuracy on Codeforces



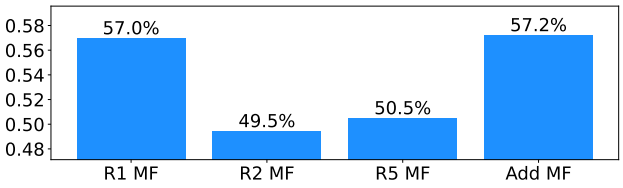
(h) MF's quantitative loss on Codeforces



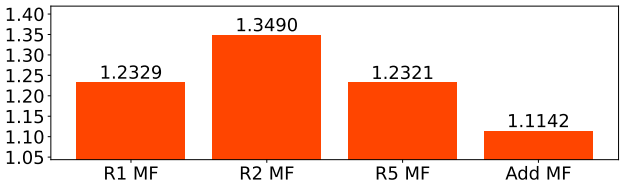
(i) MF's ordinal accuracy on Cross-Tables



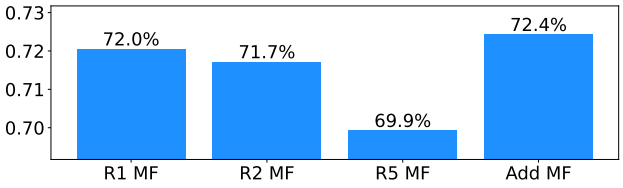
(j) MF's quantitative loss on Cross-Tables



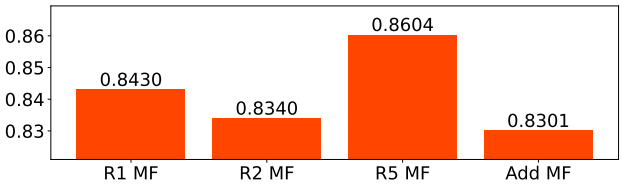
(k) MF's ordinal accuracy on F1-Full



(l) MF's quantitative loss on F1-Full

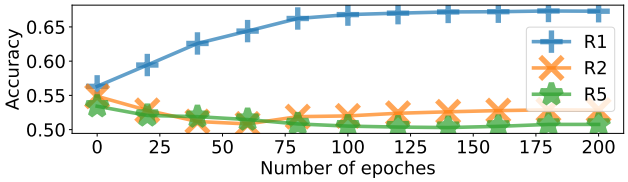


(m) MF's ordinal accuracy on Codeforces-Core

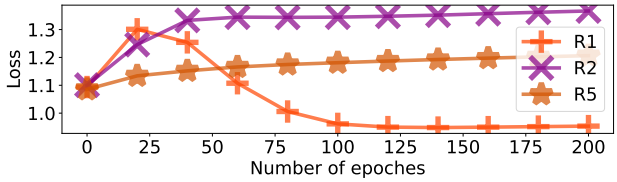


(n) MF's quantitative loss on Codeforces-Core

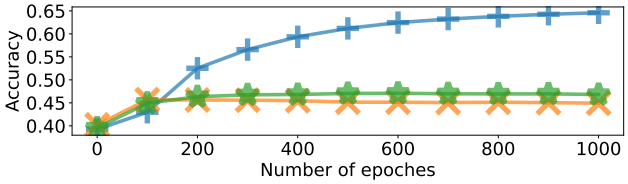
Figure 8: The performance of different variants of Matrix Factorization. The results show that R1 MF and Additive MF generally have similar performance. In contrast, R2 and R5 MF perform worse than the former.



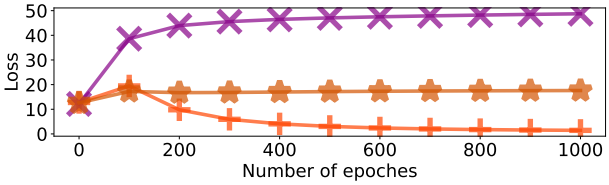
(a) MF's ordinal accuracy on Chess



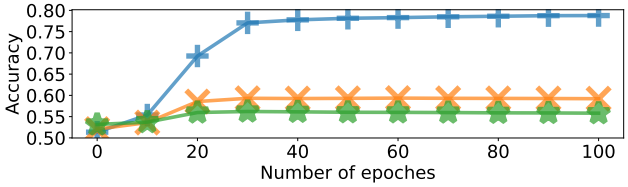
(b) MF's quantitative loss on Chess



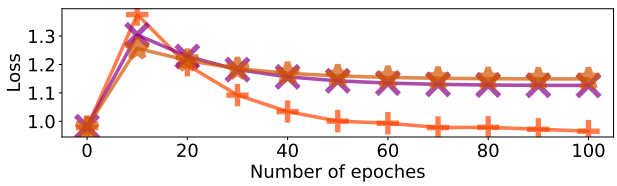
(c) MF's ordinal accuracy on F1



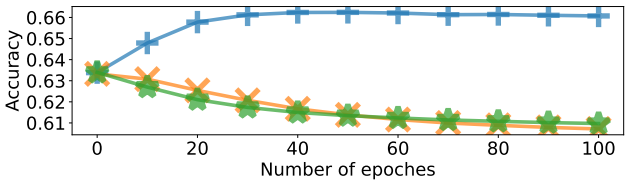
(d) MF's quantitative loss on F1



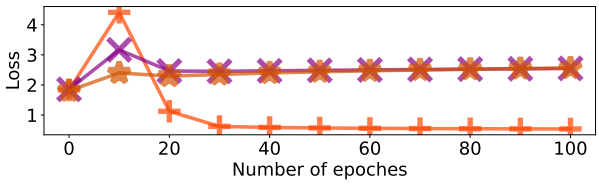
(e) MF's ordinal accuracy on Marathon



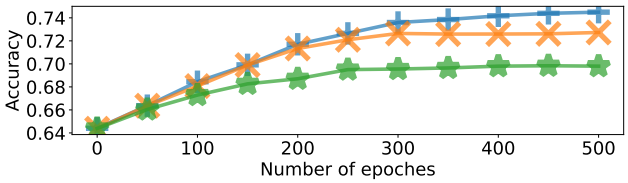
(f) MF's quantitative loss on Marathon



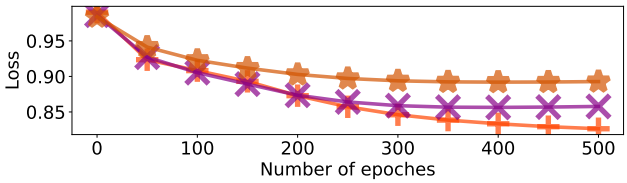
(g) MF's ordinal accuracy on Codeforces



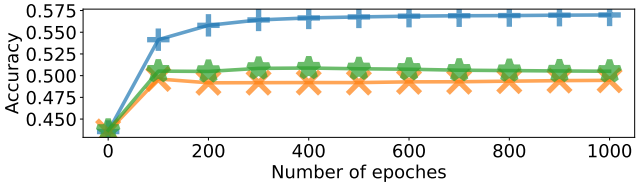
(h) MF's quantitative loss on Codeforces



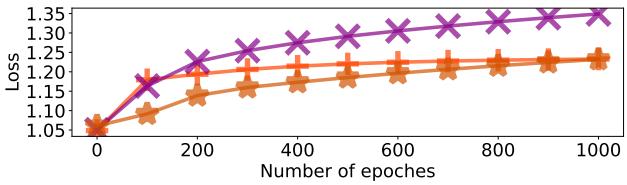
(i) MF's ordinal accuracy on Cross-Tables



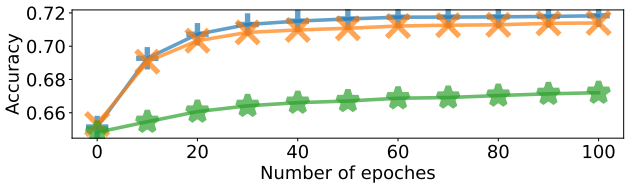
(j) MF's quantitative loss on Cross-Tables



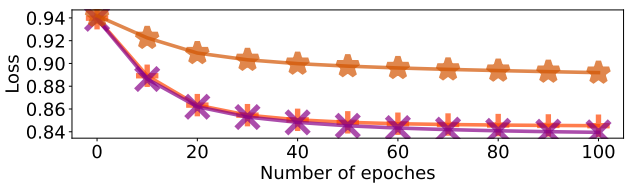
(k) MF's ordinal accuracy on F1-Full



(l) MF's quantitative loss on F1-Full

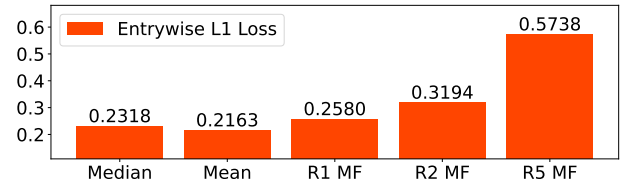


(m) MF's ordinal accuracy on Codeforces-Core

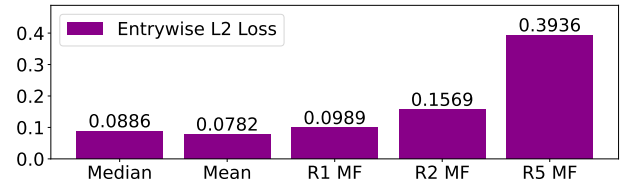


(n) MF's quantitative loss on Codeforces-Core

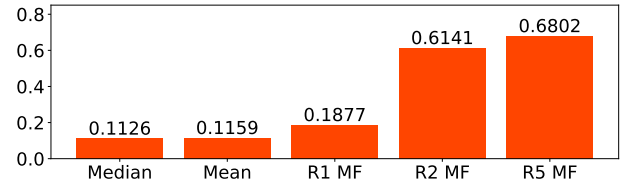
Figure 9: The performance of Matrix Factorization with different numbers of training epochs on all datasets. The results generally show that R1 MF outperforms R2 and R5 MF. Moreover, on some datasets, R2 and R5 MF's performance worsens as the number of training epochs increases. In contrast, R1 MF's performance improves as the number of training epochs increases.



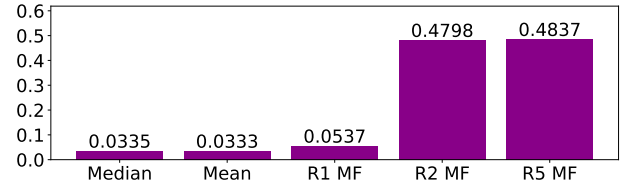
(a) Entrywise L1 loss on Chess



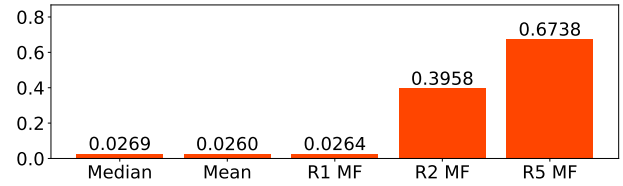
(b) Entrywise L2 loss on Chess



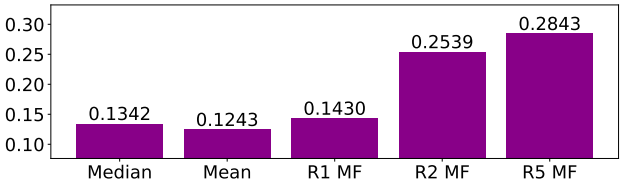
(c) Entrywise L1 loss on F1



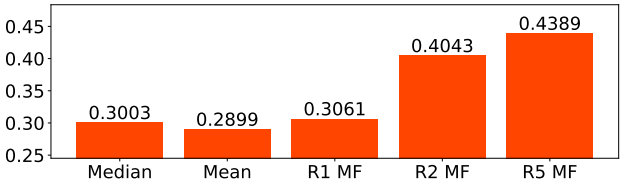
(d) Entrywise L2 loss on F1



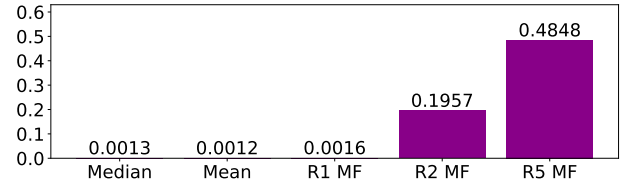
(e) Entrywise L1 loss on Marathon



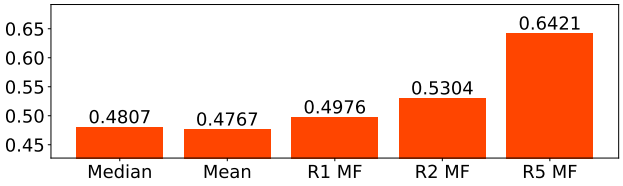
(f) Entrywise L2 loss on Marathon



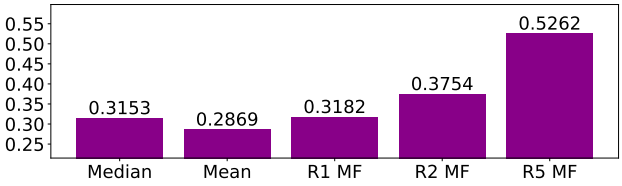
(g) Entrywise L1 loss on Codeforces



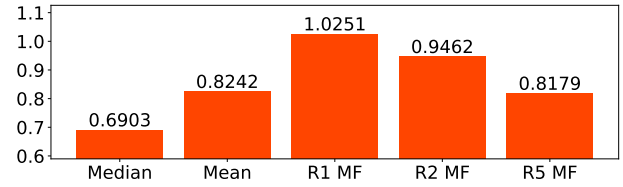
(h) Entrywise L2 loss on Codeforces



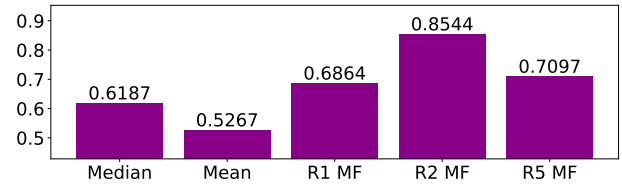
(i) Entrywise L1 loss on Cross-Tables



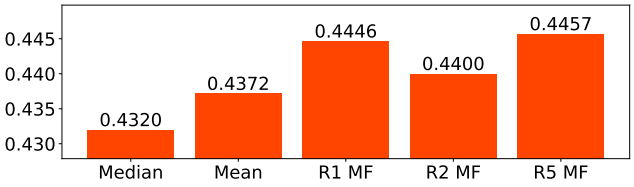
(j) Entrywise L2 loss on Cross-Tables



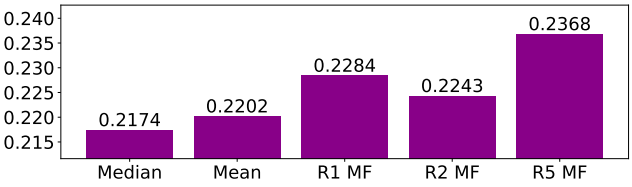
(k) Entrywise L1 loss on F1-Full



(l) Entrywise L2 loss on F1-Full



(m) Entrywise L1 loss on Codeforces-Core



(n) Entrywise L2 loss on Codeforces-Core

Figure 10: Entrywise L1 and L2 loss of Matrix Factorization, Mean, and Median. The results show that on most datasets, R1 MF outperforms R2 and R5 MF. The exceptions are F1-Full and Codeforces-Core. Moreover, Matrix Factorization does not have a clear advantage over Mean and Median on any dataset in terms of entrywise metrics.

therefore choose to present R1 MF in Section 5.

Low-Rank MF's performance over training. The observation that R2 and R5 MF perform worse than R1 MF is surprising to us. To confirm this observation, we plot the performance of these variants of MF with different numbers of training epochs on all datasets. The results are shown in Fig. 9. We can see that R1 MF generally outperforms R2 and R5 MF in terms of both ordinal accuracy and quantitative loss when trained for long enough. Moreover, R1 MF's performance on both metrics generally improves as the number of training epochs increases (the only exception is quantitative loss on F1-Full). In contrast, R2 and R5 MF's performance in terms of both metrics worsens as the number of training epochs increases on Chess, F1, and Codeforces. These observed phenomena suggest that R2 and R5 MF tend to overfit the data. The problem for R1 MF is less severe.

Experiment results on entrywise metrics. As the metrics in Section 5 are defined in a pairwise fashion and might not be well-suited for MF, we also evaluate the performance of MF in terms of entrywise L1 and L2 loss (i.e., the average absolute and squared error of the predictions on each contestant's actual score in each contest). We also normalize each of these losses by the corresponding loss of the trivial all-zero prediction. The results are shown in Fig. 10. Note that QRJA and Additive MF are not included, because their predictions can be shifted by an arbitrary constant, and thus entrywise losses do not apply to them. We can see that in terms of entrywise L1 and L2 loss, R1 MF outperforms R2 and R5 MF on most datasets. The exceptions are F1-Full and Codeforces-Core. These two datasets are different from the other ones in that F1-Full's scores are calculated with two numbers (the number of laps finished and the finishing time) and Codeforces-Core is a dense dataset constructed from Codeforces. Therefore, on these datasets, MF with higher ranks might be more suitable than R1 MF, while on the other datasets, they tend to overfit the training data. Moreover, we note that on entrywise metrics, MF generally performs worse than Mean and Median.

Summary of experiment results. In summary, experiments in this subsection show that on our datasets, R1 MF and Additive MF, which are similar in performance, generally perform better than R2 and R5 MF. Therefore, we choose to include only the results of R1 MF in Section 5.

C Axiomatic Characterization of ℓ_p QRJA

We characterize ℓ_p QRJA by giving a set of axioms for the family of transformation functions f of pairwise loss that we consider. We show that those transformation functions considered in ℓ_p QRJA are essentially the minimum set of functions satisfying these axioms.

Recall that for each judgment about a and b where a is better b by y units, the absolute error of the prediction vector $\mathbf{x}$ on this pair is $|x_a - x_b - y|$. Using this as the loss function, we obtain the ℓ_1 QRJA rule, which has been characterized using axioms in the context of social choice theory (Conitzer et al. 2016). Below we extend this characterization to ℓ_p QRJA for any positive rational number $p \in \mathbb{Q}_+$. Note that restricting p to be rational is without loss of generality, since the output of ℓ_p QRJA is continuous in p.

We consider transforming the absolute error by a transformation function f to obtain the actual pairwise loss, which is $f(|x_a - x_b - y|)$. For ℓ_p QRJA, the transformation function is $f(t) = t^p$. To characterize QRJA as a family of rules (for different $p \in \mathbb{Q}_+$), we give axioms for the corresponding family of transformation functions, i.e., t^p for $p \in \mathbb{Q}_+$. Let $\mathcal{F}$ be a family of transformation functions.

Below are the axioms we consider:

- *Identity.* There is an identity transformation $f_0 \in \mathcal{F}$, such that $f_0(t) = t$ for any $t \geq 0$.
- *Invertibility.* For each $f_1 \in \mathcal{F}$, there is an $f_2 \in \mathcal{F}$ such that f_1 composed with f_2 is identity, i.e., for any $t \geq 0$,
$$f_1(f_2(t)) = t.$$
- *Closedness under multiplication.* For any $f_1, f_2 \in \mathcal{F}$, there exists $f_3 \in \mathcal{F}$ such that for any $t \geq 0$,
$$f_1(t) \cdot f_2(t) = f_3(t).$$

We show below that the family of transformation functions corresponding to the ℓ_p QRJA rules is the minimum family of functions $\mathcal{F}^*$ satisfying the above axioms. By the first axiom, the identity transformation f_0 where $f_0(t) = t$ is in $\mathcal{F}^*$. (This corresponds to ℓ_1 QRJA.) Then by the third axiom, for any $k \in \mathbb{Z}_+$, f_0^k is also in $\mathcal{F}^*$, where $f_0^k(t) = t^k$. And by the second axiom, for any $k \in \mathbb{Z}_+$, $f_0^{1/k}$ is also in $\mathcal{F}^*$, where $f_0^{1/k}(t) = t^{1/k}$. This is because $f_0^{1/k}(f_0^k(t)) = t$. Finally, for any $r \in \mathbb{Q}_+$ where $r = p/q$ for $p, q \in \mathbb{Z}_+$, by the third axiom, $f_0^r = (f_0^{1/q})^p$ is in $\mathcal{F}^*$, where $f_0^r(t) = t^r$.

Note that the above argument establishes that $\mathcal{F}^*$ contains all transformation functions corresponding to QRJA, i.e.,

$$\{t^r \mid r \in \mathbb{Q}_+\} \subseteq \mathcal{F}^*.$$

Below we show the other direction, i.e., $\{t^r \mid r \in \mathbb{Q}_+\}$ satisfy the 3 axioms, and as a result,

$$\mathcal{F}^* \subseteq \{t^r \mid r \in \mathbb{Q}_+\}.$$

For $f_1(t) = t^{r_1}, f_2(t) = t^{r_2}$ where $r_1, r_2 \in \mathbb{Q}_+$, we have

$$f_1(t) \cdot f_2(t) = t^{r_1 + r_2},$$

where $r_1 + r_2 \in \mathbb{Q}_+$, and

$$f_1(f_2(t)) = (t^{r_2})^{r_1} = t^{r_1 \cdot r_2},$$

where $r_1 \cdot r_2 \in \mathbb{Q}_+$. This implies $\mathcal{F}^* \subseteq \{t^r \mid r \in \mathbb{Q}_+\}$. Thus $\mathcal{F}^* = \{t^r \mid r \in \mathbb{Q}_+\}$ as desired.