Quantitative Judgment Aggregation for Evaluating Contestants

Hanrui Zhang
Duke University
hrzhang@cs.duke.edu

Yu Cheng
University of Illinois at Chicago
yucheng2@uic.edu

Vincent Conitzer
Duke University
conitzer@cs.duke.edu

Abstract

Quantitative judgment aggregation is a new research topic in (computational) social choice. In this model, agents judge the relative quality of different entities, and from this we aim to obtain an aggregate assessment of their relative qualities. As is the case for the more traditional voting-by-ranking model in social choice, there are corresponding statistical problems. For example, a race provides a "judgment" of the relative abilities of the contestants, and we may wish to aggregate the results of multiple races. In this context, we no longer need to worry about, e.g., strategic misreporting by the judges, which enlarges the space of possibilities for aggregation rules. We investigate this larger space theoretically and evaluate on data from various real races.

1 Introduction

In the theory of *voting*, each voter *ranks* a set of alternatives, and a *voting rule* maps the vector of rankings to either a winning alternative or an aggregate ranking of all the alternatives. 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., [26]). Here, too, one may 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 point of intersection for these two communities is a model in which there is a latent "true" ranking of the alternatives, of which all the votes are just noisy observations. Given such a model, it is natural to try to estimate the correct ranking given the votes—and a procedure for doing so corresponds to a voting rule. (See, e.g., [43, 7, 28, 3, 35, 39] for early work, and [13] 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, which are not necessarily represented as rankings. For example, in *judgment aggregation* (for an overview, see [15]), judges assess whether certain propositions are true or false. If we simply take a majority vote of the judges for each proposition, the resulting values for the propositions may be logically inconsistent with each other. This raises the question of what are good judgment aggregation rules that do result in logically consistent outcomes. 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 a *quantitative* judgment aggregation problem [9, 10]. Specifically, the goal is to aggregate *relative quantitative judgments*, such as: "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, there are examples where similar relative "judgments" are 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. Hence, the "judgment" is that A is 1.5 times as fast as 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 here. 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 a race is not a strategic agent. As we will see, this will expand the set of rules that we may wish to consider.

Specifically, our goal is to obtain a *theoretically well motivated*, *transparent*, *interpretable*, and *auditable* aggregate measure of how relatively well contestants have performed. These other attributes, in addition to predicting future contests well, are essential for the responsible use of these techniques for, e.g., establishing a leaderboard.

1.1 Motivating Examples

In this section, we provide three motivating examples on which naïve mean/median performs poorly.

Example 1. When each race has some common "difficulty" (say, headwind, or 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 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 mean or median, Charlie's mean/median finishing time will be faster than Bob's. This is because, in this example, (it appears that) conditions were more favorable in the Chicago marathon.

Example 2. If our data shows that Alice has beaten before Bob 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). More generally, if Bob does not show up in the testing dataset, then the naïve median/mean effectively ignores all the data on Bob. However, some of the information can be useful for comparing other contestants.

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 results as in Figure 1, but with different 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 the race is much higher than the variance in the contestants' performances, taking the median will essentially focus on the result of a single race (the race with median difficulty) as illustrated in Figure 3.

Quantitative Judgment Aggregation (QJA) takes care of all the issues in previous examples because it considers *relative* performance instead of absolute performance.

1.2 Additional Related Work

Random utility models. Random utility models ([17, 46]) 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

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.

also 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 [32, 2] is a popular approach in collaborative filtering, where the goal is to recover missing entries given a partially-observed low-rank matrix. While the exact objective functions are different, our QJA approach is conceptually similar to rank-two matrix completion. We try to express the contestants' scores as the sum of two rank-one matrices: one matrix describing the strength of the candidates (same value across different races) and another one capturing the difficulty of the race (same offset across different contestants). 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 is interpreted as the "quality" of the contestant).

Preference Learning. Preference learning trains on a subset of items which have preferences toward labels and predicts the preferences for all items (see, e.g., [31]). 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 [9, 10]) 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 [14] and Microsoft's TrueSkill [20], have been developed to maintain rankings of players in various forms of games. Unlike QJA, 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.

2 Quantitative Judgment Aggregation with ℓ_p-Loss

We are now ready to formally define Quantitative Judgment Aggregation (QJA). We generalize earlier definitions [9, 10] by adding the exponent p; $p = 1$ corresponds to how it is defined previously.

Definition 1 (Quantitative Judgment Aggregation (QJA) with ℓ_p-Loss)**.** A pairwise contest result is a tuple (a, b, y), indicating that contestant a outperformed contestant b by y units in a contest. Fix $p > 0$. Given a set of contestants V with $|V| = n$, a set of pairwise contest results $E = \{a_i, b_i, y_i\}_{i=1}^m$, and weights $\{w_i\}_{i=1}^m$, the QJA problem with ℓ_p-loss asks for a vector $x \in \mathbb{R}^n$ that minimizes $\sum_{i=1}^m w_i |x_{a_i} - x_{b_i} - y_i|^p$.

The ℓ_p-loss QJA rule outputs an optimal solution $x \in \mathbb{R}^n$ to ℓ_p-loss QJA (with ties broken arbitrarily). Intuitively, x reflects the estimated skill levels of the contestants, and serves as a predictor for the outcome of a future contest. y is by how much a actually outperformed b in one contest in our data, and the quantity $x_a - x_b$ is by how much we expect contestant a to outperform contestant b in a future contest. Generally speaking, we will place the same weight $w_i = 1$ on all races.

QJA with ℓ_2-loss is in important ways similar to taking the average for each contestant.

Lemma 1. *When all contestants participate in all contests, and all weights $w_i = 1$, QJA with ℓ_2-loss is equivalent to taking the mean of each contestant's results.*

We defer the proof for the lemma, as well as all other proofs, to Appendix A in the full version of the paper. In contrast, QJA with ℓ_1-loss is closely related to taking the median. When there are only two contestants a and b, QJA with ℓ_1-loss consists in setting $x_a - x_b$ to the median of the pairwise differences in races. This is considered an essential property in true social choice applications [10]—for one, it removes incentives to strategically misreport—which is why earlier work did not consider other values of p; however, for our applications here, this property is not essential. Rather, the exponent p accounts for compatibility with different noise models. For example, ℓ_2 methods usually perform better with moderate noise, while ℓ_1 is much more robust to outliers. Intuitively, a smaller p works better for stronger noise. This intuition is justified in the following subsection, where we study the optimality of ℓ_p-loss QJA as an estimator. We also provide an axiomatic characterization of ℓ_p QJA in Appendix C.

2.1 MLE Interpretation of ℓ_p-Loss Rules

We next show that every ℓ_p-loss QJA rule can be interpreted as the maximum-likelihood estimator for the "true" relative abilities of the contestants, under a model in which contest results are noisy observations of these latent abilities.

Recall that the i-th pairwise race result (a_i, b_i, y_i) represents that player a_i performed y_i units better than player b_i. Assume there exists a latent ground-truth ability vector x^*, and the value of y_i is drawn independently for each i, from the distribution with $\Pr[y_i] \propto \exp(-|y_i - (x^*_{a_i} - x^*_{b_i})|^p)$. For example, when $p = 2$, we observe the difference between the abilities of any pair of contestants with i.i.d. Gaussian noise. We show that QJA with ℓ_p-loss is the MLE for this family of noise models.

Proposition 1. *The ℓ_p-loss QJA rule is the MLE rule for* $\Pr[y_i] \propto \exp(-|y_i - (x^*_{a_i} - x^*_{b_i})|^p)$.

3 Computational Aspects of Quantitative Judgment Aggregation

In this section, we study the computational complexity of ℓ_p-loss QJA for different values of p. We give a clean dichotomy result: we show that the problem is NP-hard when $0 < p < 1$, and can be solved in polynomial time when $p \geq 1$. We also present faster algorithms for the special cases $p \in \{1, 2\}$.

We first prove that the QJA problem is a special case of the ℓ_p-regression problem.

Definition 2 (ℓ_p-regression)**.** Given a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $z \in \mathbb{R}^n$, find a vector x that minimizes $\|Ax - z\|_p^p$.

Theorem 1. *Fix any $p > 0$. The QJA problem with ℓ_p-loss is a special case of ℓ_p-regression.*

3.1 Polynomial-Time Algorithms When $p \geq 1$

We first show that ℓ_p-loss QJA admits polynomial-time algorithms when $p \geq 1$. Because QJA with ℓ_p-loss is a special case of ℓ_p-regression, it is sufficient to show that ℓ_p-regression can be solved efficiently. ℓ_p-regression is a fundamental problem with many applications in statistical data analysis and machine learning. It has been studied extensively (see, e.g., [30, 40, 12, 11, 33, 4, 29, 5, 41]). In particular, we invoke the following recent result from [1].

Lemma 2 ([1])**.** *Fix $p \geq 1$. Given $A \in \mathbb{R}^{m\times n}$ and $z \in \mathbb{R}^n$, we can compute a vector x such that $\|Ax - z\|_p \leq \min_x \|Ax - z\|_p + \varepsilon$ in time*

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

where $\mathrm{nnz}(A)$ *is the number of non-zero entries in A, and ω is the matrix multiplication exponent.*[1]

Since the dependence on ε is $\log(1/\varepsilon)$ in Lemma 2, ℓ_p-regression can be solved up to machine precision in polynomial time. We immediately have the following corollary:

Corollary 1. *For any $p \geq 1$, QJA with ℓ_p-loss can be solved in polynomial time.*

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

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 $p = 2$ due to their conceptual simplicity and faster running time. In the next section, we present faster algorithms for QJA when $p \in \{1, 2\}$.

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

In this section, we show that ℓ_p-loss QJA admits faster algorithms when $p \in \{1, 2\}$. Recall that given $A \in \mathbb{R}^{m \times n}$ and y, the ℓ_p-loss QJA problem is to minimize $\|Ax - z\|_p^p$ over $x \in \mathbb{R}^n$. It is a special case of ℓ_p-regression where each row of A has exactly two non-zero entries ($A_{i,a_i} = w_i$ and $A_{i,b_i} = -w_i$).

We can formulate ℓ_1-regression as a linear program (LP). For ℓ_1 QJA, [45] gave a faster algorithm than using general LP solver.

Lemma 3 ([45])**.** *The ℓ_1 QJA problem can be reduced to Minimum Cost Flow. Using the flow algorithm in [18], ℓ_1 QJA can be solved in time $\widetilde{O}(m^{1.5} \log(nW))$, where $W = \max_i w_i$.*

The ℓ_2-regression problem is also commonly known as the (linear) Least-Square regression problem. We exploit the special structure of ℓ_2 QJA, and relate it to the problem of solving Laplacian linear systems [36, 25, 6]. Formally, we prove the following theorem.

Theorem 2. *QJA with ℓ_2-loss can be computed by solving a Laplacian linear system. In particular, using the Laplacian solver in [6], QJA with ℓ_2-loss can be solved in time $\widetilde{O}(m\sqrt{\log n})$.*

3.3 NP-hardness When $p < 1$

We complement our positive results by the following hardness result.

Theorem 3. *For any $0 < p < 1$, it is NP-hard to approximate the ℓ_p-loss QJA problem within a factor of $1 + \frac{c}{n^2}$ for some constant c.*

4 Experiments

In this section, we conduct experiments on real-world datasets. We make ordinal and quantitative predictions based on our aggregate evaluation of contestants (QJA). From the experiments we see that ℓ_1 QJA performs as well as or better than the best benchmark on all datasets, for both ordinal and quantitative predictions. All experiments are done on a laptop with 8GB of memory and a 2.6 GHz Intel Core i5 CPU.

To evaluate QJA, we consider settings where: (1) contests are reasonably frequent, (2) the contests provide not only a ranking of the participants but also a numerical score, (3) the outcomes vary from one contest to the next, and (4) not every contestant appears in every contest. Observing scores of several consecutive contests, we try to make ordinal/quantitative predictions of the chronologically next contest using QJA and alternative methodologies. We use the Min-Cost Circulation subroutine provided in the **LEMON** library[2] to implement ℓ_1 QJA (as in [45]). We use Sparse Least Squares Regression provided by **SciPy** [22] to implement ℓ_2 QJA.

Datasets.

- *Programming contests.* We use data from Codeforces (`codeforces.com`), a website hosting frequent online programming contests. In each contest, a score is given to each participant according to their performance.
- *NYC and Boston marathons.* Contests used for training and testing include all NYC and Boston marathons from 2013 to 2017 in chronological order (i.e. Boston 2013 followed by NYC 2013 followed by Boston 2014, etc.). We use data from `marathonguide.com`, which publishes results of all major marathon events. In each marathon, the score of a runner is her finishing time in seconds.
- *Formula One races.* We use all 20 Formula One races in 2017 (data from `www.formula1.com`). In each race, the score of a driver is the difference (in seconds) between her finishing time and the first finisher's.

[2]See `lemon.cs.elte.hu/trac/lemon`.

- *SAT solver competitions.* SAT Competition is a competitive event for solvers of the Boolean Satisfiability (SAT) problem (`satcompetition.org`). We focus on the main-track results of 2017. Each SAT instance is treated as a contest, and each solver as a contestant.

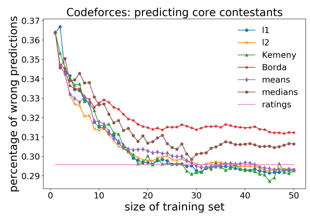
Evaluation. For Codeforces, marathon, and Formula One datasets, contests are naturally ordered by the times at which they happen. In such cases, we use several consecutive contests to predict the immediately succeeding one. For each point in a figure, we average the results over 5 runs on different test contests. Specifically, suppose there are $m + n$ contests. For every $k = 1, \dots, m$, we report the average accuracy over $n = 5$ runs. For the i-th run, we use the $(m + i)$-th contest as test set, and use the k immediately preceding contests as training data. In the SAT solver competition dataset, contests (i.e., SAT instances) exhibit no chronological order. We use all contests but one as the training set to predict the remaining one, and report the average.

Benchmarks. We evaluate ℓ_1 and ℓ_2 QJA against the following benchmarks. Means and medians are like QJA in that they use numerical information, but unlike QJA in that they use absolute instead of relative numbers; i.e., they do not correct for the difficulty of the individual contest. In one-dimensional environments like ours, means and medians are considered to be among the best imputation methods for various tasks (see, e.g., [16, 34]). Kemeny-Young and Borda are like QJA in that they are social-choice-theoretic methods that use relative rather than absolute performance, but unlike QJA in that they do not use numerical information, only the ranking of contestants. Among social-choice-theoretic methods, Kemeny-Young and Borda both have natural interpretations as estimating a "ground truth" ranking of the contestants [42, 43, 7, 38, 8]. We do not claim that these methods are superior to all other ranking-based methods (such as, for example, random utility models) for all purposes; but our measure of accuracy, explained below, strongly favors Kemeny-Young. Specifically, among ranking-based methods, Kemeny-Young optimizes exactly our measure of accuracy on the training set, giving it an advantage over every other ranking-based method. (Indeed, we will see that it significantly outperforms Borda.) See Appendix B for more details about the benchmarks and the datasets.

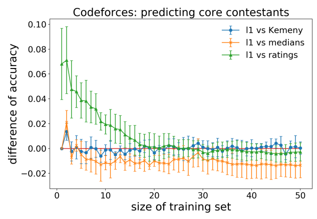
- *Means.* For every contestant in the training set, we take the mean of her scores in all training contests in which she participates.
- *Medians.* For every contestant in the training set, we take the median of her scores in all training contests in which she participates.
- *The Kemeny-Young rule [24, 44, 42].* This is a voting method that takes multiple (partial) rankings of the contestants as input. We obtain these rankings from the training contests. It outputs ordinal predictions only; specifically, it outputs a ranking that minimizes the number of *disagreements* on pairs of contestants with the input rankings.
- *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 participant in contest j receives $1 - 2(i - 1)/(n_j - 1)$ points, where n_j is the number of participants in the contest. The aggregated ranking result is obtained by sorting the contestants by their total points.
- *Ratings* (for the Codeforces dataset only). Codeforces maintains ratings for all users, using a variant of the Elo rating system [14]. The ratings are calculated based on all previous contests, and are intended to be predictive.
- *TrueSkill* (for the SAT solver dataset only). TrueSkill is a Bayesian skill rating system. The official python package (see `https://trueskill.org/`) does not work for contests consisting of more than 130 participants, so we only test it on the relatively small SAT solver dataset.

There are many other, related approaches that deserve mention in this context, but that are not ideal to compare to because they do not exactly fit our setting and/or motivation. There is work on predicting differences in final scores in football [27, 23, 19], but our setting is different, because a football match always has only two contestants. Similarly, the Bradley-Terry model [21] uses only pairwise comparisons. Thurstone's model [37], like the voting methods, only takes ranking information as input. QJA in some ways resembles matrix factorization, but the focus in that line of work tends to be on recovering the ground truth from a relatively small number of entries, rather than aggregating results in a simple, transparent manner.

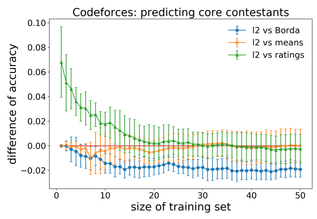
Ordinal predictions. We first focus on ordinal predictions, which allows us to compare to benchmarks that only output a ranking. We measure the accuracy by the percentage of incorrect pairwise predictions on core contestants in the test contest's ranking.



(a)

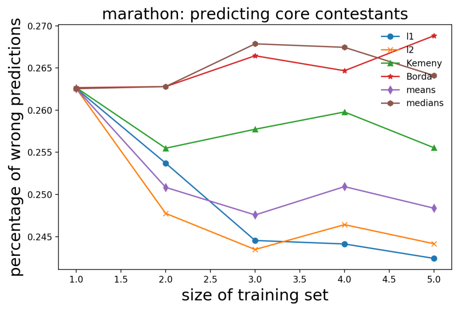


(b)

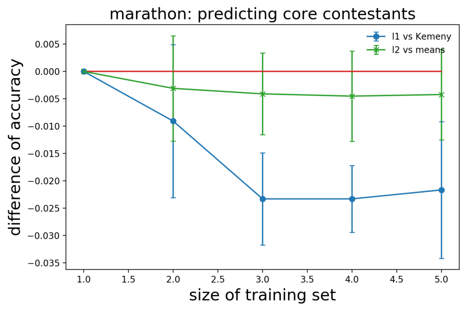


(c)

Figure 4: Comparison of accuracy on the Codeforces dataset.

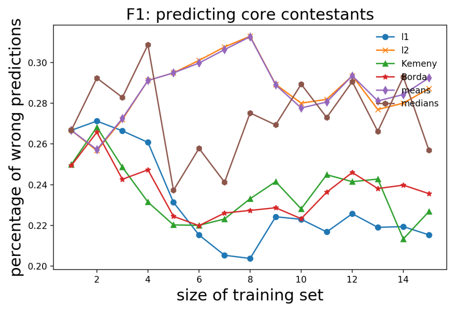


(a)

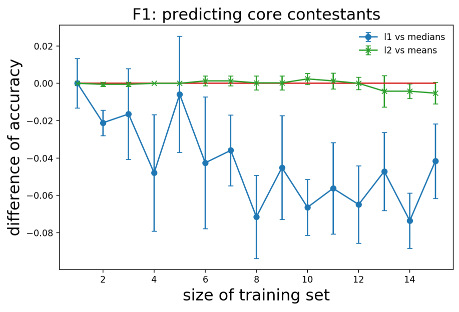


(b)

Figure 5: Comparison of accuracy on the marathon dataset.



(a)



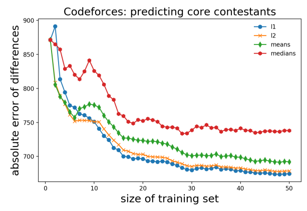
(b)

Figure 6: Comparison of accuracy on the F1 dataset.

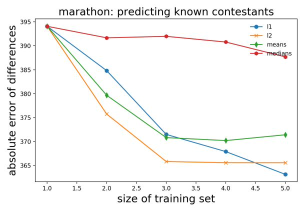
Figure 4 shows the accuracies of QJA and the benchmarks on the Codeforces dataset. The Figure 4a shows the error rates of all methods. Note that Codeforces ratings are based on the full history, and we cannot restrict it to a limited training set. Hence, its performance is a constant in the figure. In Figure 4b, we pairwise compare ℓ_1 QJA to the benchmarks most related to it, Kemeny-Young and medians, as well as to Codeforces ratings. Numbers below 0 indicate that ℓ_1 QJA performs better. In Figure 4c, we compare ℓ_2 QJA to the benchmarks most related to it, Borda and means, as well as to Codeforces ratings. The figure shows that training on 50 contests, both versions of QJA, Kemeny-Young, and means are competitive with Codeforces ratings. In addition, ℓ_1 QJA performs similarly to Kemeny-Young, and ℓ_2 QJA performs similarly to means.

Figure 5 shows the accuracies of QJA and the benchmarks on the marathon dataset. Figure 5a shows the error rates of all methods, and Figure 5b plots the differences between ℓ_1 QJA and Kemeny-Young, and between ℓ_2 QJA and means, respectively. Medians and Borda are not plotted in the right chart, since the gaps to QJA are already large enough in 5a. Figure 6 shows similar statistics for the Formula One dataset. We note that ℓ_2 QJA and means are quite close on the Formula One dataset, since, as discussed above, in that dataset, almost all contestants are present in all contests.

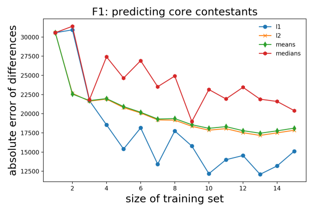
The first row of Table 1 shows the accuracies of QJA and the benchmarks on the SAT solver dataset. The size of the training set here is always the same, consisting of all instances but one. There are 46 qualifying instances in total, as discussed above. ℓ_2 QJA does not appear separately since it is equivalent to means on this dataset. Again, ℓ_1 QJA beats all the benchmarks, including the carefully designed TrueSkill rating system (which, as argued above, would handle online prediction tasks better).



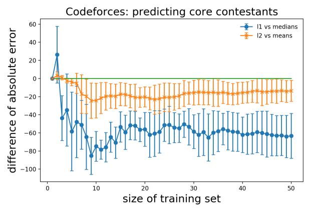
(a)



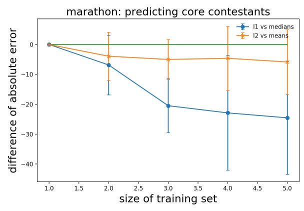
(b)



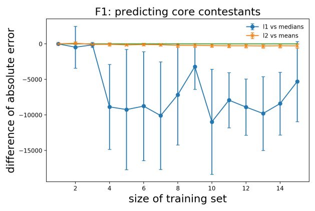
(c)



(d)



(e)



(f)

Figure 7: Comparison of absolute error of differences on the Codeforces, marathon, and F1 datasets.

method	ℓ_1 QJA	Kemeny-Young	Borda	TrueSkill	means	medians
average error rate	34.84%	35.98%	35.76%	35.35%	38.25%	39.75%
average absolute error	327.17	N/A	N/A	N/A	372.85	340.95

Table 1: SAT solver competition results.

Quantitative predictions. We now turn our attention to quantitative predictions—where we aim to predict not only whether A finishes ahead of B, but also by how much. For this, we measure accuracy by the average absolute error between predicted pairwise differences of scores and the actual differences. We compare QJA only to means and medians, since the other benchmarks are designed only for ordinal predictions. Figure 7 shows the errors of QJA and the quantitative benchmarks on the Codeforces, marathon, and Formula One datasets. Figure 7a shows the error rates of all methods on the Codeforces dataset. In 7d, we plot, on the Codeforces dataset, the differences of errors between (1) ℓ_1 QJA and medians, and (2) ℓ_2 QJA and means. QJA is significantly better than the benchmarks in terms of quantitative predictions. Figures 7b and 7e (resp. 7c and 7f) similarly compares QJA and the benchmarks on the marathon (resp. Formula One) dataset. ℓ_2 QJA and means are close on the Formula One dataset for exactly the same reason, as discussed above. The second row of Table 1 show the absolute errors of QJA and the benchmarks on the SAT solver dataset. ℓ_1 QJA beats the benchmarks on average.

Summary of Experimental Results. From the experiments we see that the benchmark methods often perform very differently across datasets and tasks. For example, Kemeny-Young performs quite well on the Codeforces and SAT solver datasets, but much worse on the marathon dataset. Means gives good pairwise predictions on the Codeforces dataset, but performs less well on the other two. Also, despite its remarkable performance in pairwise predictions as a simple method, means is consistently worse than both versions of QJA in terms of quantitative predictions. We suspect this instability of performance is caused in part by the difference in the underlying noise models of the respective datasets. On the other hand, the two versions of QJA are never significantly worse than their benchmark counterparts, and in particular ℓ_1 QJA performs as well as or better than the best benchmark on all datasets, for both ordinal and quantitative predictions.

Broader Impact

Our methods can be used to evaluate contestants in a more accurate, transparent, interpretable and auditable way. By taking into account more information in a theoretically principled way, our methods can help prevent potential bias induced by complicated blackbox ranking and preference learning methods. Of course, implemented in irresponsible ways, our methods could lead to privacy issues (as they produce predictions of the true, potentially private, "strength" of the contestants), and could be maliciously manipulated in certain scenarios.

References

[1] Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, and Yuanzhi Li. 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)*, pages 1130–1137. ACM, 2018.

[2] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[3] Ioannis Caragiannis, Ariel D Procaccia, and Nisarg Shah. When do noisy votes reveal the truth? In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 143–160. ACM, 2013.

[4] Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90. ACM, 2013.

[5] Michael B. Cohen and Richard Peng. ℓ_p row sampling by lewis weights. In *Proceedings of the 47th annual ACM Symposium on Theory of Computing (STOC)*, pages 183–192. ACM, 2015.

[6] Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $m \log^{1/2} n$ time. In *Proc. 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 343–352, 2014.

[7] Vincent Conitzer and Tuomas Sandholm. Common voting rules as maximum likelihood estimators. In *Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 145–152, Edinburgh, UK, 2005.

[8] Vincent Conitzer, Matthew Rognlie, and Lirong Xia. Preference functions that score rankings and maximum likelihood estimation. pages 109–115, 2009.

[9] Vincent Conitzer, Markus Brill, and Rupert Freeman. Crowdsourcing societal tradeoffs. In *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1213–1217, Istanbul, Turkey, 2015.

[10] Vincent Conitzer, Rupert Freeman, Markus Brill, and Yuqian Li. Rules for choosing societal tradeoffs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 460–467, Phoenix, AZ, USA, 2016.

[11] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W Mahoney. Sampling algorithms and coresets for ℓ_p regression. *SIAM Journal on Computing*, 38(5): 2060–2078, 2009.

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

[13] Edith Elkind and Arkadii Slinko. Rationalizations of voting rules. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 8. Cambridge University Press, 2015.

[14] Arpad E Elo. *The rating of chessplayers, past and present*. Arco Pub., 1978.

[15] Ulle Endriss. Judgment aggregation. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 17. Cambridge University Press, 2015.

[16] Jean Mundahl Engels and Paula Diehr. Imputation of missing longitudinal data: a comparison of methods. *Journal of clinical epidemiology*, 56(10):968–976, 2003.

[17] Mohsen Ahmadi Fahandar, Eyke Hüllermeier, and Inés Couso. Statistical inference for incomplete ranking data: the case of rank-dependent coarsening. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1078–1087. JMLR. org, 2017.

[18] Harold N. Gabow and Robert E. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18(5):1013–1036, 1989.

[19] Shengbo Guo, Scott Sanner, Thore Graepel, and Wray Buntine. Score-based bayesian skill learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 106–121. Springer, 2012.

[20] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill$^{\text{tm}}$: A bayesian skill rating system. In *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, pages 569–576, 2006.

[21] David R Hunter et al. Mm algorithms for generalized bradley-terry models. *The annals of statistics*, 32(1):384–406, 2004.

[22] Eric Jones, Travis Oliphant, and Pearu Peterson. {SciPy}: open source scientific tools for {Python}. 2014.

[23] Dimitris Karlis and Ioannis Ntzoufras. Bayesian modelling of football outcomes: using the skellam's distribution for the goal difference. *IMA Journal of Management Mathematics*, 20(2): 133–145, 2008.

[24] John Kemeny. Mathematics without numbers. *Daedalus*, 88:575–591, 1959.

[25] Ioannis Koutis, Gary L. Miller, and Richard Peng. A nearly-$m \log n$ time solver for SDD linear systems. In *Proc. 52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 590–598, 2011.

[26] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–231, 2009.

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

[28] Marina Meila, Kapil Phadnis, Arthur Patterson, and Jeff Bilmes. Consensus ranking under the exponential model. In *Proceedings of the 23rd Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 285–294, Vancouver, BC, Canada, 2007.

[29] Xiangrui Meng and Michael W. Mahoney. 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)*, pages 91–100. ACM, 2013.

[30] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.

[31] Tapio Pahikkala, Evgeni Tsivtsivadze, Antti Airola, Jouni Järvinen, and Jorma Boberg. An efficient algorithm for learning to rank from preference graphs. *Machine Learning*, 75(1): 129–165, 2009.

[32] Jason D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 713–719, 2005.

[33] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for ℓ_1-regularized loss minimization. *Journal of Machine Learning Research*, 12(Jun):1865–1892, 2011.

[34] Fiona M Shrive, Heather Stuart, Hude Quan, and William A Ghali. Dealing with missing data in a multi-question depression scale: a comparison of imputation methods. *BMC medical research methodology*, 6(1):57, 2006.

[35] Hossein Azari Soufiani, David C Parkes, and Lirong Xia. A statistical decision-theoretic framework for social choice. In *Advances in Neural Information Processing Systems*, pages 3185–3193, 2014.

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

[37] Louis L Thurstone. A law of comparative judgment. *Psychological review*, 34(4):273, 1927.

[38] Michel Truchon. Borda and the maximum likelihood approach to vote aggregation. *Mathematical Social Sciences*, 55(1):96–102, 2008.

[39] Lirong Xia. Quantitative extensions of the condorcet jury theorem with strategic agents. In *AAAI*, pages 644–650, 2016.

[40] Guoliang Xue and Yinyu Ye. An efficient algorithm for minimizing a sum of p-norms. *SIAM Journal on Optimization*, 10(2):551–579, 2000.

[41] Jiyan Yang, Yin-Lam Chow, Christopher Ré, and Michael W. Mahoney. Weighted SGD for ℓ_p regression with randomized preconditioning. In *Proceedings of the 27th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 558–569. SIAM, 2016.

[42] H. Peyton Young. Condorcet's theory of voting. *American Political Science Review*, 82: 1231–1244, 1988.

[43] H. Peyton Young. Optimal voting rules. *Journal of Economic Perspectives*, 9(1):51–64, 1995.

[44] H. Peyton Young and Arthur Levenglick. A consistent extension of Condorcet's election principle. *SIAM Journal of Applied Mathematics*, 35(2):285–300, 1978.

[45] Hanrui Zhang, Yu Cheng, and Vincent Conitzer. A better algorithm for societal tradeoffs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2229–2236, 2019.

[46] Zhibing Zhao, Tristan Villamil, and Lirong Xia. Learning mixtures of random utility models. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

A Omitted Proofs

Proof of Lemma 1. Consider any pair of contestants a and b and all $k = m/\binom{n}{2}$ pairwise contest results between a and b: $\{(a, b, y_i)\}_{i=1}^k$. Let $t_{a,i}$ and $t_{b,i}$ be the results of a and b in the i-th race, we have $y_i = t_{a,i} - t_{b,i}$.

Observe that $\sum_{i=1}^k (x_a - x_b - y_i)^2$ is minimized when $x_a - x_b = \frac{1}{k}\sum_i y_i$, and choosing $x_a = \frac{1}{k}\sum_i t_{a,i}$ and $x_b = \frac{1}{k}\sum_i t_{b,i}$ satisfies this condition. Thus, by setting x to be the mean vector of all contest results, we minimize the ℓ_2-loss between all pairs of contestants simultaneously. □

Proof of Proposition 1. The MLE selects a vector x that maximizes the following quantity (which is proportional to the likelihood of x): $\prod_i \exp(-|y_i - (x_{a_i} - x_{b_i})|^p)$. The logarithm of this quantities is $-\sum_i |y_i - (x_{a_i} - x_{b_i})|^p$. Maximizing this quantity is equivalent to minimizing $\sum_i |y_i - (x_{a_i} - x_{b_i})|^p$, which is exactly the objective function of ℓ_p-loss QJA. □

Proof of Theorem 1. Given a QJA instance (V, E) with $|V| = n$, $E = \{a_i, b_i, y_i\}_{i=1}^m$ and weights $\{w_i\}_{i=1}^m$, we construct an ℓ_p-regression instance (A, z) as follows. The matrix A has dimensions $m \times n$; for the i-th row of A, let $A_{i,a_i} = w_i$, $A_{i,b_i} = -w_i$, and $A_{i,j} = 0$ for all other $j \notin \{a_i, b_i\}$. Let z denote the n-dimensional vector with $z_i = w_i y_i$. We conclude the proof by noting that any optimal solution x to the ℓ_p-regression instance (A, z) is an optimal solution to the original QJA instance (V, E). This is because x minimizes $\|Ax - z\|_p^p = \sum_{i=1}^m ((Ax)_i - z_i)^p = \sum_{i=1}^m w_i |x_{a_i} - x_{b_i} - y_i|^p$. □

Proof of Theorem 2. We first show that ℓ_2-regression corresponds to solving a linear system. Let $f(x)$ denote the objective value of ℓ_2-regression at x. The goal is to minimize

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

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

Let $L = A^\top A$. Because A has only two non-zero entries in each row and they sum up to 0, we can view A as an edge-vertex incidence matrix and thus L is a graph Laplacian. [3]

Finally, we invoke the main result of [6]: given an $n \times n$ Laplacian matrix L with m non-zero entries, one can compute an ε-approximate solution to a linear system in L in time $O(m\sqrt{\log n}\log(1/\varepsilon))$. □

Proof of Theorem 3. We reduce Max-Cut to QJA. For a Max-Cut instance $G = (V, E)$ where $|V| = n$ and $|E| = m$, we construct an instance with $n + 2$ activities, $\{0, 1\} \cup V$, and $O(n + m)$ weighted edges:

- From 1 to 0 with difference 1 and weight w_1.
- From 0 to u for $u \in V$ with difference 0 and weight w_2.
- From 1 to u for $u \in V$ with difference 0 and weight w_2.
- From u to v for $(u, v) \in E$ with difference 1 and weight 1.

We further require that $w_1 > n w_2$ and $w_2 > \frac{2n}{1-p}$.

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

Lemma 4. *Any optimal solution of the ℓ_p-regression 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 4.

Lemma 5. *For any $d \in (0, \frac{1}{2}]$, $p \in (0, 1)$,*

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

[3] A graph Laplacian matrix L satisfies $L_{i,j} < 0$ for any $i \neq j$, and $L_{i,i} = \sum_{j \neq i} |L_{i,j}|$.

Proof. 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 $\left(0, \frac{1}{2}\right]$ and continuous at 0. Now we only have to check that $f(0) \geq 0$, which is trivial, and $f\left(\frac{1}{2}\right) \geq 0$. For the latter, let

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

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

Proof of Lemma 4. Let $x(a)$ be the potential of activity a. W.l.o.g. assume that in any solution, $x(0) = 0$. We first show that if $x(1) \neq 1$, then moving it to 1 strictly improves the solution. Suppose $|x(1) - 1| = d$. By moving $x(1)$ to 1, we decrease the cost on edge $(1,0)$ by $w_1 d^p$. For any other edge $(1,u)$ incident on 1, the cost can increase by no more than $w_2 d^p$, since

$$|x(1) - x(u) - d|^p \leq |x(1) - x(u)|^p + d^p.$$

Overall, the cost decreases by at least

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

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

- $0 < x(u) < 1$. W.l.o.g. $0 < x(u) \leq \frac{1}{2}$ and we try to move it to 0 by a displacement of $d = x(u)$. The total cost on $(0,u)$ and $(1,u)$ decreases by $w_2(d^p + (1-d)^p - 1)$, while the total cost on edges of form (u,v) and (v,u) can increase by no more than $n(d^p + (2+d)^p - 2^p)$. We show that the former amount is larger by taking the ratio.
$$\begin{aligned} \frac{w_2(d^p + (1-d)^p - 1)}{n(d^p + (2+d)^p - 2^p)} &> \frac{2(d^p + (1-d)^p - 1)}{(1-p)(d^p + (2+d)^p - 2^p)} \\ &\geq \frac{2(d^p + (1-d)^p - 1)}{2(1-p)d^p} \\ &= \frac{d^p + (1-d)^p - 1}{(1-p)d^p} \\ &\geq \frac{d^p - pd^p}{(1-p)d^p} \\ &= 1. \end{aligned}$$
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 cost on $(0,u)$ and $(1,u)$ decreases by $w_2(d^p + (1+d)^p - 1)$, while the total cost on edges of form (u,v) and (v,u) can increase by no more than $n(d^p + (2+d)^p - 2^p)$. It is easy to see that
$$\begin{aligned} n(d^p + (2+d)^p - 2^p) &< n(d^p + (1+d)^p - 1) \\ &< \frac{2n}{1-p}(d^p + (1+d)^p - 1) \\ &< w_2(d^p + (1+d)^p - 1). \end{aligned}$$

We conclude that in any optimal solution, $x(0) = 0$, $x(1) = 1$, and for any $u \in V$, $x(u) \in \{0,1\}$. □

Lemma 6. *A Max-Cut instance has a solution of size at least k iff its corresponding L_p-regression instance has a solution of cost at most $nw_2 + 2(m-k) + k2^p$. Moreover, with such a solution to the L_p-regression instance, one can construct the Max-Cut solution of the claimed size.*

Proof. Given a Max-Cut solution of size at least k, setting the potentials of the two vertex sets to be 0 and 1 respectively gives a L_p-regression solution with cost at most $nw_2 + 2(m-k) + k2^p$.

Given a ℓ_p-regression solution of cost at most $nw_1 + 2(m-k) + k2^p$, we first round the solution to the form stated in Lemma 4. 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. □

Any approximation with an additive error less than $2 - 2^p$ can be rounded to produce an optimal solution to the optimization version of Max-Cut, not to mention Max-Cut itself is APX-hard. □

B More Details of the Experiments

Datasets.

- *Programming contests.* We use data from Codeforces (`codeforces.com`), a website hosting frequent online programming contests. In each contest, a score is given to each participant according to their performance. We train on up to 50 consecutive contests to predict the next one. Contests used for training and testing are all Division 1 contests (meaning only more skilled users can participate) ranging from "VK Cup 2016 - Round 1" to "Codeforces Round #467". The corresponding time interval is about 2 years. There are usually 500 to 1000 contestants in each contest.
- *NYC and Boston marathons.* Contests used for training and testing include all NYC and Boston marathons from 2013 to 2017 in chronological order (i.e. Boston 2013 followed by NYC 2013 followed by Boston 2014, etc.). We use data from `marathonguide.com`, which publishes results of all major marathon events. In each marathon, the score of a runner is her finishing time in seconds. The numbers of finishers of both marathons are usually very large (more than 20000). We take only the first 1000 finishers in each marathon for training and evaluation. We train QJA on up to 5 (corresponding to a time interval of about 2.5 years) consecutive contests.
- *Formula One races.* We use all 20 Formula One races in 2017 (data from `www.formula1.com`). In each race, the score of a driver is the difference (in seconds) between her finishing time and the first finisher's. If a driver fails to finish, her score is 100 times the number of laps by which she is behind the first finisher. There are 20 drivers in each race, and 24 drivers in total. Almost all drivers are present in all races.
- *SAT solver competitions.* SAT Competition is a competitive event for solvers of the Boolean Satisfiability (SAT) problem (`satcompetition.org`). We focus on the main-track results of 2017. Each SAT instance is treated as a contest, and each solver as a contestant. Since most instances are solved by few solvers, we restricted our attention to instances where more than 90% of the solvers terminate. The score of a solver, when it terminates, is the time used in seconds. If a solver does not terminate, its score is the time limit. There are 32 contestants (solvers) and 46 qualifying contests (instances). Note that for the SAT solver dataset, all contestants (solvers) appear in each contest (instance).

Benchmarks. We evaluate ℓ_1 and ℓ_2 QJA against the following benchmarks. Means and medians are like QJA in that they use numerical information, but unlike QJA in that they use absolute instead of relative numbers; i.e., they do not correct for the difficulty of the individual contest. In one-dimensional environments like ours, means and medians are considered to be among the best imputation methods for various tasks (see, e.g., [16, 34]). Kemeny-Young and Borda are like QJA in that they are social-choice-theoretic methods that use relative rather than absolute performance, but unlike QJA in that they do not use numerical information, only the ranking of contestants. Among social-choice-theoretic methods, Kemeny-Young and Borda both have natural interpretations as estimating a "ground truth" ranking of the contestants [42, 43, 7, 38, 8]. We do not claim that these methods are superior to all other ranking-based methods (such as, for example, random utility models) for all purposes; but our measure of accuracy, explained below, strongly favors Kemeny-Young. Specifically, among ranking-based methods, Kemeny-Young optimizes exactly our measure of accuracy on the training set, giving it an advantage over every other ranking-based method. (Indeed, we will see that it significantly outperforms Borda.)

- *Means.* For every contestant in the training set, we take the mean of her scores in all training contests in which she participates. This quantitative aggregation also induces an ordinal aggregation, simply by sorting the means. Note that if all contestants appear in every contest (as in the SAT solver dataset), then ℓ_2 QJA is equivalent to taking means.
- *Medians.* For every contestant in the training set, we take the median of her scores in all training contests in which she participates. Medians more closely resembles ℓ_1 QJA.[4]
- *The Kemeny-Young rule [24, 44, 42].* This is a voting method that takes multiple (partial) rankings of the contestants as input. We obtain these rankings from the training contests. It outputs ordinal predictions only; specifically, it outputs a ranking that minimizes the number of *disagreements* on pairs of contestants with the input rankings. This rule has a natural interpretation as estimating

[4]For example, if there are only two contestants who participate in every contest, and one obtains the exact same score in every contest, then ℓ_1 QJA comes down to the median score of the other.

the "correct" ranking [42, 43], and is again closely related to ℓ_1 QJA and medians.[5] Finding the Kemeny-Young outcome is known to be NP-hard. There are exact methods based on integer programming that can solve instances with up to hundreds of candidates in practice, but for larger-scale data (like the datasets we consider), those methods do not scale. It is therefore unrealistic to compare against certifiably optimal Kemeny-Young outcomes, or to make predictions based on such outcomes in practice. We use an open-source heuristic solver[6]. Based on our tests on smaller instances, the heuristic solver almost always produces outcomes of the same or similar quality to the certifiably optimal ones.

- *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 participant in contest j receives $1 - 2(i-1)/(n_j - 1)$ points, where n_j is the number of participants in the contest. The aggregated ranking result is obtained by sorting the contestants by their total points. Borda can be viewed as a variant of means [7], and is therefore also closely related to ℓ_2 QJA.
- *Ratings* (for the Codeforces dataset only). Codeforces maintains ratings for all users, using a variant of the Elo rating system [14]. The ratings are calculated based on all previous contests, and are intended to be predictive. We use the ratings right before each contest to predict its result. Note that although the ratings are quantitative, they are not on a scale related to the actual scores of the contestants, so we only use them to provide ordinal predictions.
- *TrueSkill* (for the SAT solver dataset only). TrueSkill is a Bayesian skill rating system. The official python package (see `https://trueskill.org/`) does not work for contests consisting of more than 130 participants, so we only test it on the relatively small SAT solver dataset. It is observed that TrueSkill generally outperforms the Elo rating system [20], so we only compare to the former. We remark that the prediction of TrueSkill depends considerably on how we order contests in the training set. To calibrate for this uncertainty, we shuffle the contests and take the average of error rates across 100 runs. In each run, the error rate is calculated in the same way as described in the paper.

There are many other, related approaches that deserve mention in this context, but that are not ideal to compare to because they do not exactly fit our setting and/or motivation. There is work on predicting differences in final scores in football [27, 23, 19], but our setting is different, because a football match always has only two contestants. Similarly, the Bradley-Terry model [21] uses only pairwise comparisons. Thurstone's model [37], like the voting methods, only takes ranking information as input. QJA in some ways resembles matrix factorization, but the focus in that line of work tends to be on recovering the ground truth from a relatively small number of entries, rather than aggregating results in a simple, transparent manner.

Contestants to make predictions for. Not every contestant appears in every contest, and some contestants in the test contest may not appear in *any* of the training contests. Making predictions for such contestants only muddles the evaluation of our methods. On the other hand, suppose we make predictions for all the candidates that have appeared in *some* training set. Then, as the size of the training set increases, we make predictions for more participants who do not participate regularly, potentially worsening the accuracy. Also, as the size of the training set grows, more and more contestants may become seen and therefore more or less predictable. But on the other hand, to study the effect of the growth of the training set, we want to keep the set of contestants to predict stable. We therefore try to make predictions for the largest possible set of contestants that is meaningful and stable when the size of the training set grows: predict only for the set of contestants that appear in the testing set and the chronologically last training contest. For the Codeforces dataset, we have no less than 130, normally about 170 core contestants. For the marathon dataset, we have no less than 40, normally about 60 core contestants. For the SAT solver competition dataset, the number is always 32 since every contestant participates in every contest. As we will see, in all of our datasets, there are enough core contestants to predict to suppress variance across contestants.

Multiplicative vs. additive differences. Our aim is to consider and predict *relative* performance of contestants, but this can be interpreted in multiple ways. In the experiments, we consider *additive* differences between contestants ("A finished 30 seconds before B"), instead of using log to translate

[5]To see the similarity, one may again consider the example with two contestants, one of which gets the same score every time. The outcomes of the three rules are all determined by the median score of the other contestant.

[6]See `numerical.recipes/whp/ky/kemenyyoung.html`

[7]Borda assigns scores to contestants linear in their ranks, and when every contestant participates in all contests, the output ranking is induced by the mean of the scores assigned.

to multiplicative differences ("A's finish time was 0.98 of B's"). For all our benchmarks, considering additive differences does not make any difference in how they are computed; but, for QJA, it does matter. This only affects QJA. Experimental results show that the choice affects ℓ_1 QJA to only a barely noticeable extent. This is perhaps not surprising because ℓ_1 QJA is more similar to median-based approaches that are invariant to any monotone transformation. In contrast, experiments show that working with additive differences makes ℓ_2 QJA more accurate. Hence, we only report results in the additive model.

C Axiomatic Characterization

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

Recall that for each pair of contestants a and b where a outperforms b by y, the absolute error of the prediction vector $\{x\}$ on this pair is $|x_a - x_b - y|$. Using this as the loss function, we obtain the ℓ_1 QJA rule, which has been characterized using axioms in the context of social choice theory [10]. Below we extend this characterization to ℓ_p QJA 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 QJA is continuous in p.

The idea is to 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 QJA, it is easy to see that the transformation function is $f(t) = t^p$. To characterize QJA 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 F be a family of transformation functions. Below are the axioms we consider:

- *Identity.* There is an identity transformation $f_0 \in F$, such that $f_0(t) = t$ for any $t \geq 0$.
- *Invertibility.* For each $f_1 \in F$, there is an $f_2 \in F$ 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 $f_1, f_2 \in F$, there exists $f_3 \in F$ 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 F^* corresponding to the QJA rules is the minimum family of functions satisfying the above axioms.

By the first axiom, the identity transformation f_0 where $f_0(t) = t$ is in F^* — this corresponds to ℓ_1 QJA. Now by the third axiom, for any $k \in \mathbb{Z}_+$, f_0^k is also in F, where $f_0^k(t) = t^k$. By the second axiom, for any $k \in \mathbb{Z}_+$, $f_0^{1/k}$ is also in 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 F^*, where $f_0^r(t) = t^r$.

Note that the above argument establishes that F^* contains all transformation functions corresponding to QJA, i.e.,
$$\{t^r \mid r \in \mathbb{Q}_+\} \subseteq F^*.$$
Below we show the other direction, i.e., $\{t^r \mid r \in \mathbb{Q}_+\}$ satisfy the 3 axioms, and as a result,
$$F^* \subseteq \{t^r \mid r \in \mathbb{Q}_+\}.$$
For $f_1(t) = t^{r_1}$ and $f_2(t) = t^{r_2}$ 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 $F^* = \{t^r \mid r \in \mathbb{Q}_+\}$ as desired.