Who is the Winning Algorithm? Rank Aggregation for Comparative Studies

Anonymous authors
Paper under double-blind review

Abstract

Consider a collection of m competing machine learning algorithms. Given their performance on a benchmark of datasets, we would like to identify the best performing algorithm. Specifically, which algorithm is most likely to "win" (rank highest) on a future, unseen dataset. The standard maximum likelihood approach suggests counting the number of wins per each algorithm. In this work, we argue that there is much more information in the complete rankings. That is, the number of times that each algorithm finished second, third and so forth. Yet, it is not entirely clear how to effectively utilize this information for our purpose. In this work, we study the problem of estimating future win probabilities for each of the m algorithms, and propose a statistically motivated weighting scheme that effectively incorporates the complete rankings over a benchmark of datasets. Our proposed framework improves upon currently known methods in synthetic and real-world examples.

1 Introduction

Introducing a new machine learning (ML) algorithm is a complex and demanding task. It often begins with a novel concept, a fresh perspective on an existing problem, or insights borrowed from other disciplines. The development process typically involves theoretical analysis, computational considerations, and extensive implementation work. Once the research and development phase is complete, we aim to assess how well the new algorithm performs compared to existing methods. In the ML community, this evaluation is usually done by testing the algorithm on a suite of benchmark datasets and comparing its results to those of state-of-the-art approaches. However, such comparisons rarely yield clear conclusions. Ideally, a strong algorithm should achieve high rankings across many datasets. But how do we confidently conclude that it outperforms others? Consider, for example, three algorithms evaluated on five datasets. Suppose that one algorithm ranks first on one dataset and second on the remaining four. Another algorithm wins two datasets but ranks third in the rest. Which algorithm is better? Shall we just count the number of wins? Consider the average rank? Or perhaps present the full ranking and leave the interpretation to the reader?

In this work, we address the fundamental problem of identifying the best-performing algorithm based on benchmark results. We argue that a natural measure of interest is the probability of winning a future dataset. For each algorithm, our goal is to estimate the *win probability* – the likelihood that it will outperform its competitors in a new predictive modeling task. This measure is particularly interesting in real-world applications, where the objective is to choose the most promising algorithm for deployment. While this problem has been previously explored, as discussed in Section 2, a common approach based on maximum likelihood estimation (MLE), which essentially amounts to counting the number of dataset wins for each algorithm. However, we argue that there is much more information in the complete rankings. The task of effectively leveraging this information falls under the broader scope of rank aggregation, a topic that has received notable attention in the ML literature. Nonetheless, existing methods are often heuristic in nature or rely on strong probabilistic assumptions. We show that accurately estimating win probabilities from full ranking data is a challenging and non-trivial task.

In this work, we introduce a risk-based weighting approach for estimating the win probability in comparative studies. We propose a simple and intuitive scheme that expresses the estimated win probability as a weighted

linear combination of the rankings an algorithm achieves across benchmark datasets. The key challenge lies in finding the optimal weights which minimize the estimation error, measured in terms of total variation (TV) distance and Kullback-Leibler (KL) divergence. Naturally, our estimator generalizes the MLE, which corresponds to the special case where all weight is placed solely on dataset wins. We propose two weighting strategies: the first is a conservative, data-independent scheme, where weights are fixed in advance and designed to control worst-case estimation error. The second is a flexible, data-dependent scheme, where weights are adapted based on the observed data. We evaluate both schemes through synthetic and real-world experiments, demonstrating their effectiveness in generating more accurate and reliable comparisons among competing algorithms. Notably, our method leads to clearer and more robust conclusions about algorithmic performance in comparative studies. An implementation of our framework is publicly available on the author's webpage[1].

It is important to emphasize the practical contribution of our work. In comparative studies, multiple machine learning algorithms are typically evaluated across a range of datasets. Naturally, data-handling, preprocessing and hyperparameter tuning have a significant impact on reported results. In this study, we implicitly assume that the performance of each algorithm reflects the end-to-end pipeline, according to recommended guidelines. In that sense, these results represent the outcome a practitioner could realistically achieve. Our contribution lies in proposing a rank-aggregation approach for estimating the win probabilities based on these results. That is, although our estimator has the form of a positional scoring rule, our focus is on statistical estimation of future win probabilities rather than preference aggregation.

2 Related Work

Evaluating multiple algorithms across a collection of datasets is a common practice. Numerous methods have been proposed for this purpose, differing in their objectives, statistical foundations, presentation style (qualitative or quantitative), visualization strategies, and more. In this section, we provide a brief overview of some of the more widely used approaches and examine their strengths and limitations in the context of our objectives.

The *average performance* is a simple averaging of the algorithms' performance over all the benchmark datasets, according to a predetermined measure of interest. For example, the averaged mean square error (MSE) for regression problems. While this measure is simple and intuitive, it fails to capture the differences between the datasets. For example, consider two datasets where the typical MSE in one dataset is of scale of 10^{-1}, while in the other is of scale of 10^2. It is quite misleading to directly average them, since the former would be negligible compared to the latter.

To address this issue, the *average ranking* approach considers the mean rank that each algorithm attains across a collection of datasets. In this framework, the best-performing algorithm on a given dataset is assigned rank first, the second-best is ranked second, and so on. Consequently, a lower average rank indicates stronger overall performance across all datasets. This method is widely used for comparing multiple algorithms simultaneously (Demšar, 2006; Fernández-Delgado et al., 2014), largely due to its simplicity and interpretability. However, the average ranking method has notable limitations. One key drawback is its implicit assumption that all rank differences carry equal weight. That is, the gap between ranking first and fifth is treated the same as the gap between ranking 21st and 25th. This assumption is problematic: while a first-versus-fifth difference often signals performance superiority, the difference between 21st and 25th is typically negligible, especially when many algorithms are being compared. To mitigate this, weighted averaging can be used. A classical example is the *Borda Count* method (Emerson, 2013), which assigns each rank a weight based on how many ranks fall below it. For example, a first-place finish earns a weight of $m-1$, second place $m-2$, and so forth. Although Borda Count is intuitive and easy to implement, it remains a heuristic and lacks statistical performance guarantees.

Statistical comparisons is yet another alternative to assess dominance among multiple algorithms. Here, we define a single or multiple hypotheses and test whether the data agree with it. A common example is pairwise comparisons. For every pair of algorithms we test the null hypothesis that they perform equally well. We

[1] `https://anonymous.4open.science/r/WhoIstheBestAlgorithm-6780/`

repeat this process for every pair of algorithms and correct for multiplicity (Demšar, 2006). Unfortunately, this routine is not always adequate. First, the number of hypotheses grows quadratically with the number of algorithms, which makes the multiplicity correction very conservative. Second, the result may not be clear enough. That is, a typical outcome may be "Algorithm A is better than C, and B is better than E", as the rest of comparisons do not yield statistically significant conclusions. Naturally, there exist more sophisticated statistical tests for the problem. The ANOVA is a parametric test (Fisher, 1956). It defines a null hypothesis that all the algorithms perform the same and the observed differences are merely random. Unfortunately, this approach does not suggest which is the better algorithm. The Friedman test (Friedman, 1937; 1940) is a non-parametric counter-part of the ANOVA. It considers the same null as ANOVA and defines a rank statistic with a known null distribution. If the null hypothesis is rejected, one may proceed with a post-hoc test. The Nemenyi test (Nemenyi, 1963) is similar to the Tukey test for ANOVA (Tukey, 1949) and is used when all algorithms are compared to each other. The performance of two algorithms is significantly different if the corresponding average ranks differ by at least a critical difference, defined in (Demšar, 2006). There exist additional alternative tests, such as the Bonferroni-Dunn test (Dunn, 1961), Hommel's procedure (Hommel, 1988), Holm's procedure (Holm, 1979) and others. The interested reader is referred to Section 3.2 of Demšar (2006) for a comprehensive discussion. The statistical methods mentioned above differ from one to another in several ways. However, they all have a common advantage. They provide a solid framework for deciding on the dominant algorithms, if such exist. However, these methods do not provide a decisive conclusion in cases where the differences are less evident. Furthermore, seeking statistically significant differences on the given dataset does not necessarily imply that an algorithm would outperform its alternatives on a future dataset. In fact, this is one of the major differences between statistical inference and prediction. We further discuss this issue later in Section 6. It is worth mentioning that statistical comparisons typically accompany other measures. For example, it is very common to report the average ranking together with a statistical test that validates which rankings are significant. The presentation of such results are provided in tabular form or as visual representation such as critical difference diagrams, heat-maps, box-plots and others (Ye et al., 2024).

Another approach to assess dominance is the Bradley–Terry model (Bradley & Terry, 1952). Here, the outcome of pairwise comparisons between two algorithms, A and B, is modeled by $\mathbb{P}(A > B) = \alpha_A/(\alpha_A+\alpha_B)$ where $A > B$ means that Algorithm A is "better" than B, and α_A is a positive real-valued score assigned to Algorithm A. The Bradley–Terry model can be generalizes to multiple competitors such that $\mathbb{P}(A > \{B, C, D\}) = \alpha_A/(\alpha_A + \alpha_B + \alpha_C + \alpha_D)$. This model is known in the literature as the Plackett-Luce model (see Section 5 of Hunter (2004)). Given the performance on a benchmark of datasets, we may write a likelihood function according to the model and obtain maximum likelihood estimates for the α's. The Bradley–Terry and the Plackett-Luce models are very popular in sport tournaments, commercial applications and other domains (Hunter, 2004). Indeed, they strive to provide a direct answer to our problem. Yet, these models provide a principled parametric approach to ranking aggregation. Our focus differs in that we seek a nonparametric estimator of future win probabilities and therefore avoid imposing a specific ranking model.

Recently, Fernández-Delgado et al. (2014) introduced an alternative measure of dominance. The *Probability of Achieving the Best Accuracy* (PAMA) criterion measures the proportion of datasets on which a given algorithm attains the highest accuracy. Notably, PAMA focuses solely on the top-ranked algorithm for each dataset, implicitly assuming that win probability is determined only by past wins, rather than the complete rankings. Yet, this approach directly assess our objective of interest. Building on this concept, our work extends the idea by estimating win probability using complete ranking information.

3 Definitions and Problem Statement

Consider a collection of m ML algorithms, applied to n independent datasets. Let X_i be a vector of rankings of the m algorithms over the ith dataset. For example, $x_i = [2, 3, 1, 4]$ indicates that Algorithm 2 achieved the best performance on the ith dataset, followed by Algorithms $3, 1, 4$ respectively. Naturally, there are $m!$ possible rankings for the m algorithms. We denote the probability vector of all possible rankings as P and assume that X_i are independent and identically distributed observations from P. This model implies two basic assumptions. First, the rankings of the algorithms are independent between datasets (X_i are independent variables). Second, the datasets are modeled as draws from a common ranking distribution P

(X_i are identically distributed). Although these assumptions may seem strong, they are commonly used by most of the methods discussed in Section 2. In practice, benchmark collections may contain heterogeneous subclasses of datasets (for example, regression versus classification tasks, small versus large tabular datasets, or datasets with differing feature characteristics). When such structure is known, the proposed methodology can be applied separately within each subgroup, yielding subgroup-specific estimates of the win probabilities.

A natural objective is to estimate P from a collection of observations $x^n = x_1, ..., x_n$. However, this task is both very difficult and unnecessary in practice. First, notice that the alphabet size of P (the number of possible outcomes) is $m!$ which may be much greater than n. In fact, comparing ten algorithms we have $10! \approx 3.6M$ possible outcomes (rankings), while a typical sample only consists of tens (or hundreds) of datasets. Second, we are practically not interested in the complete rankings of all algorithms. Our goal is to estimate the probability that the jth algorithm wins a future dataset, for $j = 1, ..., m$. We denote the win probability vector as p. Notice that p_j is the sum of all the rankings in P in which the jth algorithm finishes first. Formally, define $\mathcal{C}_j$ as the collection of all possible rankings in which the jth algorithm is top-ranked. Then, $p_j = \sum_{l \in \mathcal{C}_j} P_l$. Thus, our objective is to estimate the much smaller vector p (of size m) rather than the full distribution P (of size $m!$). Since rankings can be obtained by sorting the coordinates of p, estimating p provides a probabilistic characterization of algorithm dominance rather than only an ordinal ranking. Consequently, estimating p provides both a ranking and a calibrated measure of uncertainty regarding algorithm dominance.

In this work we focus on two fundamental divergence measures between p and its estimate $\hat{p}$. Namely, the *total variation* (TV) and the Kullback-Leibler (KL) Divergence,

$$D_{\mathrm{TV}}(p, \hat{p}) = \sum_{j=1}^{m} |p_j - \hat{p}_j|, \quad D_{\mathrm{KL}}(p||\hat{p}) = \sum_{j=1}^{m} p_j \log \frac{p_j}{\hat{p}_j} \tag{1}$$

respectively (Jiao et al., 2015; Painsky & Wornell, 2018). TV and KL are standard losses for probability estimation and quantify both ranking uncertainty and probability calibration. Since both measures are random (as they depend on $\hat{p}$), we focus on their expected performance, also denoted as risk, $\mathbb{E}D_{\mathrm{TV}}(p, \hat{p})$ and $\mathbb{E}D_{\mathrm{KL}}(p||\hat{p})$.

A natural estimator for p is of course the maximum likelihood estimator (MLE), $\hat{p}_{ml}$. Given a sample $x_1, ..., x_n$, we define its likelihood as

$$\mathcal{L} = \frac{n!}{\prod_{l=1}^{m!} r_l!} \prod_{l=1}^{m!} P_l^{r_l}$$

where r_l is the number of times that the lth ranking appears in the sample. Notice $\mathcal{L}$ is simply a multinomial likelihood function over P, and the corresponding MLE is given by $\hat{P}^{ml} = r/n$. As mentioned above, we are interested in a MLE for p. Hence, by the basic properties of the MLE we have that $\hat{p}^{ml} = r^{(1)}/n$ where $r_j^{(1)}$ is the number of times that the jth algorithm is top-ranked. In other words, the MLE corresponds to the proportion of datasets each algorithm wins. This is exactly the PAMA criterion, as discussed in Section 3.

In this work we seek $\hat{p}$ that minimizes the expected divergence (1). Our point of departure is the MLE, which only considers $r^{(1)}$, the number of wins per each algorithm. We argue that there may be more valuable information about p in additional statistics, such as the number of times that each algorithm finishes second, third and so forth. Hence, we propose the following estimator

$$\hat{p}^{w} = \sum_{j=1}^{m} \frac{w_j}{n} r^{(j)} \tag{2}$$

where $r^{(j)}$ corresponds to the number of times each algorithm is ranked in the jth position and w_j are non-negative weights satisfying $\sum_j w_j = 1$. The coefficient w_j represents the contribution of a jth place finish toward estimating future wins. Larger values indicate that the corresponding rank position carries information about future first-place performance. For example, $w_2 > 0$ reflects the belief that frequent runner-up finishes may provide evidence of future victories. In other words, this model generalizes the MLE

(which corresponds to the special case where $w_1 = 1$) by allowing the weight to be distributed across all ranks. Notice that $\hat{p}_j^w \geq 0$ and $\sum_j \hat{p}_j^w = 1$ for every vector of w's defined above, as desired. Hence, our goal is to minimize the expected divergence (1) over a choice of w. We introduce two complementary strategies to the problem. The first is a worst-case, data-independent strategy, where the weights w are fixed prior to observing any data. The second is a data-dependent approach, in which the weights are determined adaptively based on the observed sample.

Equation (2) resembles classical positional scoring rules such as Borda Count and related scoring-function approaches. The distinction is conceptual rather than algebraic. Classical scoring rules aggregate rankings to produce an ordering of alternatives, whereas our objective is estimation of future win probabilities. Consequently, we study the choice of rank-position weights through statistical risk criteria and cross-validation rather than social-choice axioms or preference aggregation criteria.

Our analysis focuses on estimation of the win-probability vector p. While accurate estimation of p is a natural objective for predicting future winners, it does not directly imply guarantees for winner recovery, top-k identification, or ranking accuracy without additional assumptions on the separation between competing algorithms. Studying such guarantees is an interesting direction for future work.

4 The Data Independent Scheme

We begin our analysis by considering the expected total variation under our proposed model with fixed weights. Specifically, we would like to minimize $\mathbb{E}D_{\mathrm{TV}}(p||\hat{p}^w)$ with respect to a fixed vector w. Notice that the main difficulty with this approach is that p is unknown in practice. This means that the obtained weights would also depend on the unknown p. Therefore, we take a more conservative minimax approach and consider a universal scheme that minimizes the expected divergence for the worst-case scenario. Formally, define $\Delta_{m!}$ as the collection of all possible distributions over an alphabet size $m!$. Further, let $\mathcal{W}$ be the collection of all w such that $w_j \geq 0$ for $j = 1, ..., m$ and $\sum_{j=1}^m w_j = 1$. Then, our goal is

$$\min_{w \in \mathcal{W}} \max_{P \in \Delta_{m!}} \mathbb{E}D_{\mathrm{TV}}\left(p, \hat{p}^w\right). \tag{3}$$

Unfortunately, this minimax problem in (3) is not an easy task. Notice that for a choice of $w_1 = 1$, it degenerates to

$$\max_{p \in \Delta_m} \mathbb{E}D_{\mathrm{TV}}\left(p, \hat{p}^{ml}\right)$$

which corresponds to the worst-case convergence rate of the MLE. This problem was extensively studied over the years, with several key results. In fact, it was shown that

$$\max_{p \in \Delta_m} \mathbb{E}D_{\mathrm{TV}}\left(p, \hat{p}^{ml}\right) \leq \sqrt{\frac{m}{n}} \tag{4}$$

where the bound is asymptotically tight under a uniform distribution $p = 1/m$ (Jiao et al., 2015). Following these important contributions, we first introduce an upper bound to the worst-case expected divergence, and then minimize it with respect to $w \in \mathcal{W}$. Finally, we discuss the tightness of our results. For the simplicity of the presentation, we first focus on a simpler special case where $\hat{p}^w = (wr^{(1)} + (1-w)r^{(2)})/n$. That is, we only consider the number of times that every algorithm is top-ranked, $r^{(1)}$, or runner-up, $r^{(2)}$, over the given testbench. Theorem 1, whose proof is located in Section 7.1 , introduces our proposed upper bound for this setup.

Theorem 1. *Let $p \in \Delta_m$ be the win probability for each of the algorithms. Let $\hat{p}^w = (wr^{(1)} + (1-w)r^{(2)})/n$ be our proposed minimax estimator. Then,*

$$\max_{P \in \Delta_{m!}} \mathbb{E}D_{TV}(p, \hat{p}^w) \leq \sqrt{m}\sqrt{2(1-w)^2 + \frac{1}{n}(w^2 + (1-w)^2)}. \tag{5}$$

Further, the weight w^* which minimizes (5) is given by

$$w^* = 1 - \frac{1}{2n+2}.$$

Let us examine the obtained result. First, we observe that the minimax weight, $w^* = 1 - 1/(2n+2)$ approaches 1 for sufficiently large n. It is also relatively close to it, even for smaller n. Further, the proposed upper bound (5) is almost the same as the MLE's (4) for a choice of w^*. This is not quite surprising. For example, consider a case where one of the algorithms is always ranked second (with probability 1), while the rest of the algorithms are equally likely to finish first. In this worst-case scenario, putting weight on $r^{(2)}$ is counter-productive with respect to estimating p. Furthermore, the estimator's worst-case performance is governed by the (almost) uniform nature of p, leading to an (almost) similar bound as the MLE. In other words, we construct a distribution p for which the proposed bound in (5) becomes tight, as desired. Despite the above, the proposed minimax scheme with w^* may be useful in cases where n is relatively small and a conservative approach is required. We further discuss this scenario in Section 6. It is important to stress that the worst-case distribution that nearly achieves the bound in (5) is highly unlikely in practical scenarios. That is, w^* controls a worst-case "pathological" P, where one algorithm always ranks second while the rest are equally likely to be first. In real-world applications, such behavior is improbable, making the use of w^* overly conservative. On the other hand, imposing additional constraints on P to overcome this issue requires additional assumptions on the underlying model.

To further refine and generalize our results, we introduce Theorem 2 which extends the analysis to the top K ranks, $r^{(1)}, ..., r^{(K)}$. This theorem provides an upper bound that can be efficiently computed through a straightforward convex optimization over K parameters. The full proof of Theorem 2 is provided in Section 7.2.

Theorem 2. *Let $p \in \Delta_m$ be the probability that each algorithm wins. Let $\hat{p}^w = \frac{1}{n}\sum_{j=1}^{K} w_j r^{(j)}$ be our proposed estimator. Assume that $w_1 \geq w_2 \geq ... \geq w_K$ and $w_1 \geq \frac{8n-\sqrt{8n}}{8n-1} \approx 1 - 1/\sqrt{8n}$. Then,*

$$\mathbb{E}D_{TV}(p, \hat{p}^w) \leq \max_{t=[t_1,...t_K]^T \in \Delta_K} \sum_{j=1}^{K-1} \sqrt{((1-w_1)t_j - w_{j+1})^2 + \frac{1}{n}(w_1^2 t_j + w_2^2)} + \sqrt{(1-w_1)^2 t_K^2 + \frac{(m-K+1)}{n}(w_1^2 t_K + w_2^2(m-K+1))}$$

Further, under the stated conditions, the maximization problem above is convex in $t = [t_1, ...t_K]^T \in \Delta_K$.

Theorem 2 provides a worst-case upper bound on the expected TV between p and its corresponding linear estimator that incorporates the top K ranks. While this theorem enhances model flexibility, it does not yield a closed-form expression for the optimal weights w^* that minimize the bound. In practice, this means we must numerically evaluate various weight vectors w that satisfy the theorem's conditions and select the one which minimizes the bound.

Despite this limitation, Theorem 2 introduces a notable improvement over Theorem 1, as it allows for a richer model that leverages more ranking information. To illustrate this, we compare Theorem 2 for $K = 2$ and $K = 3$ with Theorem 1 and the MLE bound (4). Figure 1 presents the obtained upper bounds for $m = 5$ and varying values of n. As we can see, the $K = 2$ bound closely matches Theorem 1, with minor differences attributed to its numerical nature. Yet, we observe a notable improvement in our results as the model increases to $K = 3$, demonstrating the benefit of incorporating additional rank information.

These results should not be interpreted as establishing the optimality of our proposed scheme. Rather, they demonstrate that the proposed upper bound strictly improves upon the best currently known result (4). In other words, although (4) is known to be asymptotically tight (Jiao et al., 2015), its finite-sample behavior is subject to further study. Consequently, our contribution does not constitute an improvement over a provably tight MLE finite-sample result. Instead, we improve the strongest available upper bound to date, yielding refined performance guarantees. In the following sections, we complement our theoretical findings with a detailed study of the proposed bound, evaluated on both synthetic and real-world examples.

Next, we would like to consider a similar minimax approach for the expected KL divergence (1). Unfortunately, this task is quite problematic. Notice that the KL divergence is unbounded in cases where $p_j > 0$ while $\hat{p}_j = 0$. This means that any estimator that assigns zero probabilities is prone to obtain unbounded loss (and hence unbounded expected loss). For this reason, minimax analysis of discrete distributions under KL

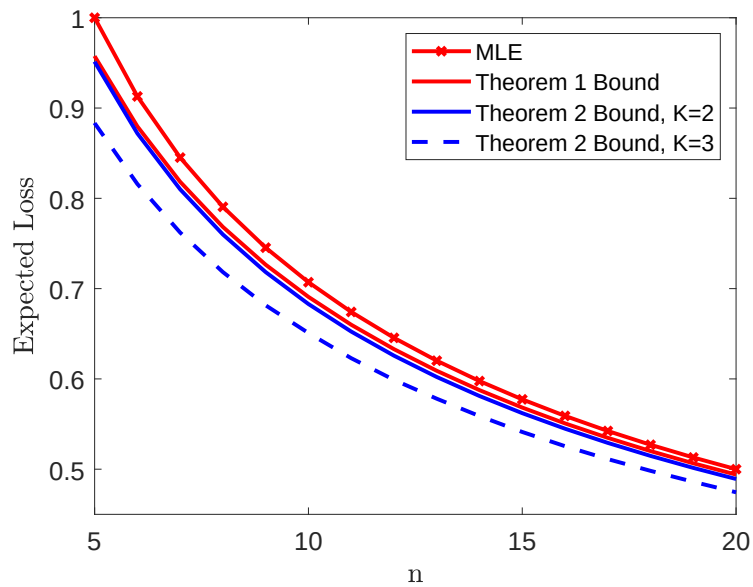

Figure 1: Data-independent upper bounds for $m = 5$, under the expected TV distance

divergence require additional assumptions and constraints, especially in a finite sample regime. See (Orlitsky & Suresh, 2015) for a comprehensive discussion. Therefore, we refrain from a worst-case KL analysis and study this divergence only in a data-dependent regime.

To conclude, the minimax approach provides data-independent weights w while considering the worst-case distribution $P \in \Delta_{m!}$. While inherently conservative, this strategy produces robust and non-trivial weights that offer a meaningful improvement over the classical MLE, especially in small-sample settings (see Section 6). In the following section we introduce an alternative data-dependent scheme which does not consider a worst-case distribution and obtains favorable weights which depend on the data in hand.

5 The Data Dependent Scheme

Ideally, we would like to minimize the expected loss, where the expectation is taken over the true underlying model P. Unfortunately, this model is unknown, and so we require an alternative approach. In Section 4 we focus on a minimax criterion, where we consider the worst-case underlying model. Naturally, this leads to conservative results. In this section we consider an empirical risk minimization scheme, which replaces the expectation with its empirical counter-part. This approach is perhaps the most common paradigm in learning theory, with many favorable properties. We begin our analysis with the KL divergence, and later introduce a complementary TV scheme. First, we observe that minimizing $\mathbb{E}D_{\mathrm{kl}}(p||\hat{p})$ with respect to $\hat{p}$ is equivalent to minimizing $-\mathbb{E}\sum_{j=1}^{m} p_j \log \hat{p}_j$. Hence, the corresponding empirical risk is given by

$$R^{\mathrm{KL}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbb{I}(y_i = j)\log(\hat{p}_j) = -\frac{1}{n}\sum_{j=1}^{m} r_j^{(1)} \log \hat{p}_j \tag{6}$$

where y_i denotes the winning algorithm over the ith dataset. This is exactly the well-known cross-entropy loss function. Plugging our proposed model (2) to the above and minimizing over w yields $w_1 = 1$, which corresponds to the MLE. This is not surprising as the MLE is known to be the minimizer of the cross-entropy function (over any choice of $\hat{p}$).

However, it is well known that directly minimizing the empirical loss may lead to overfitting the observed data. To address this issue, we adopt the widely used leave-one-out (loo) strategy as an alternative. Given a collection of n datasets and m algorithms, the proposed procedure is as follows: in each iteration, we remove

one dataset and compute the estimator, $\hat{p}^w$, based on the $n-1$ remaining datasets. Then, we evaluate the loss on the excluded dataset. This process is repeated for each of the n datasets, and we select the weight vector w that minimizes the average loss across all loo iterations. Formally, let $x^n = x_1, ..., x_n$ be the collection of rankings over all the datasets (as defined in Section 3). Denote $x^n[-i]$ the collection of all the datasets, with the ith dataset removed. Likewise, define $r^{(j)}[-i]$ as the number of times that each algorithm finished at the jth position, over the collection of datasets $x^n[-i]$. Then, the corresponding estimator is $\hat{p}^w[-i] = (1/n)\sum_{j=1}^m w_j r^{(j)}[-i]$. This serves as our loo estimator for the ith iteration. Therefore, the loo loss is given by

$$L_{loo}^{\mathrm{KL}} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbb{I}(y_i = j)\log(\hat{p}_j^w[-i]), \tag{7}$$

and w^{loo} is the vector of weights $w \in \mathcal{W}$ which minimizes (7). Notice that given x^n, the loo loss minimization (7) is convex in w and may be evaluated by standard convex optimization. Algorithm 1 below summarizes our proposed method.

Algorithm 1 Leave-one-out Estimation Scheme

Require: x^n, m

1: Define $x^n[-i]$ as the collection of all the dataset, excluding the ith dataset.
2: Define $r^{(j)}[-i]$ as the number of times that each algorithm finished at the jth position, over the collection $x^n[-i]$.
3: Denote $\hat{p}^w[-i] = (1/n)\sum_{j=1}^m w_j r^{(j)}[-i]$.
4: Minimize (7) with respect to $w \in \mathcal{W}$.
5: Define w^{loo} as the minimizer of Step 4.
6: **return** $\hat{p}^w = \sum_{j=1}^m w_j^{loo} r^{(j)}$.

As opposed to the minimax approach, the proposed loo scheme introduces data-dependent weights, which are less conservative and consider the data in hand. Notice that for every fixed w, the loo loss is an (almost) unbiased estimate of the expected loss $\mathbb{E}D_{\mathrm{KL}}(p||\hat{p}_w)$. Hence, we obtain an (almost) unbiased estimate for our desired measure of interest. The loo approach is not new to learning theory and practice. In fact, is also quite common in highly related parameter estimation problems. For example, the seminal Good-Turing estimator (Good, 1953), which is perhaps the most common scheme for distribution estimation over large alphabets (that is, where $m \gg n$) can also be interpreted as a loo scheme (McAllester & Schapire, 2000). In that sense, our proposed method utilizes a similar conceptual framework of parameter estimation in a discrete and relatively large alphabet regime, as later discussed in Section 6. While Algorithm 1 allows minimizing the loo loss with respect to w_j, where $j = 1, ..., m$, we typically restrict our attention to simpler models where $j \leq 3$. This corresponds to a model which only considers the top three ranks in each dataset (similarly to $K = 3$ in Theorem 2). We observe that for higher order models, the weights are typically negligible. In addition, we add a constraint to step 4 in Algorithm 1, which requires that w_j^{loo} are non increasing with j. This constraint is quite natural in practice, as $r^{(j)}$ should be more informative about the winning algorithm than $r^{(j+1)}$.

Finally, we note that a similar loo approach applies to any divergence measure of interest. For example, the empirical counter-part of the TV is known to be $R^{TV} = \frac{1}{n}\sum_{i=1}^n\sum_{j=1}^m |\mathbb{I}(y_i = j) - \hat{p}_j|$. Hence, the corresponding loo scheme is given by

$$R_{loo}^{TV} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}|\mathbb{I}(y_i = j) - \hat{p}_j^w[-i]|, \tag{8}$$

which is again convex in $w \in \mathcal{W}$ and holds all the favorable properties of (7). We demonstrate, compare and discuss both the proposed estimators in the following section.

6 Experiments

Let us now illustrate the performance of our suggested scheme in synthetic and real-world experiments.

6.1 Synthetic Experiments

First, we study six example distributions, which are common benchmarks for probability estimation and related problems (Orlitsky & Suresh, 2015). The Zipf's law distribution is a typical benchmark in large alphabet probability estimation; it is a commonly used heavy-tailed distribution, mostly for modeling natural (real-world) quantities in physical and social sciences, linguistics, economics and others fields (Saichev et al., 2009). The Zipf's law distribution follows $P(u; s, m) = u^{-s}/\sum_{v=1}^{m} v^{-s}$ where s is a skewness parameter. Additional examples of commonly used heavy-tailed distributions are the geometric distribution, $P(u; \alpha) = (1-\alpha)^{u-1}\alpha$, the negative-binomial distribution (specifically, see (Efron & Thisted, 1976)), $P(u; l, r) = \binom{u+l-1}{u} r^u (1-r)^l$ and the beta-binomial distribution $P(u; m, \alpha, \beta) = \binom{m}{u} B(u+\alpha, m-u+\beta)/B(\alpha, \beta)$. Notice that the support of the geometric and the negative-binomial distributions is infinite. Therefore, for the purpose of our experiments, we truncate them to an alphabet size m and normalize accordingly. Additional example distributions are the uniform, $p(u) = 1/m$, and the step distribution, $P(u) \propto 1$ for $u \leq m/2$ and $P(u) \propto 1/2$ otherwise. In our context, the heavy-tailed distributions correspond to a case where a small number of algorithms outperform the others, while the more uniform setups correspond to the case where all algorithms are more equally ranked.

In our first experiment we study $m = 6$ competing algorithms. Therefore, we have $m! = 720$ possible rankings and P is the probability of each of these rankings. The ordering of the alphabet (that is, which ranking corresponds to which probability) is assigned at random. We draw n samples from P. Notice that the ith sample corresponds to a complete ranking of the m algorithms on i^{th} dataset. Given the n samples, we estimate p and evaluate the divergence of interest (1). We repeat this process 1000 times (that is, draw n samples, compute $\hat{p}$ and evaluate $D_{\mathrm{KL}}(p||\hat{p})$, $D_{\mathrm{TV}}(p, \hat{p})$) to attain the averaged divergence. We compare the following estimators. First, we compute the MLE as a benchmark method. Then, we evaluate our proposed data-dependent scheme (Algorithm 1) with only three weights w_1, w_2, w_3, as discussed above. We further compute the data-independent scheme where relevant. As a lower bound, we evaluate an Oracle estimator who knows the true p and computes the optimal weights by minimizing the true expected divergence. Last, we consider an additional alternative, as we directly estimate P with Good-Turing large alphabet estimator (Gale & Sampson, 1995; Painsky, 2022), and obtain a corresponding estimation for p. Figure 2 demonstrates the results we achieve under KL divergence, for the six distributions above, as n increases.

As we can see, our proposed scheme (in blue) outperforms the MLE (in red) quite significantly. This is especially evident with the uniform and step distributions, where it is only natural to put more weight on $r^{(2)}$ and $r^{(3)}$, as opposed to completely ignoring them. Further, our proposed scheme is relatively close to the Oracle, where the gap closes as n increases. This is not quite surprisingly, as Algorithm 1 is a leave-one-out, which performs better as n increases. Finally, notice that all three methods converge to the same performance for sufficiently large n, as expected. We omit the GT approach from Figure 2 as it is not competitive for with the alternatives. Let us further examine the results of Algorithm 1. Figure 3 presents the average weights of Algorithm 1, compared with the Oracle weights. We focus our attention to the Zipf's Law and the uniform distributions for brevity. First, consider the Zipf's Law experiment (left). We notice that w_1^{loo} tends to 1 as n increases. On the other hand, the weights are much more uniform (and steady) with the uniform distribution (right). Again, this is quite obvious. The Zipf's Law distribution is heavy-tailed, which means that a few algorithms are dominant, This means we put more weight of the winner. On the other hand, the uniform distribution suggests that all algorithms are equally likely to win, which means that $r^{(2)}$ and $r^{(3)}$ are not negligible, compared to $r^{(1)}$. Our proposed scheme successfully adapts this behavior, given only the data in hand.

Next, we repeat the same experiment under the TV distance. Here, we focus our attention to a very small number of datasets (small n), which is also quite common in the literature. We illustrate our minimax approach (Section 4), which is better suited for such limited-sample setups than the leave-one-out method. Figure 4 demonstrates the results we achieve for Theorem 1 and Theorem 2 with $K = 3$. Once again, we

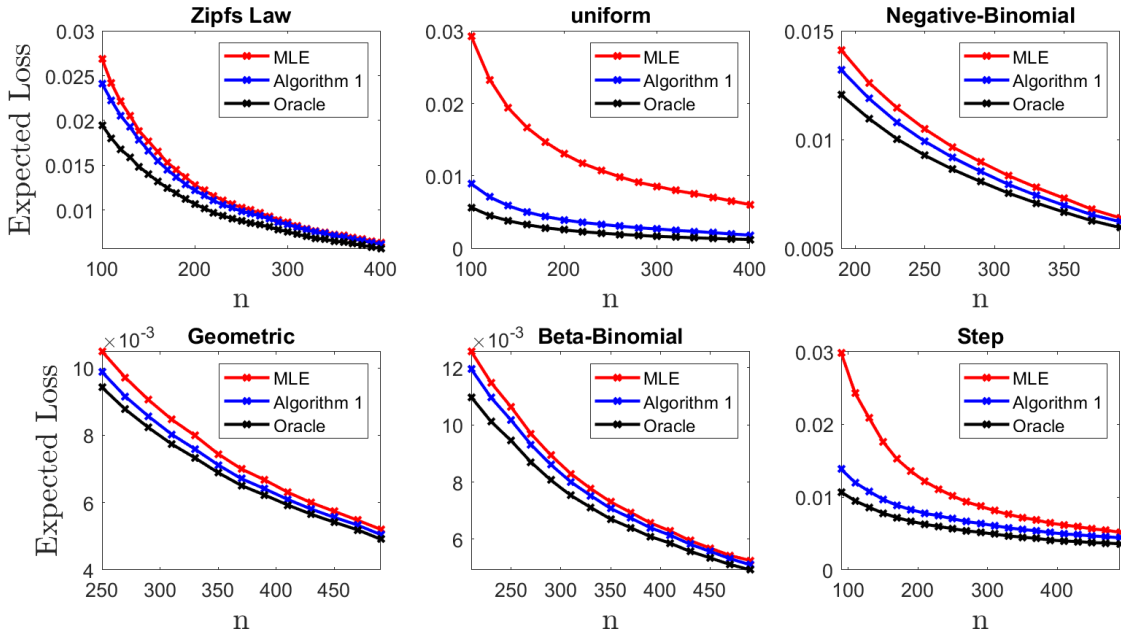

Figure 2: Comparing Algorithm 1 (blue) with the MLE (red) and an Oracle estimator (black) in two synthetic experiments. We use the following parameters: Zipf's Law: $s = 1.01$, Geometric: $\alpha = 0.4$, Negative-Binomial: $l = 1, r = 0.003$, Beta-Binomial: $\alpha = \beta = 2$.

compare our results to an Oracle who knows P but restricted to a $K = 2$ model. The observed trends closely resemble those from the previous experiment: our proposed estimators consistently outperform alternative approaches (with GT omitted again due to poor performance). Notably, both minimax-based methods significantly outperform the MLE, and we observe a notable improvement when extending the model from $K = 2$ to $K = 3$, following Theorem 2.

6.2 Real-world Experiments

We now proceed to real-world experiments. Here, we consider large comparative studies, where multiple key ML algorithms are compared against each other over a benchmark of datasets. First, we revisit the work of McElfresh et al. (2024). In this work, the authors introduce a benchmark of 176 tabular datasets, and compare a total of 19 popular ML algorithms. Their goal is to study the performance of gradient boosted decision trees (GBDT) compared to neural networks (NN), over tabular data. They conclude that "for a surprisingly high number of datasets, either the performance difference between GBDTs and NNs is negligible, or light hyper-parameter tuning on a GBDT is more important than choosing between NNs and GBDTs". In addition, they introduce the *TabZilla* benchmark suite, which is a collection of 36 of their "hardest" datasets. The performance of the studied algorithms over TabZilla is provided in Table 4 of McElfresh et al. (2024), where they report the rankings of the top three algorithms on each dataset. As mentioned above, the conclusions are not decisive. The top-performing algorithms (according to TabZilla) vary quite significantly, depending on the measure of interest. The PAMA criterion (which corresponds to the MLE), suggests that the top three algorithms are *CatBoost*, *XGBoost* and *LightGBM* with corresponding estimates of $0.2667, 0.2667$ and 0.2333, respectively. Notice that the averaged rankings cannot be evaluated from Table 4, as it requires the full ranking and not just the top three for each dataset. Applying our suggested data-dependent scheme to Table 4 (under KL divergence, following Algorithm 1), we obtain almost uniform w^{loo}. This leads to the top three algorithms: *XGBoost*, *CatBoost* and *LightGBM* with corresponding estimates of $0.2777, 0.2447$ and 0.1563, respectively. In other words, while the MLE suggests that only $r^{(1)}$ is an informative statistic, our proposed scheme suggests that there is also information in $r^{(2)}, r^{(3)}$. In fact, notice that while *CatBoost* and *XGBoost* attain the same number of wins, they differ quite significantly in the

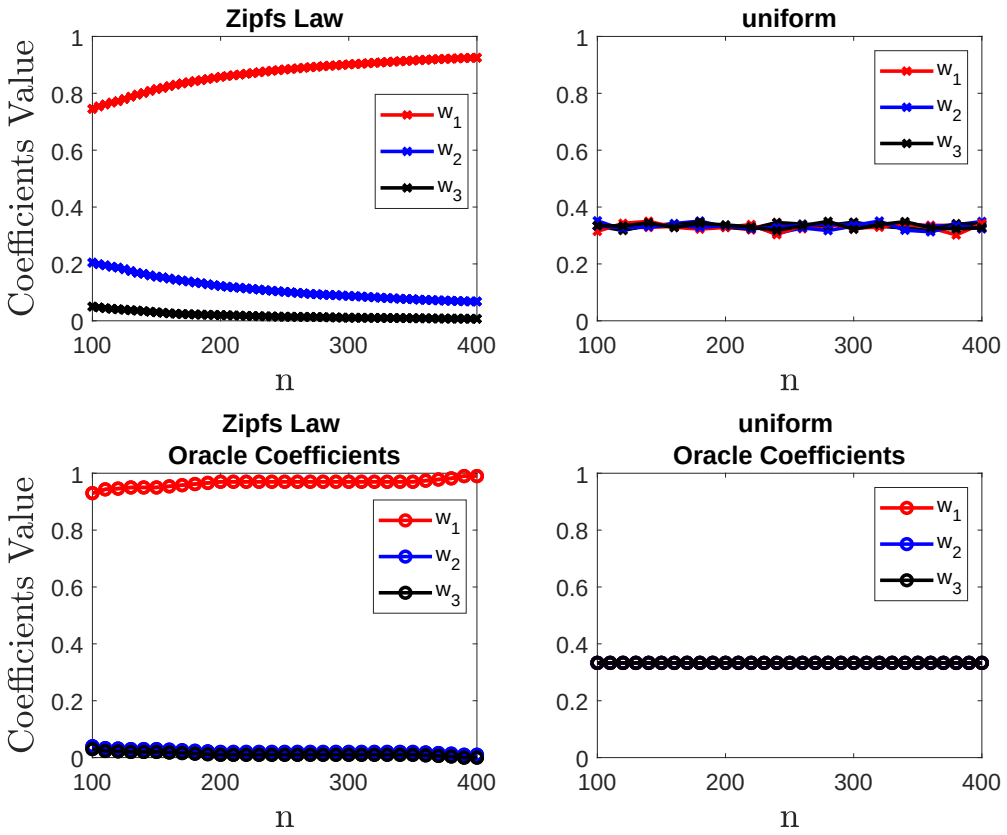

Figure 3: The obtained weights of Algorithm 1 (upper charts) and the Oracle (lower charts) in the synthetic experiment above (Figure 2)

number of times they finish second (*CatBoost* with 5 times, while *XGBoost* with 16). Our proposed scheme successfully utilizes this information and leads to a different estimate of future win probabilities. Further, we apply the Borda Count and the Bradley-Terry models (see Section 2). Interestingly, the Borda Count model leads to the following top three algorithms: *CatBoost*, *XGBoost* and *ResNet*. On the other hand, the Bradley-Terry model suggests that the best-performing algorithms are *CatBoost*, *LightGBM* and *ResNet*, where the model's scores are evaluated only according to the available top three ranks. It is important to note that the additional statistical tests discussed in Section 2 are not applicable to our comparison. Specifically, our objective is to derive a point estimate for the best-performing algorithm, while statistical tests are designed to assess whether differences in benchmark performance are statistically significant. Therefore, comparing point estimators with hypothesis tests is conceptually misleading, as they serve fundamentally different purposes.

Finally, we conduct the following experiment to validate our results. We define a subset of datasets on which we evaluate $\hat{p}$ (namely, the "train-set"). Then, we examine the estimators' performance on the remaining datasets ("test-set"). We perform k-fold splitting, and repeat the above for every split. We report the average loss (in terms of cross-entropy). Our proposed method outperforms the MLE with an average loss of 1.78 compared to 1.86, with a corresponding p-value of 0.0075. These results further support the effectiveness of our approach. The Borda Count attains an average loss of 1.91 while the Bradley-Terry model achieves an average loss of 1.97. Table 1 summarizes our results.

Next, we focus on the work Fernández-Delgado et al. (2014). In this work, the authors evaluate a total of 179 classifiers (arising from 17 basic models), implemented on Weka, R, C and Matlab. They use 121 data sets, which represent the entire UCI repository and additional real problems, in order to achieve significant conclusions about the classifiers' behavior. They conclude that the classifiers most likely to be the best are

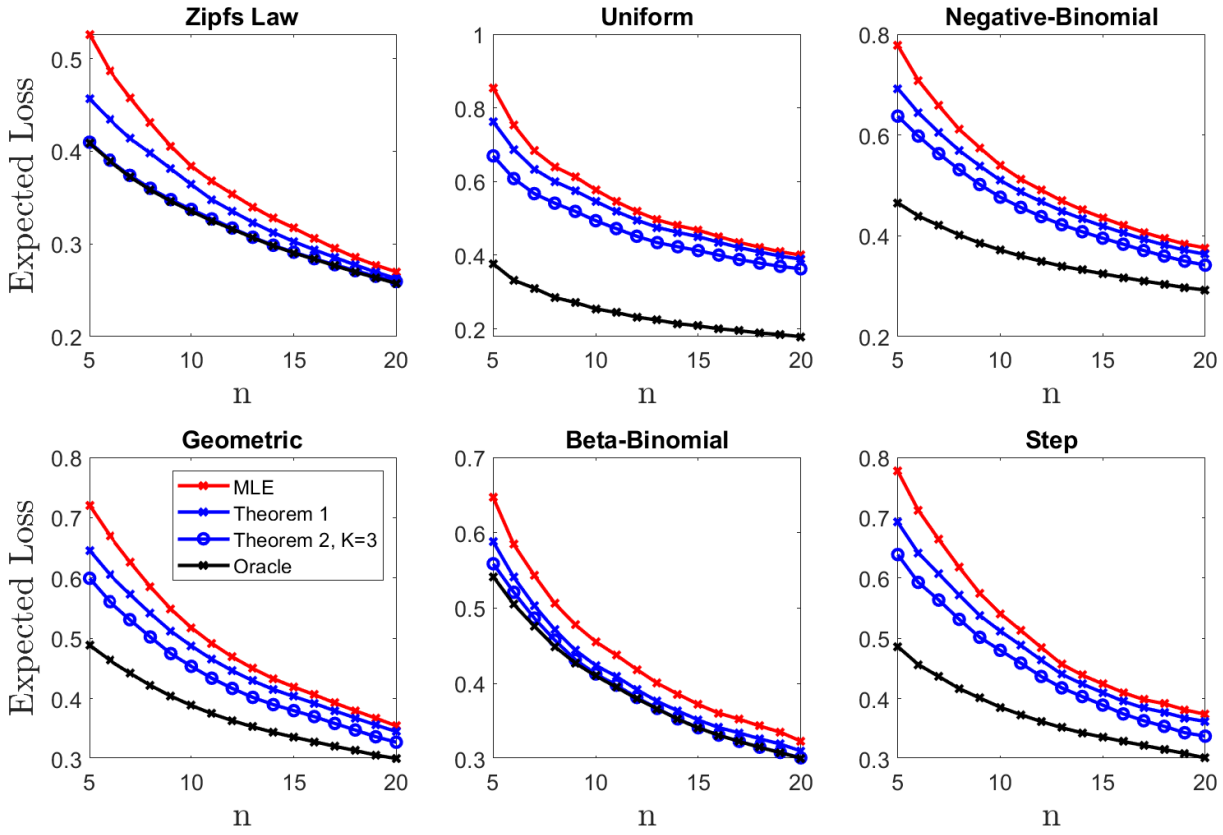

Figure 4: Comparing the data-independent schemes (Theorems 1 and 2) with the MLE (red) and an Oracle estimator (black) in small sample synthetic experiments. The distributions' parameters are described in Figure 2.

Table 1: *TabZilla* benchmark results (McElfresh et al., 2024). We report the three top-performing algorithms according to each scheme with a corresponding average cross-validation loss, as described in the text. The statistically significant best performing scheme (at level 0.05) is **bolded**.

Scheme	First	Second	Third	Average Loss
MLE	*CatBoost*	*XGBoost*	*LightGBM*	1.86
Bradley-Terry	*CatBoost*	*LightGBM*	*ResNet*	1.97
Borda Count	*CatBoost*	*XGBoost*	*ResNet*	1.91
Our proposed method	*XGBoost*	*CatBoost*	*LightGBM*	**1.78**

the random forest (RF) versions. However, the gap with the second best (SVM with a Gaussian kernel) is not statistically significant. We apply our proposed scheme to the results of Fernández-Delgado et al. (2014). First, notice that the study of Fernández-Delgado et al. (2014) compares an extremely large number of algorithms, which even exceeds the number of datasets. Furthermore, many of the algorithms are practically identical, as they only differ in the platform that they were implemented on. Hence, analyzing the winner among this set of algorithms is very noisy and prone to random results. Therefore, as a first step in reducing unnecessary comparisons, we only study one version of each algorithm. That is, we do not compare the same algorithm, implemented on two different platforms. For simplicity, let us focus on the Weka algorithms, which consist of $m = 51$. The top performing Weka algorithms according to the MLE criterion are *Rotation Forest*, *Random Committee*, *END*, *LibLINEAR* and *Decorate* with estimated probabilities of 0.1368, 0.0684, 0.0684, 0.0598 and 0.0598, respectively. Applying Algorithm 1 we conclude that the top algorithms are *Rotation Forest*, *Random Committee*, *Decorate*, *END* and *Classification Via Regression* with estimated probabilities of 0.1033, 0.0689, 0.0684, 0.0536 and 0.0534, respectively. Importantly, $w^{loo} = 0.48, 0.33, 0.19$. This suggests

that the differences between the algorithms are relatively small. Our proposed scheme identifies almost the same top-performing algorithms as the MLE. However, both the order and the estimated probabilities vary. The minimum average rank criterion suggests that the top performing algorithms are *Rotation Forest*, *Classification Via Regression*, *Random Committee*, *Decorate* and *Simple Logistic*. Notice that the minimum average rank of the best performing algorithm according to this criterion is 10.08, which only emphasizes the noisy nature of this comparison. The Borda Count scheme suggests that the best performing algorithms are *Rotation Forest*,*LibLINEAR*, *END*, *Decorate* and *Classification Via Regression*. On the other hand, the Bradley-Terry results shows that the best performing algorithm is *Rotation Forest*, followed by *Classification Via Regression*, *LibLINEAR*,*Random Committee* and *END*. Finally, to validate our results we again perform a k-fold cross validation as described above. Once again, our proposed scheme attains better results with 3.32 compared to 3.34 of the MLE. The corresponding p-value is 0.07. Naturally, the difference here is less significant due to the noisy comparison. Further, the Borda Count attains an average error of 3.46 while the Bradley-Terry model is even less competitive with an average error of 3.51. Table 2 summarizes our results.

Table 2: Fernández-Delgado et al. (2014) comparative study. We report the three top-performing algorithms according to each scheme with a corresponding average cross-validation loss, as described in the text. The statistically significant best performing scheme (at level 0.1) is **bolded**.

Scheme	**First**	**Second**	**Third**	**Average Loss**
MLE	*Rotation Forest*	*Random Committee*	*END*	3.34
Bradley-Terry	*Rotation Forest*	*Classification via Regression*	*LibLINEAR*	3.51
Borda Count	*Rotation Forest*	*LibLINEAR*	*END*	3.46
Our proposed method	*Rotation Forest*	*Random Committee*	*Decorate*	**3.32**

Finally, we turn to the comparative study of Shmuel et al. (2024). In this work, the authors introduce a comprehensive benchmark, aimed at better characterizing the types of datasets where deep learning models excel. They evaluate 111 datasets with 20 different models, including both regression and classification tasks. These datasets vary in scale and include both those with and without categorical variables. Applying the ML criterion, the top three algorithms are *AutoGluon*, *SVM* and *ResNet* with corresponding estimated probabilities of 0.3514, 0.0901 and 0.0811, respectively. Our proposed scheme suggests that the leading algorithms are *AutoGluon*, *CatBoost* and *LightGBM* with corresponding estimated probabilities of 0.2787, 0.0814 and 0.0780, respectively. Interestingly, the minimum average rank criterion agrees with our proposed scheme and recommends the same top algorithms. The Borda Count criterion leads to *AutoGluon*, *ResNet* and *LightGBM* as the top-performing models while Bradley-Terry suggests *CatBoost*, *AutoGluon* and *ResNet*. Evaluating our results using k-fold cross validation as described above, we again observe a slight improvement with an average loss of 2.42, compared to the alternative MLE with 2.45, where the corresponding p-value is 0.01. Once again, the Borda Count and the Bradley-Terry model demonstrate inferior results. The Borda Count attains an average loss of 2.59 while the Bradley-Terry model achieves an average loss of 2.72. Table 3 summarizes our results.

Table 3: Shmuel et al. (2024) comparative study. We report the three top-performing algorithms according to each scheme with a corresponding average cross-validation loss, as described in the text. The statistically significant best performing scheme (at level 0.05) is **bolded**.

Scheme	**First**	**Second**	**Third**	**Average Loss**
MLE	*AutoGluon*	*SVM*	*ResNet*	2.45
Bradley-Terry	*CatBoost*	*AutoGluon*	*ResNet*	2.72
Borda Count	*AutoGluon*	*ResNet*	*LightGBM*	2.59
Our proposed method	*AutoGluon*	*CatBoost*	*LightGBM*	**2.42**

7 Proofs

7.1 A proof for Theorem 1

First, we have

$$\mathbb{E}D_{\mathrm{TV}}\left(p,\hat{p}^{w}\right) = \mathbb{E}\sum_{j=1}^{m}|p_j-\hat{p}_j^{w}| = \mathbb{E}\sum_{j=1}^{m}\sqrt{\left(p_j-\hat{p}_j^{w}\right)^2} \leq \sum_{j=1}^{m}\sqrt{\mathbb{E}\left(p_j-\hat{p}_j^{w}\right)^2} \tag{9}$$

where the last inequality follows from Jensen inequality. Denote $q \in \Delta_m$ as a distribution vector which corresponds to the probability that an algorithm finishes second. That is q_j is the probability that the jth algorithm is the second ranked algorithm over a given dataset. From the classical variance-bias decomposition, we attain

$$\mathbb{E}\left(p_j-\hat{p}_j^{w}\right)^2 = \left(p_j-\mathbb{E}(\hat{p}_j^{w})\right)^2 + \mathrm{var}\big(\hat{p}_j^{w}\big),$$

where

$$\mathbb{E}\big(\hat{p}_j^{w}\big) = \mathbb{E}\big(wr_j^{(1)}+(1-w)r_j^{(2)}\big)/n = wp_j+(1-w)q_j$$

and

$$\begin{aligned}
&\mathrm{var}\big(\hat{p}_j^{w}\big) = \\
&\frac{w^2}{n^2}\mathrm{var}\big(r_j^{(1)}\big)+\frac{(1-w)^2}{n^2}\mathrm{var}\big(r_j^{(2)}\big)+\frac{2w(1-w)}{n^2}\mathrm{cov}\big(r_j^{(1)},r_j^{(2)}\big) = \\
&w^2\frac{p_j(1-p_j)}{n}+(1-w)^2\frac{q_j(1-q_j)}{n}-2w(1-w)\frac{p_jq_j}{n}.
\end{aligned} \tag{10}$$

Putting together the above, we obtain

$$\begin{aligned}
&\mathbb{E}\left(p_j-\hat{p}_j^{w}\right)^2 = \\
&p_j^2-2p_j(wp_j+(1-w)q_j)+(wp_j+(1-w)q_j)^2+ \\
&\frac{1}{n}\left(w^2p_j+(1-w)^2q_j-(wp_j+(1-w)q_j)^2\right) = \\
&p_j^2-2p_j(wp_j+(1-w)q_j)+\left(1-\frac{1}{n}\right)(wp_j+(1-w)q_j)^2+ \\
&\frac{1}{n}\left(w^2p_j+(1-w)^2q_j\right) < \\
&(p_j-wp_j-(1-w)q_j))^2+\frac{1}{n}\left(w^2p_j+(1-w)^2q_j\right) = \\
&(1-w)^2(p_j-q_j)^2+\frac{1}{n}\left(w^2p_j+(1-w)^2q_j\right)
\end{aligned} \tag{11}$$

where the inequality follows from $1-1/n<1$. Hence,

$$\begin{aligned}
&\mathbb{E}D_{\mathrm{TV}}\left(p,\hat{p}^{w}\right) \leq \\
&\sum_{j=1}^{m}\sqrt{(1-w)^2(p_j-q_j)^2+\frac{1}{n}\left(w^2p_j+(1-w)^2q_j\right)} \leq \\
&\max_{P\in\Delta_{m!}}\sum_{j=1}^{m}\sqrt{(1-w)^2(p_j-q_j)^2+\frac{1}{n}\left(w^2p_j+(1-w)^2q_j\right)}.
\end{aligned} \tag{12}$$

Denote $t_j = (1-w)^2(p_j-q_j)^2+\frac{1}{n}\left(w^2p_j+(1-w)^2q_j\right)$. Notice we have that

$$0 \leq t_j \leq (1-w)^2+\frac{1}{n}(w^2+(1-w)^2) \triangleq g_1(w) \tag{13}$$

$$0 \leq \sum_{j=1}^{m}t_j \leq 2(1-w)^2+\frac{1}{n}(w^2+(1-w)^2) \triangleq g_2(w) \tag{14}$$

where both upper bounds are attained for $p_u = 1$ and $q_v = 1$ for some $u \neq v$. Putting it together, we would like to bound from above $\sum_{j=1}^{n} \sqrt{t_j}$ subject to (14) and (13). This is a concave maximization problem over a convex set, and the maximum is attain for $t_j^* = g_2(w)/m$. Hence, we get that

$$\mathbb{E}D_{\mathrm{TV}}\left(p, \hat{p}^w\right) \leq \sqrt{m \cdot g_2(w)} = \sqrt{m}\sqrt{2(1-w)^2 + \frac{1}{n}(w^2 + (1-w)^2)}.$$

Notice that for $w = 1$ we obtain $\mathbb{E}D_{\mathrm{TV}}\left(p, \hat{p}^w\right) \leq \sqrt{m/n}$ as desired (4). Finally, we would like to minimize the above with respect to w. Simple derivation shows that $w^* = 1 - 1/(2n+2)$.

7.2 A proof for Theorem 2

First, we have

$$\mathbb{E}D_{\mathrm{TV}}\left(p, \hat{p}^w\right) = \mathbb{E}\sum_{j=1}^{m} |p_j - \hat{p}_j^w| = \mathbb{E}\sum_{j=1}^{m} \sqrt{\left(p_j - \hat{p}_j^w\right)^2} \leq \sum_{j=1}^{m} \sqrt{\mathbb{E}\left(p_j - \hat{p}_j^w\right)^2} \tag{15}$$

where the last inequality follows from Jensen inequality. Denote $p^{(k)} \in \Delta_m$ as a distribution vector which corresponds to the probability that an algorithm finishes k^{th} place. That is $p_j^{(k)}$ is the probability that the jth algorithm is the kth ranked algorithm over a future dataset. To avoid an overload of notation, we refer to $p^{(1)}$ as p interchangeably. From the classical variance-bias decomposition, we attain

$$\mathbb{E}\left(p_j - \hat{p}_j^w\right)^2 = \left(p_j - \mathbb{E}(\hat{p}_j^w)\right)^2 + \mathrm{var}(\hat{p}_j^w),$$

where

$$\mathbb{E}(\hat{p}_j^w) = \frac{1}{n}\mathbb{E}\left(\sum_{k=1}^{K} w_k r_j^{(k)}\right) = \sum_{k=1}^{K} w_k p_j^{(k)}$$

and

$$\begin{aligned}\mathrm{var}(\hat{p}_j^w) = &\sum_{k=1}^{K} \frac{w_k^2}{n^2}\mathrm{var}(r_j^{(k)}) + \sum_{u \neq v} \frac{2w_u w_v}{n^2}\mathrm{cov}(r_j^{(u)}, r_j^{(v)}) = \\ &\sum_{k=1}^{K} w_k^2 \frac{p_j^{(k)}(1-p_j^{(k)})}{n} - 2\sum_{u \neq v} w_u w_v \frac{p_j^{(u)} p_j^{(v)}}{n}.\end{aligned} \tag{16}$$

Putting together the above, we obtain

$$\begin{aligned}\mathbb{E}\left(p_j - \hat{p}_j^w\right)^2 = &\left((1-w_1)p_j - \sum_{k=2}^{K} w_k p_j^{(k)}\right)^2 + \\ &\frac{1}{n}\left(\sum_{k=1}^{K} w_k^2 p_j^{(k)}(1-p_j^{(k)}) - 2\sum_{u \neq v} w_u w_v p_j^{(u)} p_j^{(v)}\right) \leq \\ &\left((1-w_1)p_j - \sum_{k=2}^{K} w_k p_j^{(k)}\right)^2 + \frac{1}{n}\sum_{k=1}^{K} w_k^2 p_j^{(k)}.\end{aligned} \tag{17}$$

Hence,

$$\mathbb{E}D_{\mathrm{TV}}\left(p, \hat{p}^w\right) \leq \sum_{j=1}^{m} \sqrt{\left((1-w_1)p_j - \sum_{k=2}^{K} w_k p_j^{(k)}\right)^2 + \frac{1}{n}\sum_{k=1}^{K} w_k^2 p_j^{(k)}}. \tag{18}$$

We now bound (18) from above with respect to $p^{(1)}, ..., p^{(k)}$. Let $\mathcal{U}_m$ be a doubly stochastic matrix such that for every $U \in \mathcal{U}_m$ we have $\sum_{i=1}^{m} U_{ij} = \sum_{j=1}^{m} U_{ij} = 1$ and $U_{ij} \geq 0$. We use $U \in \mathcal{U}_m$ to represent $p^{(1)}, ..., p^{(m)}$.

Specifically, recall that $p_j^{(k)}$ is the probability that the jth algorithms is the kth ranked algorithm. This means that $\sum_{k=1}^m p_j^{(k)} = 1$ for every $j = 1, ..., m$ and $\sum_{j=1}^m p_j^{(k)} = 1$ for every $k = 1, ..., m$. Therefore, we represent $p^{(1)}, ..., p^{(m)}$ with U, such that the kth column of U corresponds to $p^{(k)}$. That is, $U = [p^{(1)}, ..., p^{(m)}]$. Yet, our optimization problem is only with respect to $p^{(1)}, ..., p^{(K)}$. Hence, we define a collection $\mathcal{V}_{K,m}$ is the following manner. Let $U \in \mathcal{U}_m$ be a doubly stochastic matrix. Define V as a subset of K columns of U. Let $\mathcal{V}_{K,m}$ be the collection of all possible V matrices. Naturally, for every $V \in \mathcal{V}_{K,m}$ we have $\sum_{i=1}^m V_{ij} = 1$ for every $j = 1, ..., K$ but $\sum_{j=1}^K V_{ij} \leq 1$ for every $i = 1, ..., m$. Therefore, we represent $p^{(1)}, ..., p^{(K)}$ as the columns of V such that $V = [p^{(1)}, ..., p^{(K)}] \in \mathcal{V}_{K,m}$ and obtain

$$\mathbb{E} D_{\mathrm{TV}}\left(p, \hat{p}^w\right) \leq \max_{p^{(1)},...,p^{(K)} \in \mathcal{V}_{K,m}} \sum_{j=1}^m f\left(p_j^{(1)}, ..., p_j^{(k)}\right). \tag{19}$$

where

$$f\left(p_j^{(1)}, ..., p_j^{(k)}\right) = \sqrt{\left((1-w_1)p_j - \sum_{k=2}^K w_k p_j^{(k)}\right)^2 + \frac{1}{n}\sum_{k=1}^K w_k^2 p_j^{(k)}}.$$

Let us now relax the maximization above (19) by considering $p^{(1)}$ and $p^{(2)}, ..., p^{(K)}$ separately. That is,

$$\max_{p^{(1)},...,p^{(K)} \in \mathcal{V}_{K,m}} \sum_{j=1}^m f\left(p_j^{(1)}, ..., p_j^{(k)}\right) \leq \max_{p^{(1)} \in \Delta_m} \left(\max_{p^{(2)}...p^{(K)} \in \mathcal{V}_{K-1,m}} \sum_{j=1}^m f\left(p_j^{(1)}, ..., p_j^{(K)}\right) \right)$$

In words, we maximize the objective over a marginal $p^{(1)}$ which does not depend on the choice of the remaining distributions. Let us first consider the maximization over $p^{(2)}, ..., p^{(K)}$. Proposition 1 below shows that the maximum of $\sum_j f\left(p_j^{(1)}, ..., p_j^{(K)}\right)$ over $p^{(2)}, ..., p^{(K)} \in \mathcal{V}_{K-1,m}$ is attained on the vertices of the set. That is, for degenerate distributions $p^{(k)}$, $k = 2, ..., K$.

Formally, define $\mathcal{T}_m$ as a collection of permutation matrices. That is, every $T \in \mathcal{T}_m$ is a doubly stochastic matrix and $T_{ij} \in \{0, 1\}$. Next, define S as a *partial permutation matrix* of K columns. That is, for every $T \in \mathcal{T}_m$ we define S as a subset of K columns of T. Let $\mathcal{S}_{K,m}$ be the collection of all partial permutation matrices S. For example, assume $m = 3$ and $K = 2$. Then, $\mathcal{T}_m$ is the collection

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

and $\mathcal{S}_{K,m}$ is simply

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Notice that $|\mathcal{S}_{K,m}| = |\mathcal{T}_m|$. Further, every $S \in \mathcal{S}_{K,m}$ satisfies the following:

- $\sum_i S_{ij} = 1$ for every $j = 1, ..., m$.
- $\sum_j S_{ij} \in \{0, 1\}$ for every $i = 1, ..., K$.
- $S_{ij} \in \{0, 1\}$ for every $j = 1, ..., m$ and $i = 1, ..., K$.

In words, every columns in $S \in S_{K,m}$ corresponds to a degenerate distribution and no two columns are identical.

Proposition 1. *Let $S_{K-1,m}$ be a collection of partial permutation matrices. Assume that $w_2 \geq w_3 \geq ... \geq w_K$. Then,*

$$\max_{p^{(2)}...p^{(K)} \in \mathcal{V}_{K-1,m}} \sum_{j=1}^{m} f\left(p_j^{(1)}, ..., p_j^{(K)}\right) \leq \max_{p^{(2)}...p^{(K)} \in \mathcal{S}_{K-1,m}} \sum_{j=1}^{m} g\left(p_j^{(1)}, ..., p_j^{(K)}\right)$$

where

$$g\left(p_j^{(1)}, ..., p_j^{(K)}\right) = \sqrt{\left((1-w_1)p_j^{(1)} - \sum_{k=2}^{K} w_k p_j^{(k)}\right)^2 + \frac{1}{n}\left(w_1^2 p_j^{(1)} + \frac{1}{n} w_2^2\right)}$$

Proof. First, we have that

$$\begin{aligned} f\left(p_j^{(1)}, ..., p_j^{(K)}\right) = & \sqrt{\left((1-w_1)p_j^{(1)} - \sum_{k=2}^{K} w_k p_j^{(k)}\right)^2 + \frac{1}{n}\sum_{k=1}^{K} w_k^2 p_j^{(k)}} \leq \\ & \sqrt{\left((1-w_1)p_j^{(1)} - \sum_{k=2}^{K} w_k p_j^{(k)}\right)^2 + \frac{1}{n}\left(w_1^2 p_j^{(1)} + w_2^2\right)} \triangleq \\ & g\left(p_j^{(1)}, ..., p_j^{(K)}\right). \end{aligned} \tag{20}$$

where the inequality is due to

$$\sum_{k=1}^{K} w_k^2 p_j^{(k)} = w_1^2 p_j^{(1)} + \sum_{k=2}^{K} w_k^2 p_j^{(k)} \leq w_1^2 p_j^{(1)} + \max_{k=2,...,K} w_k^2 = w_1^2 p_j^{(1)} + w_2^2$$

as $w_2 \geq w_3 \geq ... \geq w_K$ and $\sum_{k=2}^{K} p_j^{(k)} \leq 1$. Next, it is immediate to show that the Hessian of $g\left(p_j^{(1)}, ..., p_j^{(K)}\right)$ with respect to $p_j^{(2)}, ..., p_j^{(K)}$ is positive semi-definite. This means that $g\left(p_j^{(1)}, ..., p_j^{(K)}\right)$ is convex with respect to $p_j^{(2)}, ..., p_j^{(K)}$. Now, define $\tilde{V} = [p^{(2)}, ..., p^{(K)}]$ and $G(p^{(1)}, \tilde{V}) = \sum_{j=1}^{n} g\left(p_j^{(1)}, ..., p_j^{(K)}\right)$. Then, $G(p^{(1)}, \tilde{V})$ is convex in $\tilde{V}$, as a sum of convex functions. Formally, let $\tilde{V}_1 = [p^{(2)}, ..., p^{(K)}]$ and $\tilde{V}_2 = [q^{(2)}, ..., q^{(K)}]$ such that $\tilde{V}_1, \tilde{V}_2 \in \mathcal{V}_{K-1,m}$. Then, for every $\lambda \in [0, 1]$,

$$\begin{aligned} & G(p^{(1)}, \lambda\tilde{V}_1 + (1-\lambda)\tilde{V}_2) = \\ & \sum_{j=1}^{m} \sqrt{\left((1-w_1)p_j^{(1)} - \sum_{k=2}^{K} w_k \left(\lambda p_j^{(k)} + (1-\lambda) q_j^{(k)}\right)\right)^2 + \frac{1}{n}\left(w_1^2 p_j^{(1)} + w_2^2\right)} \leq \\ & \lambda \sum_{j=1}^{m} \sqrt{\left((1-w_1)p_j^{(1)} - \sum_{k=2}^{K} w_k p_j^{(k)}\right)^2 + \frac{1}{n}\left(w_1^2 p_j^{(1)} + w_2^2\right)} + \\ & (1-\lambda) \sum_{j=1}^{m} \sqrt{\left((1-w_1)p_j^{(1)} - \sum_{k=2}^{K} w_k q_j^{(k)}\right)^2 + \frac{1}{n}\left(w_1^2 p_j^{(1)} + w_2^2\right)} = \\ & \lambda G(p^{(1)}, \tilde{V}_1) + (1-\lambda) G(p^{(1)}, \tilde{V}_2) \end{aligned} \tag{21}$$

where the inequality follows from the convexity of $g\left(p_j^{(2)}, ..., p_j^{(K)}\right)$. Finally, Birkhoff–von Neumann theorem states that every doubly stochastic real matrix $U \in \mathcal{U}_m$ is a convex combination of permutation matrices, $T \in \mathcal{T}_m$ with the permutation matrices T being precisely the extreme points (the vertices) of the Birkhoff polytope $\mathcal{T}_m$ (Jurkat & Ryser, 1967). In other words, every doubly stochastic matrix $U \in \mathcal{U}_m$ may be represented as a convex combination of the permutation matrices $T \in \mathcal{T}_m$. This means that every $\tilde{V} \in \mathcal{V}_{K-1,m}$ is also a convex combination of partial permutation matrices $S \in \mathcal{S}_{K-1,m}$, following the construction of $\tilde{V}$.

That is, assume that $\tilde{V}$ is defined by $K-1$ columns of a doubly stochastic matrix U. By the Birkhoff–von Neumann theorem, $U = \sum_i \lambda_i T_i$ is a convex combination of $T_i \in \mathcal{T}_m$. This means that $\tilde{V} = \sum_i \lambda_i S_i$ is a convex combination of the corresponding $S_i \in \mathcal{S}_{K-1,m}$. By the convexity of $G(p^{(1)}, \tilde{V})$ with respect to $\tilde{V}$ we attain

$$\max_{p^{(2)}...p^{(K)} \in \mathcal{V}_{K-1,m}} \sum_{j=1}^{m} f\left(p_j^{(1)}, ..., p_j^{(K)}\right) \leq \max_{p^{(2)}...p^{(K)} \in \mathcal{S}_{K-1,m}} \sum_{j=1}^{m} g\left(p_j^{(1)}, ..., p_j^{(K)}\right)$$

□

Plugging Proposition 1 to (19) yields

$$\mathbb{E} D_{\mathrm{TV}}\left(p, \hat{p}^w\right) \leq \max_{p^{(1)} \in \Delta_m} \left(\max_{p^{(2)}...p^{(K)} \in \mathcal{S}_{K-1,m}} \sum_{j=1}^{m} g\left(p_j^{(1)}, ..., p_j^{(K)}\right) \right) \tag{22}$$

Let us now focus on the maximization with respect to $p^{(1)}$. Proposition 2 below shows that under the stated condition, $\sum_{j=1}^{m} g\left(p_j^{(1)}, ..., p_j^{(K)}\right)$ is concave in $p^{(1)}$ for every $p^{(2)}, ..., p^{(K)} \in \mathcal{S}_{K-1,m}$.

Proposition 2. *Assume* $w_1 \geq \frac{8n-\sqrt{8n}}{8n-1}$. *Then,* $\sum_{j=1}^{m} g\left(p_j^{(1)}, ..., p_j^{(K)}\right)$ *is concave in* $p^{(1)}$ *for every* $p^{(2)}, ..., p^{(K)} \in \mathcal{S}_{K-1,m}$.

Proof. Define $z(t) = \sqrt{(at-b)^2 + ct + d}$. The function $z(t)$ is concave in t if $d^2 z(t)/dt^2 \leq 0$. Hence, we require that

$$2a^2\left((at-b)^2 + ct + d\right) - \frac{1}{2}(2a(at-b)+c)^2 \leq 0,$$

which holds if and only if $2abc + 2a^2 d - c^2/2 \leq 0$. Considering

$$g\left(p_j^{(1)}, ..., p_j^{(K)}\right) = \sqrt{\left((1-w_1)p_j^{(1)} - \sum_{k=2}^{K} w_k p_j^{(k)}\right)^2 + \frac{1}{n}\left(w_1^2 p_j^{(1)} + w_2^2\right)}$$

with respect to $p_j^{(1)}$ we have that $a = (1-w_1)$, $b = \sum_{k=2}^{K} w_k p_j^{(k)}$, $c = w_1^2/n$ and $d = w_2^2/n$. Therefore, we require

$$2(1-w_1)\frac{w_1^2}{n}\sum_{k=2}^{K} w_k p_j^{(k)} + 2(1-w_1)^2\frac{w_2^2}{n} - \frac{w_1^4}{2n^2} \leq 0 \tag{23}$$

for the desired concavity of $\sum_{j=1}^{m} g\left(p_j^{(1)}, ..., p_j^{(K)}\right)$. Notice that $w_k \leq w_2 \leq 1-w_1$ for every $k = 2, ..., K$, where the last inequality follows from $\sum_{k=1}^{K} w_k = 1$. Further, $\sum_{k=2}^{K} w_k p_j^{(k)} \leq w_2$ since every $p^{(2)}, ..., p^{(K)} \in S_{K-1,m}$ satisfies $p_j^{(k)} \in \{0,1\}$ and $\sum_{k=2}^{K} p_j^{(k)} \leq 1$. Therefore, it is enough to show that

$$\frac{2(1-w_1)^2 w_1^2}{n} + \frac{2(1-w_1)^4}{n} - \frac{w_1^4}{2n^2} \leq \frac{4(1-w_1)^2 w_1^2}{n} - \frac{w_1^4}{2n^2} \leq 0. \tag{24}$$

Solving the above quadratic inequality yields $w_1 \geq \frac{8n-\sqrt{8n}}{8n-1}$.

□

Let us now apply Proposition 2 to (22). Denote

$$q^{(2)}, ..., q^{(K)} = \operatorname*{argmax}_{p^{(2)}...p^{(K)} \in \mathcal{S}_{K-1,m}} \sum_{j=1}^{m} g\left(p_j^{(1)}, ..., p_j^{(K)}\right). \tag{25}$$

Recall that $q_j^{(k)} \in \{0,1\}$ for all $j = 1,...,m$ and $k = 2,..,K$. In addition, $\sum_{k=2}^{K} q_j^{(k)} \in \{0,1\}$ for every $j = 1,...,m$ and $\sum_{j=1}^{m} q_j^{(k)} = 1$ for every $k = 2,..,K$. Denote by $\mathcal{J}$ the collection of j indices for which $\sum_{k=2}^{K} q_j^{(k)} = 1$. Notice that only a single element in this sum equals one, while the others are zero. We denote the index of this element by k_j. That is, for every $j \in \mathcal{J}$, we have $q_j^{(k_j)} = 1$ while $q_j^{(k)} = 0$ for every $k \neq k_j$. Therefore,

$$\sum_{j=1}^{m} g\left(p_j^{(1)}, q_j^{(2)}, ..., q_j^{(K)}\right) = \sum_{j\in\mathcal{J}} g\left(p_j^{(1)}, q_j^{(2)} ..., q_j^{(K)}\right) + \sum_{j\notin\mathcal{J}} g\left(p_j^{(1)}, q_j^{(2)}, ..., q_j^{(K)}\right) = \sum_{j\in\mathcal{J}} \sqrt{\left((1-w_1)p_j^{(1)} - w_{k_j}\right)^2 + \frac{1}{n}\left(w_1^2 p_j^{(1)} + w_2^2\right)} + \sum_{j\notin\mathcal{J}} \sqrt{(1-w_1)^2\left(p_j^{(1)}\right)^2 + \frac{1}{n}\left(w_1^2 p_j^{(1)} + w_2^2\right)}.$$

Let us first focus on the second summation. Notice that all summands are identical functions of $p_j^{(1)}$. In addition, the sum is concave with respect to $p_j^{(1)}$, as suggested in Proposition 2. Denote $\pi = \sum_{j\notin\mathcal{J}} p_j^{(1)}$. By the concavity and the symmetry of the second summand, we have that

$$\sum_{j\notin\mathcal{J}} \sqrt{(1-w_1)^2\left(p_j^{(1)}\right)^2 + \frac{1}{n}\left(w_1^2 p_j^{(1)} + w_2^2\right)} \leq \sum_{j\notin\mathcal{J}} \sqrt{(1-w_1)^2\left(\frac{\pi}{m-|\mathcal{J}|}\right)^2 + \frac{1}{n}\left(w_1^2 \frac{\pi}{m-|\mathcal{J}|} + w_2^2\right)} = \sqrt{(1-w_1)^2\pi^2 + \frac{(m-|\mathcal{J}|)}{n}\left(w_1^2\pi + w_2^2(m-|\mathcal{J}|)\right)}. \tag{26}$$

That is, a uniform distribution (over a total mass of π) maximizes the summation above. For simplicity of presentation (and without loss of generality), we denote $j \in \mathcal{J}$ as $j = 1,...,J$ where $J = |\mathcal{J}|$. Notice that from maximization considerations, $J = K-1$. That is, maximizing $\sum_{j=1}^{m} g\left(p_j^{(1)}, p_j^{(2)}, ..., p_j^{(K)}\right)$ over $\tilde{V} = [p^{(2)}, ..., p^{(K)}] \in \mathcal{S}_{K-1,m}$ necessarily results in a maximal possible number of rows in $\tilde{V}$ for which the sum of the entire row is one. Hence, we have that

$$\max_{p_1^{(1)},...,p_{K-1}^{(1)}} \sum_{j=1}^{m} g\left(p_j^{(1)}, q_j^{(2)}, ..., q_j^{(K)}\right) = \max_{p_1^{(1)},...,p_{K-1}^{(1)}} \sum_{j=1}^{K-1} \sqrt{\left((1-w_1)p_j^{(1)} - w_{k_j}\right)^2 + \frac{1}{n}\left(w_1^2 p_j^{(1)} + w_2^2\right)} + \sqrt{(1-w_1)^2\pi^2 + \frac{(m-K+1)}{n}\left(w_1^2\pi + w_2^2(m-K+1)\right)} \tag{27}$$

where $\pi = 1 - \sum_{j'=1}^{K-1} p_{j'}^{(1)}$. Importantly, (27) is a concave maximization problem over a simplex of K variables, $p_1^{(1)}, ..., p_{K-1}^{(1)}, \pi$, due to Proposition 2.

Finally, notice that the (27) depends on $q^{(2)}, ..., q^{(K)} \in \mathcal{S}_{K-1,m}$ only through w_{k_j}. That is, every choice of $p^{(2)}, ..., p^{(K)} \in \mathcal{S}_{K-1,m}$ corresponds to a different set of w_{k_j}. However, $p^{(2)}, ..., p^{(K)} \in \mathcal{S}_{K-1,m}$ are degenerate and unique distributions. This means that w_{k_j} are unique values from the given $w_2, ..., w_K$. In other words, the choice of $p^{(2)}, ..., p^{(K)} \in \mathcal{S}_{K-1,m}$ only dictates a bijection between $\{w_{k_j}\}_{j=1}^{K-1}$ and $\{w_j\}_{j=2}^{K}$. Therefore, every choice $p^{(2)}, ..., p^{(K)} \in \mathcal{S}_{K-1,m}$ leads to exactly the same maximization problem over $p_1^{(1)}, ..., p_{K-1}^{(1)}$. For simplicity, and without loss of generality, we choose $w_{k_j} = w_{j+1}$ for $j = 1, ..., K-1$ which concludes the proof.

8 Conclusions

Given a set of algorithms and a benchmark of datasets, it is not entirely clear how to identify the best-performing algorithm or even how to establish a meaningful ranking among them. In this work we claim that a natural evaluation criterion is the probability that an algorithm will win (rank highest) on a future, unseen, dataset. We introduce a rank-based framework for estimating this win probability, which generalizes the ML approach. Specifically, instead of simply counting the number of past wins, we also account for the number of times each algorithm finished second, third and so forth. This way, we provide a "natural" weighting for the performance of the algorithms on the benchmark set. Our goal is not to demonstrate the superiority of a particular positional scoring rule, but rather to study weight selection when the target quantity is future win probability. We present two evaluation schemes within this framework. The first involves data-independent weights, predetermined before the evaluation begins, which are optimized to minimize worst-case risk. The second is a more adaptive, data-dependent approach based on a leave-one-out strategy, which shows significant improvements over existing alternatives. We validate both schemes through synthetic and real-world experiments.

Our analysis yields several important insights for comparative studies. First, introducing too many competing algorithms is counter-productive. Specifically, by doing so we inherently add unnecessary noise to the rankings, which makes it more difficult to conclude on the winner. We advocate for the exclusion of algorithms that are not serious contenders, as their inclusion merely inflates the scope without improving the quality of the evaluation. Second, we argue that focusing solely on winners is insufficient. Lower-ranked placements contain valuable information, and relying on the MLE is suboptimal. However, our method converges to the MLE when the number of datasets is large or when there are highly dominant algorithms, in which case win-counting suffices.

Our proposed framework is not limited to algorithm comparison. It generalizes to any scenario involving repeated competitions among multiple entities. For instance, consider a horse race with m horses, where past performance data is available and spectators aim to predict the most likely winner of a future race. Here, the horses correspond to the algorithms and their performance in past races are their benchmark results. Naturally, the goal is to determine the horse that is most likely to win a future race. We consider such applications and their natural adaptions for our future work.

References

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

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006.

Olive Jean Dunn. Multiple comparisons among means. *Journal of the American statistical association*, 56 (293):52–64, 1961.

Bradley Efron and Ronald Thisted. Estimating the number of unseen species: How many words did Shakespeare know? *Biometrika*, 63(3):435–447, 1976.

Peter Emerson. The original borda count and partial voting. *Social Choice and Welfare*, 40(2):353–358, 2013.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The journal of machine learning research*, 15(1): 3133–3181, 2014.

Ronald A Fisher. Statistical methods and scientific inference. 1956.

Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701, 1937.

Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The annals of mathematical statistics*, 11(1):86–92, 1940.

William A Gale and Geoffrey Sampson. Good-Turing frequency estimation without tears. *Journal of quantitative linguistics*, 2(3):217–237, 1995.

Irving J Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, 1953.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pp. 65–70, 1979.

Gerhard Hommel. A stagewise rejective multiple test procedure based on a modified bonferroni test. *Biometrika*, 75(2):383–386, 1988.

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

Jiantao Jiao, Kartik Venkat, Yanjun Han, and Tsachy Weissman. Minimax estimation of functionals of discrete distributions. *IEEE Transactions on Information Theory*, 61(5):2835–2885, 2015.

Wolfgang B Jurkat and Herbert John Ryser. Term ranks and permanents of nonnegative matrices. *Journal of algebra*, 5(3):342–357, 1967.

David A McAllester and Robert E Schapire. On the convergence rate of Good-Turing estimators. In *COLT*, pp. 1–6, 2000.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36, 2024.

Peter Bjorn Nemenyi. *Distribution-free multiple comparisons.* Princeton University, 1963.

Alon Orlitsky and Ananda Theertha Suresh. Competitive distribution estimation: Why is Good-Turing good. In *Advances in Neural Information Processing Systems*, pp. 2143–2151, 2015.

Amichai Painsky. Convergence guarantees for the good-turing estimator. *Journal of Machine Learning Research*, 23(279):1–37, 2022.

Amichai Painsky and Gregory Wornell. On the universality of the logistic loss function. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 936–940. IEEE, 2018.

Alexander I Saichev, Yannick Malevergne, and Didier Sornette. *Theory of Zipf's law and beyond*, volume 632. Springer Science & Business Media, 2009.

Assaf Shmuel, Oren Glickman, and Teddy Lazebnik. A comprehensive benchmark of machine and deep learning across diverse tabular datasets. *arXiv preprint arXiv:2408.14817*, 2024.

John W Tukey. Comparing individual means in the analysis of variance. *Biometrics*, pp. 99–114, 1949.

Han-Jia Ye, Si-Yang Liu, Hao-Run Cai, Qi-Le Zhou, and De-Chuan Zhan. A closer look at deep learning on tabular data. *arXiv preprint arXiv:2407.00956*, 2024.