Label Ranking through Nonparametric Regression

Dimitris Fotakis[1] **Alkis Kalavasis**[1] **Eleni Psaroudaki**[1]

Abstract

Label Ranking (LR) corresponds to the problem of learning a hypothesis that maps features to rankings over a finite set of labels. We adopt a nonparametric regression approach to LR and obtain theoretical performance guarantees for this fundamental practical problem. We introduce a generative model for Label Ranking, in noiseless and noisy nonparametric regression settings, and provide sample complexity bounds for learning algorithms in both cases. In the noiseless setting, we study the LR problem with full rankings and provide computationally efficient algorithms using decision trees and random forests in the high-dimensional regime. In the noisy setting, we consider the more general cases of LR with incomplete and partial rankings from a statistical viewpoint and obtain sample complexity bounds using the One-Versus-One approach of multiclass classification. Finally, we complement our theoretical contributions with experiments, aiming to understand how the input regression noise affects the observed output.

1. Introduction

Label Ranking (LR) studies the problem of learning a mapping from features to rankings over a finite set of labels. This task emerges in many domains. Common practical illustrations include pattern recognition (Geng & Luo, 2014), web advertisement (Djuric et al., 2014), sentiment analysis (Wang et al., 2011), document categorization (Jindal & Taneja, 2015) and bio-informatics (Balasubramaniyan et al., 2005). The importance of LR has spurred the development of several approaches for tackling this task from the perspective of the applied CS community (Vembu & Gärtner, 2010; Zhou et al., 2014b).

[1]School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece. Correspondence to: Eleni Psaroudaki <epsaroudaki@mail.ntua.gr>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

The overwhelming majority of these solutions comes with experimental evaluation and no theoretical guarantees; e.g., algorithms based on decision trees are a workhorse for practical LR and lack theoretical guarantees. Given state-of-the-art experimental results, based on Random Forests (see Zhou & Qiu, 2018), we are highly motivated not only to work towards a theoretical understanding of this central learning problem but also to theoretically analyze how efficient tree-based methods can be under specific assumptions.

LR comprises a supervised learning problem that extends multiclass classification (Dekel et al., 2003). In the latter, with instance domain $\mathbb{X} \subseteq \mathbb{R}^d$ and set of labels $[k] := \{1, \dots, k\}$, the learner draws i.i.d. labeled examples $(\boldsymbol{x}, y) \in \mathbb{X} \times [k]$ and aims to learn a hypothesis from instances to labels, following the standard PAC model. In LR, the learner observes labeled examples $(\boldsymbol{x}, \sigma) \in \mathbb{X} \times \mathbb{S}_k$ and the goal is to learn a hypothesis $h : \mathbb{X} \to \mathbb{S}_k$ from instances to *rankings of labels*, where $\mathbb{S}_k$ is the symmetric group of k elements. The ranking $h(\boldsymbol{x})$ corresponds to the preference list of the feature $\boldsymbol{x}$ and, as mentioned in previous works (Hüllermeier et al., 2008), a natural way to represent preferences is to evaluate individual alternatives through a real-valued utility (or score) function. Note that if the training data offer the utility scores directly, the problem is reduced to a standard regression problem.

In our work, we assume that there exists such an *underlying nonparametric score* function $\boldsymbol{m} : \mathbb{X} \to [0, 1]^k$, mapping features to score values. The value $m_i(\boldsymbol{x})$ corresponds to the score assigned to the label $i \in [k]$ for input $\boldsymbol{x}$ and can be considered proportional to the posterior probability $\mathbf{Pr}_{(\boldsymbol{x},y)}[y = i|\boldsymbol{x}]$. For each LR example $(\boldsymbol{x}, \sigma)$, the label σ is generated by sorting the underlying regression-score vector $\boldsymbol{m}(\boldsymbol{x})$, i.e., $\sigma = \mathrm{argsort}(\boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi})$ (with some regression noise $\boldsymbol{\xi}$). We are also interested in cases where some of the alternatives of σ are missing, i.e., we observe incomplete rankings $\sigma \in \mathbb{S}_{\leq k}$; the way that such rankings occur will be clarified later. Formally, we have:

Definition 1.1 (Distribution-free Nonparametric LR). Let $\mathbb{X} \subseteq \mathbb{R}^d$, $[k]$ be a set of labels, $\mathcal{C}$ be a class of functions from $\mathbb{X}$ to $[0, 1]^k$ and $\mathcal{D}_x$ be an arbitrary distribution over $\mathbb{X}$. Consider a noise distribution $\mathcal{E}$ over $\mathbb{R}^k$. Let $\boldsymbol{m}$ be an unknown target function in $\mathcal{C}$.

- An example oracle $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E})$ with complete rankings, works as follows: Each time $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E})$ is invoked, it returns a labeled example $(\boldsymbol{x}, \sigma) \in \mathbb{X} \times \mathbb{S}_k$, where (i) $\boldsymbol{x} \sim \mathcal{D}_x$ and $\boldsymbol{\xi} \sim \mathcal{E}$ independently and (ii) $\sigma = \mathrm{argsort}(\boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi})$. Let D_R be the joint distribution over $(\boldsymbol{x}, \sigma)$ generated by the oracle. In the noiseless case ($\boldsymbol{\xi} = \mathbf{0}$ almost surely), we simply write $\mathrm{Ex}(\boldsymbol{m})$.

- Let $\mathcal{M}$ be a randomized mechanism that given a tuple $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{X} \times \mathbb{R}^k$ generates an incomplete ranking $\mathcal{M}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{S}_{\leq k}$. An example oracle $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})$ with incomplete rankings, works as follows: Each time $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})$ is invoked, it returns a labeled example $(\boldsymbol{x}, \sigma) \in \mathbb{X} \times \mathbb{S}_{\leq k}$, where (i) $\boldsymbol{x} \sim \mathcal{D}_x$, $\boldsymbol{\xi} \sim \mathcal{E}$, (ii) $\boldsymbol{y} = \boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi}$ and (iii) $\sigma = \mathcal{M}(\boldsymbol{x}, \boldsymbol{y})$. Let $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R^{\mathcal{M}}$.

We denote $h : \mathbb{X} \to \mathbb{S}_k$ the composition $h = \mathrm{argsort} \circ \boldsymbol{m}$. Note that the oracle $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})$ generalizes $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E})$ (which generalizes $\mathrm{Ex}(\boldsymbol{m})$ accordingly) since we can set $\mathcal{M}$ to be $\mathcal{M}(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{argsort}(\boldsymbol{y})$.

1.1. Problem Formulation and Contribution

Most of our attention focuses on the two upcoming learning goals, which are stated for the abstract Label Ranking example oracle $\mathcal{O} \in \{\mathrm{Ex}(\boldsymbol{m}), \mathrm{Ex}(\boldsymbol{m}, \mathcal{E}), \mathrm{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})\}$. Let d be an appropriate ranking distance metric.

Problem 1 (Computational). The learner is given i.i.d. samples from the oracle $\mathcal{O}$ and its goal is to *efficiently* output a hypothesis $\widehat{h} : \mathbb{X} \to \mathbb{S}_k$ such that with high probability the error $\mathbf{E}_{\boldsymbol{x} \sim \mathcal{D}_x}[d(\widehat{h}(\boldsymbol{x}), h(\boldsymbol{x}))]$ is small.

Problem 2 (Statistical). Consider the median problem $h^\star = \mathrm{argmin}_h \, \mathbf{E}_{(\boldsymbol{x},\sigma)}[d(h(\boldsymbol{x}), \sigma)]$ where $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R$. The learner is given i.i.d. samples from the oracle $\mathcal{O}$ and its goal is to output a hypothesis $\widehat{h} : \mathbb{X} \to \mathbb{S}_k$ from some hypothesis class $\mathcal{H}$ such that with high probability the error $\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{D}_x}[\widehat{h}(\boldsymbol{x}) \neq h^\star(\boldsymbol{x})]$ against the median $h^\star$ is small.

The main gap in the theoretical literature of LR was the lack of computational guarantees. Problem 1 identifies this gap and offers, in combination with the generative models of the previous section, a natural and formal way to study the theoretical performance of practical methods for LR such as decision trees and random forests. We believe that this is the main conceptual contribution of our work. In Problem 1, the runtime should be polynomial in $d, k, 1/\varepsilon$.

While Problem 1 deals with computational aspects of LR, Problem 2 focuses on the statistical aspects, i.e., the learner may be computationally inefficient. This problem is extensively studied as Ranking Median Regression (Clémençon et al., 2018; Clémençon & Vogel, 2020) and is closely related to Empirical Risk Minimization (and this is why it is "statistical", since NP-hardness barriers may arise). We note that the median problem is defined w.r.t. $\mathcal{D}_R$ (over complete rankings) but the learner receives examples from $\mathcal{O}$ (which may correspond to incomplete rankings).

We study the distribution-free nonparametric LR task from either theoretical or experimental viewpoints in three cases:

Noiseless Oracle with Complete Rankings. In this setting, we draw samples from $\mathrm{Ex}(\boldsymbol{m})$ (i.e., $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E})$ with $\boldsymbol{\xi} = 0$). For this case, we resolve Problem 1 (under mild assumptions) and provide theoretical guarantees for efficient algorithms that use decision trees and random forests, built greedily based on the CART empirical MSE criterion, to interpolate the correct ranking hypothesis. This class of algorithms is widely used in applied LR but theoretical guarantees were missing. For the analysis, we adopt the *labelwise decomposition* technique (Cheng et al., 2013), where we generate one decision tree (or random forest) for each position of the ranking. We underline that decision trees and random forests are the state-of-the-art techniques for LR.

Contribution 1: We provide the first theoretical performance guarantees for these algorithms for Label Ranking under mild conditions. We believe that our analysis and the identification of these conditions contributes towards a better understanding of the practical success of these algorithms.

Noisy Oracle with Complete Rankings. We next replace the noiseless oracle of Problem 1 with $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E})$. In this noisy setting, the problem becomes challenging for theoretical analysis; we provide experimental evaluation aiming to quantify how noise affects the capability of decision trees and random forests to interpolate the true hypothesis.

Contribution 2: Our experimental evaluation demonstrates that random forests and shallow decisions trees are robust to noise, not only in our noisy setting, but also in standard LR benchmarks.

Noisy Oracle with Incomplete Rankings. We consider the oracle $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})$ with incomplete rankings. We resolve Problem 2 for the Kendall tau distance, as in previous works (so, we resolve it for the weaker oracles too). Now, the learner is agnostic to the positions of the elements in the incomplete ranking and so labelwise decomposition cannot be applied. Using *pairwise decomposition*, we compute a ranking predictor that achieves low misclassification error compared to the optimal classifier $h^\star$ and obtain sample complexity bounds for this task.

Contribution 3: Building on the seminal results of (Korba et al., 2017; Clémençon et al., 2018; Clémençon & Korba, 2018; Clémençon & Vogel, 2020), we give results for Problem 2 for incomplete rankings under appropriate conditions.

1.2. Related Work

LR has received significant attention over the years (Shalev-Shwartz, 2007; Hüllermeier et al., 2008; Cheng

& Hüllermeier, 2008; Har-Peled et al., 2003), due to the large number of practical applications. There are multiple approaches for tackling this problem (see Vembu & Gärtner, 2010; Zhou et al., 2014b, and the references therein). Some of them are based on probabilistic models (Cheng & Hüllermeier, 2008; Cheng et al., 2010; Grbovic et al., 2012; Zhou et al., 2014a). Others are tree and ensemble based, such as adaption of decision trees (Cheng et al., 2009), entropy based ranking trees and forests (de Sá et al., 2017), bagging techniques (Aledo et al., 2017), random forests (Zhou & Qiu, 2018), boosting (Dery & Shmueli, 2020), achieving highly competitive results. There are also works focusing on supervised clustering (Grbovic et al., 2013). Decomposition techniques are closely related to our work; they mainly transform the LR problem into simpler problems, e.g., binary or multiclass (Hüllermeier et al., 2008; Cheng & Hüllermeier, 2012; Cheng et al., 2013; Cheng & Hüllermeier, 2013; Gurrieri et al., 2014).

Comparison to Previous Work. To the best of our knowledge, there is no previous theoretical work focusing on the computational complexity of LR (a.k.a. Problem 1). However, there are many important works that adopt a statistical viewpoint. Closer to ours are the following seminal works on the statistical analysis of LR: Korba et al. (2017); Clémençon et al. (2018); Clémençon & Vogel (2020). Korba et al. (2017) introduced the statistical framework of consensus ranking (which is the unsupervised analogue of Problem 2) and identified crucial properties for the underlying distribution in order to get fast learning rate bounds for empirical estimators. A crucial contribution of this work (that we also make use of) is to prove that when Strict Stochastic Transitivity holds, the set of Kemeny medians (solutions of Problem 2 under the KT distance) is unique and has a closed form. Problem 2 was introduced in Clémençon et al. (2018), where the authors provide fast rates (under standard conditions) when the learner observes complete rankings, which reveal the relative order, but not the positions of the labels in the correct ranking. The work of Clémençon & Vogel (2020) provides a novel multiclass classification approach to Label Ranking, where the learner observes the top-label with some noise, i.e, observes only the partial information $\sigma_{\boldsymbol{x}}^{-1}(1)$ in presence of noise, under the form of the random label y assigned to $\boldsymbol{x}$. Our contribution concerning Problem 2 is a natural follow-up of these works where the learner observes noisy incomplete rankings (and so has only information about the relative order of the alternatives). Our solution for Problem 2 crucially relies on the conditions and the techniques developed in (Korba et al., 2017; Clémençon et al., 2018; Clémençon & Vogel, 2020). In our setting we have to modify the key conditions in order to handle incomplete rankings. Finally, our labelwise decomposition approach to Problem 1 is closely related to Korba et al. (2018), where many embeddings for ranking data are discussed.

Nonparametric Regression and CART. Regression trees constitute a fundamental approach in order to deal with nonparametric regression. Our work is closely related to the one of Syrgkanis & Zampetakis (2020), which shows that trees and forests, built greedily based on the CART empirical MSE criterion, provably adapt to sparsity in the high-dimensional regime. Specifically, Syrgkanis & Zampetakis (2020) analyze two greedy tree algorithms (they can be found at the Appendix D): (a) in the Level Splits variant, in each level of the tree, the same variable is greedily chosen at all the nodes in order to maximize the overall variance reduction; (b) in the Breiman's variant, which is the most popular in practice, the choice of the next variable to split on is locally decided at each node of the tree. In general, regression trees (Breiman et al., 1984) and random forests (Breiman, 2001) are one of the most widely used estimation methods by ML practitioners (Loh, 2011; Louppe, 2014). For further literature review and preliminaries on decision trees and random forests, we refer to the Appendix D.

Multiclass Prediction. In multiclass prediction with k labels, there are various techniques such as One-versus-All and One-versus-One (see Shalev-Shwartz & Ben-David, 2014). We adopt the OVO approach for Problem 2, where we consider $\binom{k}{2}$ binary sub-problems (Hastie & Tibshirani, 1998; Moreira & Mayoraz, 1998; Allwein et al., 2000; Fürnkranz, 2002; Wu et al., 2004) and we combine the binary predictions. A similar approach was employed for a variant of Problem 2 by Clémençon & Vogel (2020).

1.3. Notation

For vectors, we use lowercase bold letters $\boldsymbol{x}$; let x_i be the i-th coordinate of $\boldsymbol{x}$. We write $\text{poly}_\Box$ to denote that the degree of the polynomial depends on the subscripted parameters. Also, $\widetilde{O}(\cdot)$ is used to hide logarithmic factors. We denote the symmetric group over k elements with $\mathbb{S}_k$ and $\mathbb{S}_{\leq k}$ for incomplete rankings. For $i \in [k]$, we let $\sigma(i)$ denote the position of the i-th alternative.

As distance metrics we use the **Kendall Tau** (KT) distance

$$d_{KT}(\pi, \sigma) = \sum_{i<j} \mathbf{1}\{(\pi(i) - \pi(j))(\sigma(i) - \sigma(j)) < 0\},$$

the **Spearman** distance

$$d_2(\pi, \sigma) = \sum_{i \in [k]} (\pi(i) - \sigma(i))^2,$$

and the **KT coefficient**

$$k_\tau = 1 - \frac{4 d_{KT}(\pi, \sigma)}{k(k-1)},$$

i.e. the normalization of d_{KT} to the interval $[-1, 1]$.

The **Mean Squared Error** (MSE) of a function

$$f : \{0,1\}^d \to [0,1]$$

is equal to

$$\widetilde{L}(f,S) = \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\Big[\big(f(\boldsymbol{x}) - \mathop{\mathbf{E}}_{\boldsymbol{w}\sim\mathcal{D}_x}[f(\boldsymbol{w})|\boldsymbol{w}_S = \boldsymbol{x}_S]\big)^2\Big], \quad (1)$$

where $\boldsymbol{x}_S$ is the sub-vector of $\boldsymbol{x}$, where we observe only the coordinates with indices in $S \subseteq [d]$ and $\boldsymbol{x}_S \in \{0,1\}^{|S|}$. The VC dimension $\mathrm{VC}(\mathcal{G})$ of a class $\mathcal{G} \subseteq \{-1,+1\}^{\mathbb{X}}$ is the largest n such that there exists a set $T \subset \mathbb{X}$, $|T| = n$ and $\mathcal{G}$ shatters T. When $\mathrm{VC}(\mathcal{G}) < \infty$, $\mathcal{G}$ is said to be a **VC class**.

2. Our Results

We provide an overview of our contributions on distribution-free Label Ranking settings, as introduced in Definition 1.1.

2.1. Noiseless Oracle with Complete Rankings

We begin with Label Ranking as *noiseless nonparametric regression* (Tsybakov, 2008). This corresponds to the example oracle $\mathrm{Ex}(\boldsymbol{m})$ of Definition 1.1 which we recall now: For an underlying score hypothesis $\boldsymbol{m} : \mathbb{X} \to [0,1]^k$, where k is the number of labels and $m_i(\boldsymbol{x})$ is the score of the alternative $i \in [k]$ with respect to $\boldsymbol{x}$. The learner observes a labeled example $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R$. It holds that $\sigma = h(\boldsymbol{x}) = \mathrm{argsort}(\boldsymbol{m}(\boldsymbol{x}))$.

We resolve Problem 1 for the $\mathrm{Ex}(\boldsymbol{m})$ oracle: We provide the first theoretical guarantees in the LR setting for the performance of algorithms based on decision trees and random forests, when the feature space is the Boolean hypercube $\mathbb{X} = \{0,1\}^d$ under mild assumptions, using the labelwise decomposition technique (Cheng et al., 2013). We underline once again that this class of algorithms constitutes a fundamental tool for *practical works* to solve LR; this heavily motivates the design of our theory. We focus on the performance of regression trees and forests in high dimensions. Crucially, Definition 1.1 makes no assumptions on the structure of the underlying score hypothesis $\boldsymbol{m}$. In order to establish our theoretical guarantees, we are going to provide a pair of structural conditions for the score hypothesis $\boldsymbol{m}$ and the features' distribution $\mathcal{D}_x$. We will now state these conditions; for this we will need the definition of the mean squared error that can be found at (1).

Condition 1. Consider the feature space $\mathbb{X} = \{0,1\}^d$ and the regression vector-valued function $\boldsymbol{m} : \{0,1\}^d \to [0,1]^k$ with $\boldsymbol{m} = (m_1, \dots, m_k)$. Let $\mathcal{D}_x$ be the distribution over features. We assume that the following hold for any $j \in [k]$.

1. (Sparsity) The function $m_j : \{0,1\}^d \to [0,1]$ is r-sparse, i.e., it depends on r out of d coordinates.
2. (Approximate Submodularity) The mean squared error $\widetilde{L}_j$ of m_j is C-approximate-submodular, i.e., for any $S \subseteq T \subseteq [d]$, $i \in [d]$, it holds that

$$\widetilde{L}_j(T) - \widetilde{L}_j(T \cup \{i\}) \le C \cdot \Big(\widetilde{L}_j(S) - \widetilde{L}_j(S \cup \{i\})\Big)\,.$$

Some comments are in order: (1) The approximate submodularity condition for the mean squared error is the more technical condition, which however is provably *necessary* (Syrgkanis & Zampetakis, 2020) to obtain meaningful results about the consistency of greedily grown trees in high dimensions. (2) For the theoretical analysis, we constrain ourselves to the case where all features are binary. However, in the experimental part, we test the performance of the method with non-binary features too. (3) Sparsity should be regarded as a way to parameterize the class of functions $\boldsymbol{m}$, rather than a restriction. Any function is r-sparse, for some value of r. However, our results are interesting when $r \ll d$ and establish that decision trees and random forests *provably behave well under sparsity*. As we will see in Theorem 2.1, the sample complexity has an r^r dependence, which cannot be avoided, since the class of functions is nonparametric. Observe that both m_i and m_j are r-sparse but they are not constrained to depend on the same set of coordinates; the function $\boldsymbol{m}$ is at most $(k \cdot r)$-sparse, where $k \cdot r \ll d$, and we say that $\boldsymbol{m}$ is $\boldsymbol{r}$**-sparse**. These are the state-of-the-art conditions for the high-dimensional regime (Syrgkanis & Zampetakis, 2020).

Our algorithm for Problem 1 uses decision trees via the Level Splits criterion. In this criterion, a set of splits $S \subseteq [d]$ is collected greedily and any tree level has to split at the same direction $i \in [d]$. Intuitively, the approximate submodularity condition captures the following phenomenon: "If adding i does not decrease the mean squared error significantly at some point (when having the set S), then i cannot decrease the mean squared error significantly in the future either (for any superset of S)".

Our main result for Problem 1 using decision trees with Level Splits follows. Recall that $h(\boldsymbol{x}) = \mathrm{argsort}(\boldsymbol{m}(\boldsymbol{x}))$ and d_2 is the Spearman distance (i.e., L_2 squared over the rankings' positions).

Theorem 2.1 (Noiseless LR (Informal))**.** *Under Condition 1 with parameters r, C, there exists an algorithm (Decision Trees via Level-Splits - Algorithm 1) that draws $n = \widetilde{O}\left(\log(d) \cdot \mathrm{poly}_{C,r}(k) \cdot (Cr/\epsilon)^{Cr+2}\right)$ independent samples from $\mathrm{Ex}(\boldsymbol{m})$ and, in $\mathrm{poly}_{C,r}(d, k, 1/\epsilon)$ time, computes a set of splits S_n and an estimate $h^{(n)}(\cdot\,; S_n) : \{0,1\}^d \to \mathbb{S}_k$ which, with probability 99%, satisfies $\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[d_2(h(\boldsymbol{x}), h^{(n)}(\boldsymbol{x}; S_n))\right] \le \epsilon\,.$*

See also Theorem A.4. This is the first sample complexity guarantee in LR for decision tree-based algorithms. We also provide results and algorithms for the Breiman's criterion (Appendix A.1) and for Random Forests (Appendix A.2). Our result can be read as: *In practice, sparsity of the instance's "score function" is one of the reasons why such algorithms work well and efficiently in real world.*

The description of Algorithm 1 follows. Given $(\boldsymbol{x}, \sigma) \sim$

$\mathcal{D}_R$, we transform the ranking-label σ to a vector $\boldsymbol{y} = \boldsymbol{m}_C(\sigma) \in [0,1]^k$, where $\boldsymbol{m}_C(\sigma)$ is the canonical representation of the ranking σ. Specifically, one can obtain the score vector $\boldsymbol{y}$ by setting $y_i = m_{C,i}(\sigma)$ equal to $\sigma(i)/k$, i.e., the position of the i-th alternative in the permutation σ, normalized by k, where k is the length of the permutation (see Line 3 of Algorithm 1). Hence, we obtain a training set of the form $T = (\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})_{i\in[N]}$. Our goal is to fit the score vectors $\boldsymbol{y}^{(i)}$ using decision trees (or random forests depending on the black-box algorithm that we will choose to apply). During this step, we have to feed our training set T into a learning algorithm that fits the function $\boldsymbol{m}_C \circ h : \mathbb{X} \to [0,1]^k$, where $\circ$ denotes composition. We remark that since the regression function $\boldsymbol{m}$ is sparse, this vector-valued function is sparse too. We do this as follows.

Algorithm 1 Algorithm of Theorem 2.1

```
1: Set $n \leftarrow \widetilde{\Theta}(\log(d/\delta) \cdot \mathrm{poly}_{C,r}(k) \cdot (Cr/\varepsilon)^{Cr+2})$
2: Draw $n$ samples $(\boldsymbol{x}^{(j)}, \sigma^{(j)}) \sim \mathcal{D}_R, j \in [n]$
3: For any $j \in [n]$, set $\boldsymbol{y}^{(j)} \leftarrow (\sigma^{(j)}(i)/k)_{i\in[k]}$
4: Create $k$ datasets $T_i = \{(\boldsymbol{x}^{(j)}, y_i^{(j)})\}_{j\in[n]}$
5: for $i \in [k]$ do
6:    $m_i^{(n)}, S_n^{(i)} = \texttt{LevelSplits}(T_i, \widetilde{\Theta}(Cr\log(k)))$
7: endfor
8: Output argsort $\circ\, (m_1^{(n)}(\cdot; S_n^{(1)}), ..., m_k^{(n)}(\cdot; S_n^{(k)}))$
```

We decompose the training set T into k data sets T_i, where the labels are no more vectors but real values (*labelwise decomposition*, see Line 4 of Algorithm 1). For each T_i, we apply the Level Splits method and finally we combine our estimates, where we have to aggregate our estimates into a ranking (see Line 8 of Algorithm 1). Algorithm 1 uses the routine `LevelSplits`, which computes a decision tree estimate based on the input training set (see Algorithm 9 (with $h = 0$) in Appendix D). The second argument of the routine is the maximum number of splits H (height of the tree) and in the above result $H = \widetilde{\Theta}(Cr\log(k))$. The routine iterates H times, one for each level of the tree: at every level, we choose the direction $i \in [d]$ that minimizes the total empirical mean squared error (greedy choice) and the space is partitioned recursively based on whether $x_i = 0$ or 1. The routine outputs the estimated function and the set of splits $S \subseteq [d]$. For a proof sketch, see Section 3.

2.2. Noisy Oracle with Complete Rankings

We remark that the term 'noiseless' in the above regression problem is connected with the output (the ranking), i.e., in the generative process given $\boldsymbol{x} \in \mathbb{X}$, we will constantly observe the same output ranking. Let us consider the oracle $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E})$, that corresponds to *noisy nonparametric regression*: Draw $\boldsymbol{x} \sim \mathcal{D}_x$ and independently draw $\boldsymbol{\xi} \in [-1/4, 1/4]^k$ from a zero mean noise distribution $\mathcal{E}$. Compute $\boldsymbol{y} = \boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi}$, rank the alternatives $\sigma = \mathrm{argsort}(\boldsymbol{y})$ and output $(\boldsymbol{x}, \sigma)$. Due to the noise vector $\boldsymbol{\xi}$, we may observe e.g., different rankings for the same $\boldsymbol{x}$ feature. We provide the following notions of inconsistency.

Definition 2.2 (Output Inconsistency)**.** Let $\sigma = \mathrm{argsort}(\boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi})$ denote the observed ranking of $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E})$. The noise distribution $\mathcal{E}$ satisfies

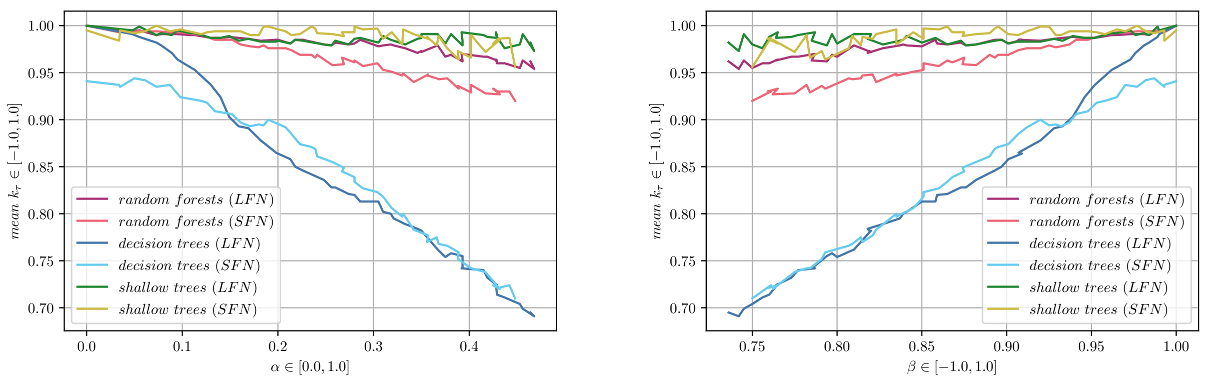
(i) the α-inconsistency property if there exists $\alpha \in [0,1]$ so that $\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\mathbf{Pr}_{\boldsymbol{\xi}\sim\mathcal{E}}[h(\boldsymbol{x}) \neq \sigma]\right] \leq \alpha$, and

(ii) the β-k_τ gap property if there exists $\beta \in [-1,1]$ so that $\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}_x}\, \mathbf{E}_{\boldsymbol{\xi}\sim\mathcal{E}}[k_\tau(h(\boldsymbol{x}), \sigma))] = \beta$.

Property (i) captures the phenomenon that the probability that the observed ranking σ differs from the correct one $h(\boldsymbol{x})$ is roughly α over the feature space. However, it does not capture the distance between these two rankings; this is why we need property (ii) which captures the expected similarity (recall the definition of the KT coefficient k_τ in Section 1.3). The case $\alpha = 0$ (resp. $\beta = 1$) gives our noiseless setting. When $\alpha > 0$ (resp. $\beta < 1$), the structure of the problem changes and our theoretical guarantees fail. Interestingly, this is due to the fact that the geometry of the input noise is structurally different from the observed output. The input noise acts additively to the vector $\boldsymbol{m}(\boldsymbol{x})$, while the output is computed by permuting the elements. Hence, the relation between the observed ranking and the expected one is no more linear and hence one needs to extend the standard 'additive' nonparametric regression setting $y = f(x) + \xi$ to another geometry dealing with rankings $\sigma = \mathcal{E}_\theta \circ f(x)$, where $f : \mathbb{X} \to \mathbb{S}_k$ is the regression function and $\mathcal{E}_\theta : \mathbb{S}_k \to \mathbb{S}_k$ is a parameterized noise operator (e.g., a Mallows model). This change of geometry is interesting and to the best of our knowledge cannot be captured by existing theoretical results (see Appendix B for the experimental results of different 'additive' nonparametric noise settings).

Our experimental results aim to complement and go beyond our theoretical understanding of the capability of the decision trees and random forests to interpolate the correct underlying regression function with the presence of regression noise. To this end, we consider the oracle $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E})$ where the noise distribution $\mathcal{E}$ satisfies either property (i) or (ii) of Definition 2.2.

For the experimental evaluation, two synthetic data set families were used, namely LFN (Large Features Number) and SFN (Small Features Number), consisting of a single noiseless and 50 noisy data sets, respectively. For either data set family, a common $\boldsymbol{m} : \{0,1\}^d \to [0,1]^k$ was employed, accordingly. Each noiseless data set was created according to the oracle $\mathrm{Ex}(\boldsymbol{m})$. It consists of 10000 samples $(\boldsymbol{x}, \sigma)$, where $\boldsymbol{x} \in \{0,1\}^d$ ($d = 1000$ for LFN and $d = 100$ for SFN) and $\sigma \in \mathbb{S}_5$ ($k = 5$), with $r = 10$ informative binary features per label (sparsity). The noisy data sets were



(a) (b)

Figure 1. Illustration of the experimental results in terms of mean k_τ for different noise distributions $\mathcal{E}$ with respect to (a) α-inconsistency; (b) β-k_τ gap.

produced according to the generative process $\text{Ex}(\boldsymbol{m}, \mathcal{E})$, each using a different zero-mean noise distribution $\mathcal{E}$. We implemented modified versions of Algorithm 1. The shallow trees (●,●), fully grown decision trees (●,●) and random forests (●,●) were built greedily based on the CART empirical MSE criterion, using the Breiman's method instead of Level Splits.

Figure 1 summarizes the experimental results for different values of $\alpha \in [0, 1]$ and $\beta \in [-1, 1]$. Results are obtained in terms of mean k_τ, using the noisy data as training set and noiseless data as validation set. The y axis of Figure 1 depicts the mean k_τ of all the corresponding pairs of the output of our model and the given noiseless ranking in the test sets. At this point we remind the reader that k_τ is a normalization of d_{KT} in $[-1, 1]$ and measures the proportion of the concordant pairs in two rankings, with $k_\tau = 1$ meaning that the two ranking have perfect agreement (i.e., the two rankings are exactly the same), while $k_\tau = -1$ that the two rankings have perfect disagreement.

As expected, decision trees as well as random forests interpolate the $\boldsymbol{m}$ function successfully in the noiseless setting, since it is $\boldsymbol{r}$-sparse. Moreover, the increase of noise level leads to the decay of the decision trees' performance, indicated by the α-inconsistency. However, the β-k_τ gap is a more appropriate noise level measure, because it quantifies the degree of deviation rather than the existence of it. The ratio of the performance in terms of mean k_τ over β-k_τ gap is approximately equal to one, revealing that decision trees fit the noise. On the contrary, shallow trees have better ability to generalize and avoid overfitting. Fully grown honest random forests are also resistant to overfitting due to bagging, and therefore are noise tolerant. This is visible in Figure 1 where the graphs of the random forests and of the shallow trees attain a mean KT coefficient close to 1 for most values of α-inconsistency and β-k_τ gap.

Appendix B presents the experimental setting in more detail and additional experimental results of LR standard benchmarks and different 'additive' nonparametric noise settings. The code and data sets to reproduce our results are available: `https://github.com/pseleni/LR-nonparametric-regression`.

Remark 2.3. We have encountered Problem 1 with the oracles $\text{Ex}(\boldsymbol{m})$ and $\text{Ex}(\boldsymbol{m}, \mathcal{E})$. A natural question is what happens if we use the incomplete oracle $\text{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})$. In this setting, the position of each alternative is not correctly observed (we observe a ranking with size $\ell \leq k$, see Section 2.3). Hence, we cannot apply our *labelwise* method for this oracle. One could obtain similar results for Problem 1 for incomplete rankings using pairwise decomposition, but we leave it for future work. Next we focus on Problem 2.

2.3. Noisy Oracle with Incomplete Rankings

We now study Problem 2, where we consider a metric d in $\mathbb{S}_k$, a distribution $\mathcal{D}_R$ over $\mathbb{X} \times \mathbb{S}_k$ that corresponds to the example oracle $\text{Ex}(\boldsymbol{m}, \mathcal{E})$ and set up the task of finding a measurable mapping $h : \mathbb{X} \to \mathbb{S}_k$ that minimizes the objective $R(h) = \mathbf{E}_{(\boldsymbol{x},\sigma)\sim\mathcal{D}_R}[d(h(\boldsymbol{x}), \sigma)]$. In this work, we focus on the Kendall tau distance $(d = d_{KT})$ and ask how well we can estimate the minimizer of the above population objective if we observe i.i.d. samples from the incomplete rankings' oracle $\text{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})$. We underline that in what follows whenever we refer to Problem 2, we have $d = d_{KT}$ in mind. A natural question is: What is the optimal solution? In binary classification, the learner aims to estimate the Bayes classifier, since it is known to minimize the misclassification error among all classifiers (Massart & Nédélec, 2006). Problem 2 deals with rankings and is well-studied in previous works when: d is the Kendall tau distance and the learner either observes complete rankings (Clémençon et al., 2018) or observes only the top element (under some BTL noise) (Clémençon & Vogel, 2020). As we will see later, the optimal solution $h^\star$ of Problem 2 is unique under mild conditions on $\mathcal{D}_R$, due to Korba et al. (2017). Our goal will be to estimate $h^\star$ from labeled examples generated by $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})$.

We are going to introduce the example oracle $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})$: We are interested in the case where the mechanism $\mathcal{M}$ generates incomplete rankings and captures a general spectrum of ways to generate such rankings (e.g., Hüllermeier et al., 2008). We begin with the incomplete rankings' mechanism $\mathcal{M}$. We assume that there exists a survival probabilities vector $\boldsymbol{q} : \mathbb{X} \to [0,1]^k$, which is feature-dependent, i.e., the vector $\boldsymbol{q}$ depends on the input $\boldsymbol{x} \in \mathbb{X}$. Hence, for the example $\boldsymbol{x}$ and alternative $i \in [k]$, with probability $q_i(\boldsymbol{x})$, we set the score y_i equal to a noisy value of the score $m_i(\boldsymbol{x})$ (the alternative i survives) and, otherwise, we set $y_i = \star$. We mention that the events of observing i and j are not necessarily independent and so do not necessarily occur with probability $q_i(\boldsymbol{x})q_j(\boldsymbol{x})$. We denote the probability of the event "Observe both i and j in $\boldsymbol{x}$" by $q_{i,j}(\boldsymbol{x})$. We modify the $\mathrm{argsort} : [0,1]^k \to \mathbb{S}_k$ routine so that it will ignore the $\star$ symbol in the ranking, e.g., $\mathrm{argsort}(0.4, \star, 0.7, \star, 0.1) = (c \succ a \succ e)$. Crucially, we remark that another variant would preserve the $\star$ symbols: this problem is easier since it reveals the correct position of the non-erased alternatives. In our model, the information about the location of each alternative is not preserved. In order to model regression noise, we consider a noise distribution $\mathcal{E}$ over the bounded cube $[-1/4, 1/4]^k$. Hence, we model $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})$ as follows:

Definition 2.4 (Generative Process for Incomplete Data)**.** Consider an underlying score hypothesis $\boldsymbol{m} : \mathbb{X} \to [0,1]^k$ and let $\mathcal{D}_x$ be a distribution over features. Let $\boldsymbol{y} : [k] \to [0,1] \cup \{\star\}$ and consider the survival probabilities vector $\boldsymbol{q} : \mathbb{X} \to [0,1]^k$. Each sample $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R^{\boldsymbol{q}}$ is generated as follows: (i) Draw $\boldsymbol{x} \in \mathbb{X}$ from $\mathcal{D}_x$ and $\boldsymbol{\xi} \in [-\frac{1}{4}, \frac{1}{4}]^k$ from $\mathcal{E}$; (ii) draw $\boldsymbol{q}(\boldsymbol{x})$-biased coins $\boldsymbol{c} \in \{-1,+1\}^k$; (iii) if $c_i > 0$, set $y_i = m_i(\boldsymbol{x}) + \xi_i$, else $y_i = \star$; (iv) compute $\sigma = \mathrm{argsort}(\boldsymbol{y})$, ignoring the $\star$ symbol.

In what follows, we resolve Problem 2 for the example oracle of Definition 2.4 and $d = d_{KT}$. As in the complete ranking case, Definition 2.4 imposes restrictions neither on the structure of the true score hypothesis $\boldsymbol{m}$ nor on the noise distribution $\mathcal{E}$. In order to resolve Problem 2, we assume the following. Recall that $q_{i,j}(\boldsymbol{x})$ is the probability of the event "Observe both i and j in $\boldsymbol{x}$".

Condition 2. Let $p_{ij}(\boldsymbol{x}) = \mathbf{Pr}_{\boldsymbol{\xi}\sim\mathcal{E}}[m_i(\boldsymbol{x}) + \xi_i > m_j(\boldsymbol{x}) + \xi_j | \boldsymbol{x}]$ for $\boldsymbol{x} \in \mathbb{R}^d$. For any $1 \le i < j \le k$, we assume that the following hold.

1. (Strict Stochastic Transitivity) For any $\boldsymbol{x} \in \mathbb{R}^d$ and any $u \in [k]$, we have that $p_{ij}(\boldsymbol{x}) \neq 1/2$ and $(p_{iu}(\boldsymbol{x}) > 1/2 \wedge p_{uj}(\boldsymbol{x}) > 1/2) \Rightarrow p_{ij}(\boldsymbol{x}) > 1/2$.
2. (Tsybakov's Noise Condition) There exists $a \in [0,1]$ and $B > 0$ so that the probability that a random feature $\boldsymbol{x} \sim \mathcal{D}_x$ satisfies $|p_{ij}(\boldsymbol{x}) - 1/2| < 2t$, is at most $B \cdot t^{a/(1-a)}$ for all $t \ge 0$.
3. (Deletion Tolerance) There exists $\phi \in (0,1]$ so that $q_{i,j}(\boldsymbol{x}) \ge \phi$ for any $\boldsymbol{x} \in \mathbb{R}^d$, where $q_{i,j}(\boldsymbol{x})$ is the survival probability of the pair $i < j$ in $\boldsymbol{x}$.

Importance of Item 1: Our goal is to find a good estimate for the minimizer $h^\star$ of $R(h) = \mathbf{E}_{(\boldsymbol{x},\sigma)\sim\mathcal{D}_R}[d_{KT}(h(\boldsymbol{x}), \sigma)]$. As observed in previous works (Korba et al., 2017; Clémençon et al., 2018), this problem admits a unique solution (with closed form) under mild assumptions on $\mathcal{D}_R$ and specifically this is assured by Strict Stochastic Transitivity (SST) of the pairwise probabilities $p_{ij}(\boldsymbol{x})$. The first condition guarantees (Korba et al., 2017) that the minimizer of $R(h)$ is almost surely unique and is given, with probability one and for any $i \in [k]$, by

$$h^\star(\boldsymbol{x}; i) = 1 + \sum_{j \neq i} \mathbf{1}\{p_{ij}(\boldsymbol{x}) < 1/2\}\,. \tag{2}$$

This is the well-known Copeland rule and we note that SST is satisfied by most natural probabilistic ranking models.

Importance of Item 2: The Tsybakov's noise condition is standard in binary classification and corresponds to a realistic noise model (Boucheron et al., 2005). In its standard form, this noise condition naturally requires that the regression function of a binary problem $\eta(\boldsymbol{x}) = \mathbf{E}[Y|X = \boldsymbol{x}] = 2\,\mathbf{Pr}[Y = +1|X = \boldsymbol{x}] - 1$ is close to the the critical value 0 with low probability over the features, i.e., the labels are not completely random for a sufficiently large portion of the feature space. We consider that, for any two alternatives $i \neq j$, this condition must be satisfied by the true score function $\boldsymbol{m}$ and the noise distribution $\mathcal{E}$ (specifically the functions m_i and m_j and the random variables ξ_i, ξ_j). This condition is very common in binary classification since it guarantees "fast rates" and has been previously applied to LR (Clémençon & Vogel, 2020).

Importance of Item 3: The last condition is natural in the sense that we need to observe the pair (i,j) at some frequency in order to achieve some meaningful results. Variants of this condition have already appeared in the incomplete rankings literature (see e.g., Fotakis et al., 2021) and in previous works in Label Ranking (see Clémençon & Vogel, 2020). We note that if we relax this condition to state that we only observe the pair i, j only in a portion of $\mathbb{R}^d$ (e.g., $q_{ij}(\boldsymbol{x}) = 0$ for 40% of $\boldsymbol{x}$'s), then we will probably miss a crucial part of the structure of the underlying mapping. This intuitively justifies the reason that we need deletion tolerance to hold for any $\boldsymbol{x} \in \mathbb{R}^d$.

Having described our conditions, we continue with our approach. We first remind the reader that the labelwise perspective we adopted in the complete case now fails. Second, since our data are incomplete, the learner cannot recover the optimal ranking rule $h^\star = \mathrm{argmin}_h \mathbf{E}[d_{KT}(h(\boldsymbol{x}), \sigma)]$ by simply minimizing an empirical version of this objective. To tackle this problem, we adopt a pairwise comparisons approach. In fact, the key idea is the closed form of the optimal ranking rule: From (2), we can write $h^\star(\boldsymbol{x}; i) = 1 + \sum_{j \neq i} \mathbf{1}\{h^\star_{ij}(\boldsymbol{x}) = -1\}$ where $h^\star_{ij}$ is the Bayes optimal classifier of the *binary sub-problem of the pair* $i \neq j$. We provide our result based on the standard One-Versus-One (OVO) approach, reducing this complex problem into multiple binary ones: We reduce the ranking problem into $O(k^2)$ binary sub-problems and each sub-problem corresponds to a pairwise comparison between the alternatives i and j for any $1 \leq i < j \leq k$. We solve each sub-problem separately by obtaining the Empirical Risk Minimizer $\widehat{h}_{ij}$ (for a fixed VC class $\mathcal{G}$, as in the previous works) whose risk is compared to the optimal $h^\star_{ij}$ and then we aggregate the $\binom{k}{2}$ binary classifiers into a single output hypothesis $\widehat{h} : \mathbb{R}^d \to \mathbb{S}_k$. We compare the generalization of this empirical estimate $\widehat{h}$ with the optimal predictor $h^\star$ of (2). We set $L_{ij}(g) = \mathbf{E}[g(\boldsymbol{x}) \neq \mathrm{sgn}(\sigma(i) - \sigma(j)) | \sigma \ni \{i, j\}]$ where $(\boldsymbol{x}, \sigma) \sim \mathcal{D}^{\boldsymbol{q}}_R$. Our main result in this setting follows.

Theorem 2.5 (Noisy and Incomplete LR). *Let $\epsilon, \delta \in (0, 1)$. Consider a hypothesis class $\mathcal{G}$ of binary classifiers with finite VC dimension. Under Condition 2 with parameters a, B, ϕ, there exists an algorithm (Algorithm 2) that draws*

$$n = \widetilde{O}\left(\frac{k^{\frac{4(1-a)}{a}}}{\mathrm{poly}_a(\phi \cdot \varepsilon)} \cdot \max\left\{ \log\left(\frac{k}{\delta}\right), \mathrm{VC}(\mathcal{G}) \right\} \right)$$

samples from $\mathcal{D}^{\boldsymbol{q}}_R$, as in Definition 2.4, and computes an estimate $\widehat{h} : \mathbb{R}^d \to \mathbb{S}_k$ so that $\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{D}_x}[\widehat{h}(\boldsymbol{x}) \neq h^\star(\boldsymbol{x})]$ is, with probability $1 - \delta$, at most

$$\frac{C_{a,B}}{\phi^2} \left(2 \sum_{i<j} \left(\inf_{g \in \mathcal{G}} L_{i,j}(g) - L^\star_{i,j} \right)^a \right) + \varepsilon \,, \qquad (3)$$

where $h^\star$ is the optimal predictor of (2) and $L^\star_{i,j}$ is the loss of the binary Bayes classifiers $h^\star_{i,j}$ for $1 \leq i < j \leq k$, where $C_{a,B}$ is a constant depending on a, B.

Our result for incomplete rankings is a PAC result, in the sense that we guarantee that, when optimizing over a VC class $\mathcal{G}$, the gap between the empirical estimate (the algorithm's output) and the optimal predictor of (2) is at most $C \cdot \mathrm{OPT} + r_n(\delta)$, where OPT is (a function of) the gap between the best classifier in the class $(\mathrm{argmin}_{g \in \mathcal{G}} L(g))$ and the Bayes classifier and $r_n(\delta)$ is a function which tends to 0 as the number of samples n increases (see (3)). We remark that the algorithm does not come with a computational efficiency guarantee, since the results are based on the computation of the ERM of each pairwise comparison $i < j$. In general, this is NP-hard but if the binary hypothesis class $\mathcal{G}$ is "simple" then we also obtain computational guarantees.

Algorithm 2 Algorithm of Theorem 2.5

```
1: $T \leftarrow n$ i.i.d. samples $(\boldsymbol{x}, \sigma) \sim \mathcal{D}^{\boldsymbol{q}}_R$ (as in Theorem 2.5)
2: For any $i \neq j$, set $T_{ij} = \emptyset$
3: for $1 \leq i < j \leq k$ do
4:     if $(\boldsymbol{x}, \sigma) \in T$ and $\sigma \ni \{i, j\}$ then
5:         Add $(\boldsymbol{x}, \mathrm{sgn}(\sigma(i) - \sigma(j)))$ to $T_{ij}$
6:     endif
7: endfor
8: $\widehat{\boldsymbol{s}} \leftarrow$ EstimateAggregate($T_{ij}$ for $i < j, \mathcal{G}$)
9: On input $\boldsymbol{x} \in \mathbb{R}^d$, output argsort($\widehat{\boldsymbol{s}}(\boldsymbol{x})$) breaking arbitrarily possible ties.
```

Algorithm 2 works as follows: Given a training set T of the form $(\boldsymbol{x}^{(i)}, \sigma^{(i)})$ with incomplete rankings, the algorithm creates $\binom{k}{2}$ datasets T_{ij} with the following criterion: For any $i < j$, if $(\boldsymbol{x}, \sigma) \in T$ and $\sigma \ni \{i, j\}$, the algorithm adds to the dataset T_{ij} the example $(\boldsymbol{x}, \mathrm{sgn}(\sigma(i) - \sigma(j)))$. For any such binary dataset, the algorithm computes the ERM and aggregates the estimates to $\widehat{\boldsymbol{s}}$ (these routines can be found as Algorithm 6). This aggregate rule is based on the structure of the optimal classifier $h^\star$ (that is valid due to the SST condition). The final estimator is the function $\widehat{h}$ that, on input $\boldsymbol{x} \in \mathbb{R}^d$, outputs the ranking $\widehat{h}(\boldsymbol{x}) = \mathrm{argsort}(\widehat{\boldsymbol{s}}(\boldsymbol{x}))$ (by breaking ties randomly).

Remark 2.6. (i) The oracle $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})$ and Definition 2.4 can be adapted to handle *partial* rankings (see Appendix C). (ii) Theorem 2.4 directly controls the risk gap $R(\widehat{h}) - R(h^\star) \leq \mathbf{E}_{\boldsymbol{x}}[d_{KT}(\widehat{h}(\boldsymbol{x}), h^\star(\boldsymbol{x}))]$, since $d_{KT}(\pi, \sigma) \leq k^2 \mathbf{1}\{\pi \neq \sigma\}$. (iii) We studied Problem 2 for $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E}, \mathcal{M})$. Our results can be transferred to the oracles $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E})$ and $\mathrm{Ex}(\boldsymbol{m})$ under Condition 2 with $\phi = 1$. (iv) This result is similar to Clémençon & Vogel (2020), where the learner observes $(\boldsymbol{x}, y)$ where $y = \sigma_{\boldsymbol{x}}^{-1}(1)$ is the top-label. To adapt our incomplete setting to theirs, we must erase positions instead of alternatives. If $\widetilde{q}_i$ is the probability that the i-th position survives, then we have that, in Clémençon & Vogel (2020): $\widetilde{q}_1(\boldsymbol{x}) = 1$ and $\widetilde{q}_{i \neq 1}(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in \mathbb{R}^d$. Also, the generative processes of the two works are different (noisy nonparametric regression vs. Plackett-Luce based models). In general, our results and our analysis for Problem 2 are very closely related and rely on the techniques of Clémençon & Vogel (2020).

3. Technical Overview

Proof Sketch of Theorem 2.1 Our starting point is the work of Syrgkanis & Zampetakis (2020), where they provide a collection of nonparametric regression algorithms based on decision trees and random forests. We have to provide a vector-valued extension of these results. For decision trees, we show (see Theorem A.3) that if the learner observes i.i.d. samples $(\boldsymbol{x}, \boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi})$ for some unknown target $\boldsymbol{m}$ satisfying Condition 1 with $r, C > 0$, then there exists an algorithm $\mathtt{ALGO}$ that uses $n = \widetilde{O}\left(\log(d) \cdot \mathrm{poly}_{C,r}(kCr/\varepsilon)\right)$ samples and computes an estimate $\boldsymbol{m}^{(n)}$ which satisfies $\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left\|\boldsymbol{m}(\boldsymbol{x}) - \boldsymbol{m}^{(n)}(\boldsymbol{x})\right\|_2^2\right] \leq \varepsilon$ with probability 99%. In our setting, we receive examples from $\mathrm{Ex}(\boldsymbol{m})$ and set $h(\boldsymbol{x}) = \mathrm{argsort}(\boldsymbol{m}(\boldsymbol{x}))$. As described in Algorithm 1, we design the training set $T = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}$, where $\boldsymbol{y}^{(i)} = (h(\boldsymbol{x}^{(i)}; j))_{j\in[k]}$ (we make the ranking $h(\boldsymbol{x})$ a vector). We provide T as input to $\mathtt{ALGO}$ (with $\boldsymbol{\xi} = \mathbf{0}$) and get a vector-valued estimation $\boldsymbol{m}^{(n)}$ that approximates the vector $(h(\cdot;1), ..., h(\cdot;k))$. Our next goal is to convert the estimate $\boldsymbol{m}^{(n)}$ to a ranking by setting $\widehat{h} = \mathrm{argsort} \circ \boldsymbol{m}^{(n)}$. In Theorem A.4, we show that rounding our estimations back to permutations will yield bounds for the expected Spearman distance $\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}_x}[d_2(h(\boldsymbol{x}), \widehat{h}(\boldsymbol{x}))]$.

Proof Sketch of Theorem 2.5 Consider the VC class $\mathcal{G}$ consisting of mappings $g : \mathbb{R}^d \to \{-1, +1\}$. Let $\mathcal{G}_{i,j} = \{g_{i,j} : \mathbb{R}^d \to \{-1, +1\}\}$ be a copy of $\mathcal{G}$ for the pair (i, j). We let $\widehat{g}_{i,j}$ and $g^\star_{i,j}$ be the algorithm's empirical classifier and the Bayes classifier respectively for the pair (i, j). The first key step is that the SST property (which holds thanks to Item 1) implies that the optimal ranking predictor $h^\star$ is unique almost surely and satisfies (2). Thanks to the structure of the optimal solution, we can compute the score estimates $\widehat{s}(\boldsymbol{x}; i) = 1 + \sum_{j\neq i} \mathbf{1}\{\widehat{g}_{i,j}(\boldsymbol{x}) = -1\}$ for any $i \in [k]$ and we will set $\widehat{h}$ to be $\mathrm{argsort} \circ \widehat{s}$. We first show that $\mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{D}_x}[\widehat{h}(\boldsymbol{x}) \neq h^\star(\boldsymbol{x})] \leq \sum_{i<j} \mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{D}_x}[\widehat{g}_{i,j}(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x})]$. Hence, we have reduced the problem of bounding the LHS error to a series of binary sub-problems. At this moment the problem is binary classification, we can use tools for generalization bounds (Boucheron et al., 2005), where the population loss function for the pair (i, j) of the classifier g in the VC class is $L_{i,j}(g) = \mathbf{E}_{(\boldsymbol{x},\sigma)\sim\mathcal{D}_R^{\boldsymbol{q}}}[\mathbf{1}\{g(\boldsymbol{x}) \neq \mathrm{sgn}(\sigma(i) - \sigma(j))\}|\sigma \ni \{i, j\}]$. We can control the incurred population loss of the ERM against the Bayes classifier exploiting Condition 2, similar to Clémençon & Vogel (2020). We show that for a training set T_n with elements $(\boldsymbol{x}, y)$ with $y = \mathrm{sgn}(\sigma(i) - \sigma(j))$ where $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R^{\boldsymbol{q}}$ conditioned that $\sigma \ni \{i, j\}$, it holds that: $L_{i,j}(\widehat{g}_{i,j}) - L_{i,j}(g^\star) \leq 2 \cdot (\inf_{g\in\mathcal{G}} L_{i,j}(g) - L_{i,j}(g^\star)) + r_n$, with high probability, where $\widehat{g}_{i,j} = \mathrm{argmin}_{g\in\mathcal{G}} \widehat{L}_{i,j}(g; T_n)$ and $g^\star$ is the Bayes classifier and $r_n = O(n^{-\frac{1}{2-a}} \cdot \mathrm{VC}(\mathcal{G})^{\frac{1}{2-a}})$, where a is the parameter of the Tsybakov's noise; note that when $a = 1$, we obtain the fast rate $1/n$ and when $a = 0$, we get that standard rate $1/\sqrt{n}$. It remains to aggregate the above $O(k^2)$ binary classifiers into a single one (see Claim 3) in order to get the result of Theorem 2.5. For the sample complexity bound, see Claim 4. For the full proof, see Theorem A.7.

Acknowledgments

This work is supported by NTUA Basic Research Grant (PEBE 2020) "Algorithm Design through Learning Theory: Learning-Augmented and Data-Driven Online Algorithms (LEADAlgo)".

References

Aledo, J. A., Gámez, J. A., and Molina, D. Tackling the supervised label ranking problem by bagging weak learners. *Information Fusion*, 35:38–50, 2017.

Allwein, E. L., Schapire, R. E., and Singer, Y. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. *Journal of machine learning research*, 1(Dec): 113–141, 2000.

Balasubramaniyan, R., Hüllermeier, E., Weskamp, N., and Kämper, J. Clustering of gene expression data using a local shape-based similarity measure. *Bioinformatics*, 21 (7):1069–1077, 2005.

Bartlett, P. L., Bousquet, O., and Mendelson, S. Local Rademacher Complexities. *The Annals of Statistics*, 33 (4):1497–1537, 2005.

Baum, E. B. and Haussler, D. What Size Net Gives Valid Generalization? *Neural computation*, 1(1):151–160, 1989.

Biau, G. Analysis of a Random Forests Model. *The Journal of Machine Learning Research*, 13:1063–1095, 2012.

Boucheron, S., Bousquet, O., and Lugosi, G. Theory of Classification: A Survey of Some Recent Advances. *ESAIM: probability and statistics*, 9:323–375, 2005.

Bousquet, O., Boucheron, S., and Lugosi, G. Introduction to Statistical Learning Theory. In *Summer school on machine learning*, pp. 169–207. Springer, 2003.

Breiman, L. Random Forests. *Machine learning*, 45(1): 5–32, 2001.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. Classification and Regression Trees. *Wadsworth International Group*, 37(15):237–251, 1984.

Cheng, W. and Hüllermeier, E. Instance-Based Label Ranking using the Mallows Model. In *ECCBR Workshops*, pp. 143–157, 2008.

Cheng, W. and Hüllermeier, E. Probability Estimation for Multi-class Classification Based on Label Ranking. In *Joint European Conference on Machine Learning and*

Knowledge Discovery in Databases, pp. 83–98. Springer, 2012.

Cheng, W. and Hüllermeier, E. A Nearest Neighbor Approach to Label Ranking based on Generalized Labelwise Loss Minimization. In *International Joint Conference on Artificial Intelligence (IJCAI-13)*. Citeseer, 2013.

Cheng, W., Hühn, J., and Hüllermeier, E. Decision Tree and Instance-Based Learning for Label Ranking. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 161–168, 2009.

Cheng, W., Dembczynski, K., and Hüllermeier, E. Label Ranking Methods based on the Plackett-Luce Model. In *ICML*, 2010.

Cheng, W., Henzgen, S., and Hüllermeier, E. Labelwise versus Pairwise Decomposition in Label Ranking. In *Lwa*, pp. 129–136, 2013.

Clémençon, S. and Korba, A. On Aggregation in Ranking Median Regression. In *ESANN*, 2018.

Clémençon, S. and Vogel, R. A Multiclass Classification Approach to Label Ranking. In *International Conference on Artificial Intelligence and Statistics*, pp. 1421–1430. PMLR, 2020.

Clémençon, S., Korba, A., and Sibony, E. Ranking Median Regression: Learning to Order through Local Consensus. In *Algorithmic Learning Theory*, pp. 212–245. PMLR, 2018.

Comminges, L. and Dalalyan, A. S. Tight conditions for consistency of variable selection in the context of high dimensionality. *The Annals of Statistics*, 40(5):2667–2696, 2012.

de Sá, C. R., Soares, C., Knobbe, A., and Cortez, P. Label Ranking Forests. *Expert systems*, 34(1):e12166, 2017.

Dekel, O., Singer, Y., and Manning, C. D. Log-Linear Models for Label Ranking. *Advances in neural information processing systems*, 16:497–504, 2003.

Denil, M., Matheson, D., and De Freitas, N. Narrowing the Gap: Random Forests In Theory and In Practice. In *International conference on machine learning*, pp. 665–673. PMLR, 2014.

Dery, L. and Shmueli, E. BoostLR: A Boosting-Based Learning Ensemble for Label Ranking Tasks. *IEEE Access*, 8:176023–176032, 2020.

Djuric, N., Grbovic, M., Radosavljevic, V., Bhamidipati, N., and Vucetic, S. Non-Linear Label Ranking for Large-Scale Prediction of Long-Term User Interests. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

Dudley, R. M. Balls in $\mathbb{R}^k$ do not cut all subsets of $k + 2$ points. *Advances in Mathematics*, 31(3):306–308, 1979.

Fotakis, D., Kalavasis, A., and Stavropoulos, K. Aggregating Incomplete and Noisy Rankings. In *International Conference on Artificial Intelligence and Statistics*, pp. 2278–2286. PMLR, 2021.

Friedman, J., Hastie, T., Tibshirani, R., et al. *The Elements of Statistical Learning*, volume 1. Springer series in statistics New York, 2001.

Friedman, J. H. Multivariate Adaptive Regression Splines. *The annals of statistics*, pp. 1–67, 1991.

Fürnkranz, J. Round Robin Classification. *The Journal of Machine Learning Research*, 2:721–747, 2002.

Geng, X. and Luo, L. Multilabel Ranking with Inconsistent Rankers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3742–3747, 2014.

George, E. I. and McCulloch, R. E. Approaches for Bayesian Variable Selection. *Statistica sinica*, pp. 339–373, 1997.

Grbovic, M., Djuric, N., and Vucetic, S. Learning from Pairwise Preference Data using Gaussian Mixture Model. *Preference Learning: Problems and Applications in AI*, 33, 2012.

Grbovic, M., Djuric, N., Guo, S., and Vucetic, S. Supervised clustering of label ranking data using label preference information. *Machine learning*, 93(2-3):191–225, 2013.

Gurrieri, M., Fortemps, P., and Siebert, X. Alternative Decomposition Techniques for Label Ranking. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pp. 464–474. Springer, 2014.

Har-Peled, S., Roth, D., and Zimak, D. Constraint Classification for Multiclass Classification and Ranking. *Advances in neural information processing systems*, pp. 809–816, 2003.

Hastie, T. and Tibshirani, R. Classification by Pairwise Coupling. *The annals of statistics*, 26(2):451–471, 1998.

Hüllermeier, E., Fürnkranz, J., Cheng, W., and Brinker, K. Label Ranking by Learning Pairwise Preferences. *Artificial Intelligence*, 172(16):1897–1916, 2008. ISSN 0004-3702.

James, G., Witten, D., Hastie, T., and Tibshirani, R. *An Introduction to Statistical Learning*, volume 112. Springer, 2013.

Jindal, R. and Taneja, S. Ranking in multi label classification of text documents using quantifiers. In *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, pp. 162–166. IEEE, 2015.

Karpinski, M. and Macintyre, A. Polynomial Bounds for VC Dimension of Sigmoidal and General Pfaffian Neural Networks. *Journal of Computer and System Sciences*, 54 (1):169–176, 1997.

Korba, A., Clémençon, S., and Sibony, E. A Learning Theory of Ranking Aggregation. In *Artificial Intelligence and Statistics*, pp. 1001–1010. PMLR, 2017.

Korba, A., Garcia, A., and d'Alché-Buc, F. A Structured Prediction Approach for Label Ranking. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Lafferty, J. and Wasserman, L. Rodeo: Sparse, greedy nonparametric regression. *The Annals of Statistics*, 36(1): 28–63, 2008.

Laurent, H. and Rivest, R. L. Constructing Optimal Binary Decision Trees is NP-Complete. *Information processing letters*, 5(1):15–17, 1976.

Liu, H. and Chen, X. Nonparametric Greedy Algorithms for the Sparse Learning Problem. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, pp. 1141–1149, 2009.

Loh, W.-Y. Classification and Regression Trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

Louppe, G. Understanding Random Forests: From Theory to Practice. *arXiv preprint arXiv:1407.7502*, 2014.

Lugosi, G. and Nobel, A. Consistency of Data-driven Histogram Methods for Density Estimation and Classification. *The Annals of Statistics*, 24(2):687–706, 1996.

Mallows, C. L. Non-Null Ranking Models. I. *Biometrika*, 44(1/2):114–130, 1957.

Mansour, Y. and McAllester, D. A. Generalization Bounds for Decision Trees. In *COLT*, pp. 69–74. Citeseer, 2000.

Massart, P. and Nédélec, É. Risk bounds for statistical learning. *The Annals of Statistics*, 34(5):2326–2366, 2006.

Moreira, M. and Mayoraz, E. Improved Pairwise Coupling Classification with Correcting Classifiers. In *European conference on machine learning*, pp. 160–171. Springer, 1998.

Nobel, A. Histogram Regression Estimation Using Data-Dependent Partitions. *The Annals of Statistics*, 24(3): 1084–1105, 1996.

Scornet, E., Biau, G., and Vert, J.-P. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741, 2015.

Shalev-Shwartz, S. *Online Learning: Theory, Algorithms, and Applications*. PhD thesis, The Hebrew University of Jerusalem, 2007.

Shalev-Shwartz, S. and Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge university press, 2014.

Smola, A. J. and Bartlett, P. L. Sparse Greedy Gaussian Process Regression. In *Advances in neural information processing systems*, pp. 619–625, 2001.

Syrgkanis, V. and Zampetakis, M. Estimation and Inference with Trees and Forests in High Dimensions. In *Conference on Learning Theory*, pp. 3453–3454. PMLR, 2020.

Tsybakov, A. B. *Introduction to Nonparametric Estimation*. Springer Publishing Company, Incorporated, 1st edition, 2008. ISBN 0387790519.

Vembu, S. and Gärtner, T. Label Ranking Algorithms: A Survey. In *Preference learning*, pp. 45–64. Springer, 2010.

Wager, S. and Athey, S. Estimation and Inference of Heterogeneous Treatment Effects using Random Forests. *Journal of the American Statistical Association*, 113(523): 1228–1242, 2018.

Wang, Q., Wu, O., Hu, W., Yang, J., and Li, W. Ranking social emotions by learning listwise preference. In *The First Asian Conference on Pattern Recognition*, pp. 164–168. IEEE, 2011.

Wu, T.-F., Lin, C.-J., and Weng, R. C. Probability Estimates for Multi-Class Classification by Pairwise Coupling. *Journal of Machine Learning Research*, 5(Aug): 975–1005, 2004.

Yang, Y. and Tokdar, S. T. Minimax-optimal nonparametric regression in high dimensions. *The Annals of Statistics*, 43(2):652–674, 2015.

Zhou, Y. and Qiu, G. Random Forest for Label Ranking. *Expert Systems with Applications*, 112:99–109, 2018.

Zhou, Y., Liu, Y., Gao, X.-Z., and Qiu, G. A label ranking method based on Gaussian mixture model. *Knowledge-Based Systems*, 72:108–113, 2014a.

Zhou, Y., Liu, Y., Yang, J., He, X., and Liu, L. A Taxonomy of Label Ranking Algorithms. *J. Comput.*, 9(3):557–565, 2014b.

A. Main Theoretical Results: Statements and Proofs

In this section, we provide our results formally: In particular, Appendix A.1 contains our results for LR with complete rankings and decision trees and A.2 contains our results for LR with complete rankings and random forests. Finally, in the Appendix A.3, our results for noisy LR with incomplete rankings are provided.

A.1. Noiseless Oracle with Complete Rankings and Decision Trees (Level Splits & Breiman)

A.1.1. Definition of Properties for Decision Trees

In this section, we study the Label Ranking problem in the complete rankings' setting. We first define some properties required in order to state our results. In the high-dimensional regime, we assume that the target function is sparse. We remind the reader the following standard definition of sparsity of a real-valued Boolean function.

Definition A.1 (Sparsity)**.** We say that the target function $f : \{0,1\}^d \to \mathbb{R}$ is r-sparse if and only if there exists a set $R \subseteq [d]$ with $|R| = r$ and a function $h : \{0,1\}^r \to \mathbb{R}$ such that, for every $\boldsymbol{z} \in \{0,1\}^d$, it holds that $f(\boldsymbol{z}) = h(\boldsymbol{z}_R)$. The set R is called the set of relevant features. Moreover, a vector-valued function $\boldsymbol{m} : \{0,1\}^d \to \mathbb{R}^k$ is said to be $\boldsymbol{r}$-sparse if each coordinate $m_j : \{0,1\}^d \to \mathbb{R}$ is r-sparse.[1]

For intuition about the upcoming Condition 3 and Condition 4, we refer the reader to the Appendix D.3 and Appendix D.4 respectively (and also the work of Syrgkanis & Zampetakis, 2020). Let us define the function $\widetilde{V}$ for a set $S \subseteq [d]$, given a function f and a distribution over features $\mathcal{D}_x$:

$$\widetilde{V}(S) := \mathop{\mathbf{E}}_{\boldsymbol{z}_S \sim \mathcal{D}_{x,S}} \left[\left(\mathop{\mathbf{E}}_{\boldsymbol{w} \sim \mathcal{D}_x} [f(\boldsymbol{w}) | \boldsymbol{w}_S = \boldsymbol{z}_S] \right)^2 \right], \tag{4}$$

where $\mathcal{D}_{x,S}$ is the marginal distribution $\mathcal{D}_x$ conditioned on the index set S. The function $\widetilde{V}$ can be seen as a measure of heterogeneity of the within-leaf mean values of the target function f, from the leafs created by the split S, as mentioned by Syrgkanis & Zampetakis (2020).

Condition 3 (Approximate Submodularity). Let $C \geq 1$. We say that the function $\widetilde{V}$ with respect to f (Equation (4)) is C-approximate submodular if and only if for any $T, S \subseteq [d]$, such that $S \subseteq T$ and any $i \in [d]$, it holds that

$$\widetilde{V}(T \cup \{i\}) - \widetilde{V}(T) \leq C \cdot (\widetilde{V}(S \cup \{i\})) - \widetilde{V}(S)).$$

Moreover, a vector-valued function $\boldsymbol{m} : \{0,1\}^d \to \mathbb{R}^k$ is said to be $\boldsymbol{C}$-approximate submodular if the function $\widetilde{V}$ with respect to each coordinate $m_j : \{0,1\}^d \to \mathbb{R}$ of $\boldsymbol{m}$ is C-approximate submodular.

The above condition will be used (and is necessary) in algorithms that use the Level Splits criterion. Let us also set

$$\widetilde{V}_\ell(A, \mathcal{P}) = \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{D}_x} \left[\left(\mathop{\mathbf{E}}_{\boldsymbol{z} \sim \mathcal{D}_x} [f(\boldsymbol{z}) | \boldsymbol{z} \in \mathcal{P}(\boldsymbol{x})] \right)^2 \middle| \boldsymbol{x} \in A \right], \tag{5}$$

where $\mathcal{P}$ is a partition of the hypercube, $\mathcal{P}(\boldsymbol{x})$ is the cell of the partition in which $\boldsymbol{x}$ lies and A is a cell of the partition. The next condition is the analogue of Condition 3 for algorithms that use the Breiman's criterion.

Condition 4 (Approximate Diminishing Returns). For $C \geq 1$, we say that the function $\widetilde{V}_\ell$ with respect to f (Equation (5)) has the C-approximate diminishing returns property if for any cells A, A', any $i \in [d]$ and any $T \subseteq [d]$ such that $A' \subseteq A$, it holds that

$$\widetilde{V}_\ell(A', T \cup \{i\}) - \widetilde{V}_\ell(A', T) \leq C \cdot (\widetilde{V}_\ell(A, i) - \widetilde{V}_\ell(A)).$$

Moreover, a vector-valued function $\boldsymbol{m} : \{0,1\}^d \to \mathbb{R}^k$ is said to have the $\boldsymbol{C}$-approximate diminishing returns property if the function $\widetilde{V}_\ell$ with respect to each coordinate $m_j : \{0,1\}^d \to \mathbb{R}$ of $\boldsymbol{m}$ has the C-approximate diminishing returns property.

A.1.2. The Score Problem

Having provided a list of conditions that will be useful in our theorems, we are now ready to provide our key results. In order to resolve Problem 1, we consider the following crucial problem. The solution of this problem will be used as a black-box in order to address Problem 1. We consider the following general setting.

[1]Note that each coordinate function m_i can be sparse in a different set of indices.

Definition A.2 (Score Generative Process)**.** Consider an underlying score hypothesis $\boldsymbol{m} : \mathbb{X} \to [1/4, 3/4]^k$ and let $\mathcal{D}_x$ be a distribution over features. Each sample is generated as follows:

1. Draw $\boldsymbol{x} \in \mathbb{X}$ from $\mathcal{D}_x$.
2. Draw $\boldsymbol{\xi} \in [-1/4, 1/4]^k$ from the zero mean noise distribution $\mathcal{E}$.
3. Compute the score $\boldsymbol{y} = \boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi}$.
4. Output $(\boldsymbol{x}, \boldsymbol{y})$.

We let $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$.

Under the *score generative process* of Definition A.2, the following problem arises.

Problem 3 (Score Learning)*.* Consider the **score** generative process of Definition A.2 with underlying score hypothesis $\boldsymbol{m} : \mathbb{X} \to [1/4, 3/4]^k$, that outputs samples of the form $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$. The learner is given i.i.d. samples from $\mathcal{D}$ and its goal is to *efficiently* output a hypothesis $\widehat{\boldsymbol{m}} : \mathbb{X} \to \mathbb{R}^k$ such that with high probability the error $\mathbf{E}_{\boldsymbol{x} \sim \mathcal{D}_x}[\|\widehat{\boldsymbol{m}}(\boldsymbol{x}) - \boldsymbol{m}(\boldsymbol{x})\|_2^2]$ is small.

In order to solve Problem 3, we adopt the techniques and the results of Syrgkanis & Zampetakis (2020) concerning efficient algorithms based on decision trees and random forests (for an exposition of the framework, we refer the reader to Appendix D). We provide the following vector-valued analogue of the results of Syrgkanis & Zampetakis (2020), which resolves Problem 3. Specifically, we can control the expected squared L_2 norm of the error between our estimate and the true score vector $\boldsymbol{m}$.

Theorem A.3 (Score Learning with Decision Trees)**.** *For any $\varepsilon, \delta > 0$, under the score generative process of Definition A.2 with underlying score hypothesis $\boldsymbol{m} : \{0, 1\}^d \to [0, 1]^k$ and given i.i.d. data $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$, the following hold:*

1. *There exists an algorithm (Decision Trees via Level-Splits - Algorithm 3) with set of splits S_n that computes a score estimate $\boldsymbol{m}^{(n)}$ which satisfies*

$$\Pr_{(\boldsymbol{x}^1,\boldsymbol{y}^1),\ldots,(\boldsymbol{x}^n,\boldsymbol{y}^n)\sim\mathcal{D}^n}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left\|\boldsymbol{m}(\boldsymbol{x}) - \boldsymbol{m}^{(n)}(\boldsymbol{x}; S_n)\right\|_2^2\right] > \varepsilon\right] \leq \delta\,,$$

and for the number of samples n and the number of splits $\log(t)$, we have that:

   (a) *If $\boldsymbol{m}$ is $\boldsymbol{r}$-sparse as per Definition A.1 and under the $\boldsymbol{C}$-submodularity condition (m_i and $\mathcal{D}_x$ satisfy Condition 3 for each alternative $i \in [k]$), it suffices to draw*

$$n = \widetilde{O}\left(\log(dk/\delta) \cdot k^{Cr+2} \cdot (Cr/\varepsilon)^{Cr+2}\right)$$

   samples and set the number of splits to be $\log(t) = \frac{Cr}{Cr+2}(\log(n) - \log(\log(d/\delta)))$.

   (b) *If, additionally to 1.(a), the marginal over the feature vectors $\mathcal{D}_x$ is a Boolean product probability distribution, it suffices to draw*

$$n = \widetilde{O}\left(\log(dk/\delta) \cdot 2^r \cdot k^2 \cdot (C/\varepsilon)^2\right)$$

   samples and set the number of splits to be $\log(t) = r$.

2. *There exists an algorithm (Decision Trees via Breiman - Algorithm 4) that computes a score estimate $\boldsymbol{m}^{(n)}$ which satisfies*

$$\Pr_{(\boldsymbol{x}^1,\boldsymbol{y}^1),\ldots,(\boldsymbol{x}^n,\boldsymbol{y}^n)\sim\mathcal{D}^n}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left\|\boldsymbol{m}(\boldsymbol{x}) - \boldsymbol{m}^{(n)}(\boldsymbol{x}; P_n)\right\|_2^2\right] > \varepsilon\right] \leq \delta\,.$$

and for the number of samples n and the number of splits $\log(t)$, we have that:

   (a) *If $\boldsymbol{m}$ is $\boldsymbol{r}$-sparse and under the $\boldsymbol{C}$-approximate diminishing returns condition (m_i and $\mathcal{D}_x$ satisfy Condition 4 for each alternative $i \in [k]$), it suffices to draw*

$$n = \widetilde{O}\left(\log(dk/\delta) \cdot k^{Cr+3} \cdot (Cr/\varepsilon)^{Cr+3}\right)$$

   samples and set $\log(t) \geq \frac{Cr}{Cr+3}(\log(n) - \log(\log(d/\delta)))$.

(b) If, additionally to 2.(a), the distribution $\mathcal{D}_x$ is a Boolean product distribution, it suffices to draw

$$n = \widetilde{O}\left(\log(dk/\delta) \cdot k^3 \cdot C^2 \cdot 2^r/\varepsilon^3\right)$$

samples and set $\log(t) \geq r$.

The running time of the algorithms is $\mathrm{poly}_{C,r}(d, k, 1/\epsilon)$.

Algorithm 3 Level-Splits Algorithm for Score Learning

```
1: Input: Access to i.i.d. examples of the form $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$.
2: Model: $\boldsymbol{y} = \boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi}$ (Definition A.2) with $\boldsymbol{m} : \{0,1\}^d \to [0,1]^k$.
3: Output: An estimate $\boldsymbol{m}^{(n)}(\cdot; S_n)$ that, with probability $1-\delta$, satisfies
       $$\mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{D}_x}\left[\left\|\boldsymbol{m}(\boldsymbol{x}) - \boldsymbol{m}^{(n)}(\boldsymbol{x}; S_n)\right\|_2^2\right] \leq \varepsilon\,.$$
4: LearnScore($\varepsilon, \delta$):
5: Draw $n = \widetilde{\Theta}(\log(dk/\delta)(Crk/\varepsilon)^{O(Cr)})$ samples from $\mathcal{D}$ {Under sparsity and Condition 3.}
6: $D^{(n)} \leftarrow \{\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}\}_{j \in [n]}$
7: output LearnScore-LS($D^{(n)}$)
8: LearnScore-LS ($D^{(n)}$):
9: Set $\log(t) = \Theta(r \log(rk))$
10: Create $k$ datasets $D_i = \{(\boldsymbol{x}^{(j)}, y_i^{(j)})\}_{j \in [n]}$
11: for $i \in [k]$ do
12:     $m_i^{(n)}, S_n^{(i)}$ = LevelSplits-Algo($0, D_i, \log(t)$) {Call Algorithm 9.}
13: endfor
14: Output $\boldsymbol{m}^{(n)}(\cdot; S_n^{(1)}, ..., S_n^{(k)}) = (m_1^{(n)}(\cdot; S_n^{(1)}), ..., m_k^{(n)}(\cdot; S_n^{(k)}))$
```

Algorithm 4 Breiman's Algorithm for Score Learning

```
1: Input: Access to i.i.d. examples of the form $(\boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{D}$.
2: Model: $\boldsymbol{y} = \boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi}$ (Definition A.2) with $\boldsymbol{m} : \{0,1\}^d \to [0,1]^k$.
3: Output: An estimate $\boldsymbol{m}^{(n)}(\cdot; S_n)$ that, with probability $1-\delta$, satisfies
       $$\mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{D}_x}\left[\left\|\boldsymbol{m}(\boldsymbol{x}) - \boldsymbol{m}^{(n)}(\boldsymbol{x}; P_n)\right\|_2^2\right] \leq \varepsilon\,.$$
4: LearnScore ($\varepsilon, \delta$):
5: Draw $n = \widetilde{\Theta}\left(\log(dk/\delta)(Crk/\varepsilon)^{O(Cr)}\right)$ samples from $\mathcal{D}$ {Under sparsity and Condition 4.}
6: $D^{(n)} \leftarrow \{\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}\}_{j \in [n]}$
7: output LearnScore-Breiman($D^{(n)}$)
8: LearnScore-Breiman ($D^{(n)}$):
9: Set $\log(t) = \Theta(r \log(rk))$
10: Create $k$ datasets $D_i = \{(\boldsymbol{x}^{(j)}, y_i^{(j)})\}_{j \in [n]}$
11: for $i \in [k]$ do
12:     $m_i^{(n)}, P_n^{(i)}$ = Breiman-Algo($0, D_i, \log(t)$) {Call Algorithm 10.}
13: endfor
14: Output $\boldsymbol{m}^{(n)}(\cdot; P_n^{(1)}, ..., P_n^{(k)}) = (m_1^{(n)}(\cdot; P_n^{(1)}), ..., m_k^{(n)}(\cdot; P_n^{(k)}))$
```

Proof. (of Theorem A.3) Let us set $J = [1/4, 3/4]$ and let $\boldsymbol{m} : \{0,1\}^d \to J^k$ be the underlying score vector hypothesis and consider a training set with n samples of the form $(\boldsymbol{x}, \boldsymbol{y}) \in \{0,1\}^d \times J^k$ with law $\mathcal{D}$, generated as in Definition A.2.

We decompose the mapping as $m(\boldsymbol{x}) = (m_1(\boldsymbol{x}), \dots, m_k(\boldsymbol{x}))$ and aim to learn each function $m_i : \{0,1\}^d \to J$ separately. Note that since $\boldsymbol{m}$ is $\boldsymbol{r}$-sparse, then any m_i is r-sparse for any $i \in [k]$. We observe that each sample of Definition A.2 can be equivalently generated as follows:

1. $\boldsymbol{x} \in \{0,1\}^d$ is drawn from $\mathcal{D}_x$,
2. For each $i \in [k]$:
   (a) Draw $\xi \in [-1/4, 1/4]$ from the zero mean distribution marginal $\mathcal{E}_i$.
   (b) Compute $y_i = m_i(\boldsymbol{x}) + \xi$.
3. Output $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{y} = (y_i)_{i \in [k]}$.

In order to estimate the coordinate $i \in [k]$, i.e., the function $m_i : \{0,1\}^d \to J$, we have to make use of the samples $(\boldsymbol{x}, y_i) \in \{0,1\}^d \times [0,1]$. We have that

$$\mathbf{Pr}\left[\mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{D}_x}\left[\left\|\boldsymbol{m}(\boldsymbol{x}) - \boldsymbol{m}^{(n)}(\boldsymbol{x}; S_n)\right\|_2^2\right] > \varepsilon\right] = \mathbf{Pr}\left[\sum_{i \in [k]} \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{D}_x}\left[(m_i(\boldsymbol{x}) - m_i^{(n)}(\boldsymbol{x}; S_n))^2\right] > \varepsilon\right]$$
$$\leq \mathbf{Pr}[\exists i \in [k] : B_i]\,,$$

where we consider the events

$$B_i = \mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{D}_x}\left[\left(m_i(\boldsymbol{x}) - m_i^{(n)}(\boldsymbol{x}; S_i^n)\right)^2\right] > \varepsilon / k\,,$$

for any $i \in [k]$, whose randomness lies in the random variables used to construct the empirical estimate $m_i^n(\cdot; S_i^n) = m_i^n(\cdot; S_i^n, (\boldsymbol{x}^{(1)}, y_i^{(1)}), \dots, (\boldsymbol{x}^{(n)}, y_i^{(n)}))$. We note that we have to split the dataset with examples $(\boldsymbol{x}, \boldsymbol{y})$ into k datasets $(\boldsymbol{x}, y_i)$ and execute each sub-routine with parameters $(\epsilon/k, \delta/k)$. We now turn to the sample complexity guarantees. Let us begin with the Level-Splits Algorithm.

Case 1a. If each m_i is r-sparse and under the submodularity condition, by Theorem D.2 with $f = m_i$, we have that

$$\mathbf{Pr}[B_i] \leq d \exp(-n/(Cr \cdot k/\varepsilon)^{Cr+2})\,.$$

By the union bound, we have that

$$\mathbf{Pr}[\exists i \in [k] : B_i] \leq \sum_{i \in [k]} \mathbf{Pr}[B_i]\,.$$

In order to make this probability at most δ, it suffices to make the probability of the bad event B_i at most δ/k, and, so, it suffices to draw

$$n = \widetilde{O}(\log(dk/\delta) \cdot (Crk/\varepsilon)^{Cr+2})\,.$$

Case 1b. If each m_i is r-sparse and under the submodularity and the independence of features conditions, by Theorem D.2 with $f = m_i$, we have that

$$\mathbf{Pr}[B_i] \leq d \exp(-n/(2^r (Ck/\varepsilon)^2))\,.$$

By the union bound and in order to make the probability $\mathbf{Pr}[\exists i \in [k] : B_i]$ at most δ, it suffices to draw

$$n = \widetilde{O}\left(\log(dk/\delta) \cdot 2^r \cdot (Ck/\varepsilon)^2\right)\,.$$

Hence, in each one of the above scenarios, we have that

$$\mathbf{Pr}\left[\mathop{\mathbf{E}}_{\boldsymbol{x} \sim \mathcal{D}_x}\left[\left\|\boldsymbol{m}(\boldsymbol{x}) - \boldsymbol{m}^{(n)}(\boldsymbol{x}; S_n^1, ..., S_n^k)\right\|_2^2\right] > \varepsilon\right] \leq \delta\,.$$

We proceed with the Breiman Algorithm.

Case 2a. If each m_i is r-sparse and under the approximate diminishing returns condition (Condition 4), by Theorem D.3 with $f = m_i$, we have that

$$\mathbf{Pr}[B_i] \leq d \exp(-n/(Cr \cdot k/\varepsilon)^{Cr+3})\,.$$

By the union bound, we have that

$$\mathbf{Pr}[\exists i \in [k] : B_i] \leq \sum_{i\in[k]} \mathbf{Pr}[B_i]\,.$$

In order to make this probability at most δ, it suffices to make the probability of the bad event B_i at most δ/k, and, so, it suffices to draw

$$n = \widetilde{O}(\log(dk/\delta) \cdot (Cr \cdot k/\varepsilon)^{Cr+3})\,.$$

Case 2b. If each m_i is r-sparse and under the approximate diminishing returns condition (Condition 4) and the independence of features conditions, by Theorem D.3 with $f = m_i$, we have that

$$\mathbf{Pr}[B_i] \leq d \exp(-n\varepsilon^3/(k^3 \cdot C^2 2^r))\,.$$

By the union bound and in order to make the probability $\mathbf{Pr}[\exists i \in [k] : B_i]$ at most δ, it suffices to draw

$$n = \widetilde{O}(\log(dk/\delta) \cdot k^3 C^2 2^r/\varepsilon^3)\,.$$

Hence, in each one of the above scenarios, we have that

$$\mathbf{Pr}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left\|\boldsymbol{m}(\boldsymbol{x}) - \boldsymbol{m}^{(n)}(\boldsymbol{x}; P_n^1, ..., P_n^k)\right\|_2^2\right] > \varepsilon\right] \leq \delta\,.$$

$\square$

A.1.3. MAIN RESULT FOR NOISELESS LR WITH DECISION TREES

We are now ready to address Problem 1 for the oracle $\mathrm{Ex}(\boldsymbol{m})$. Our main theorem follows. We comment that Theorem 2.1 corresponds to the upcoming case 1(a).

Theorem A.4 (Label Ranking with Decision Trees)**.** *Consider the example oracle* $\mathrm{Ex}(\boldsymbol{m})$ *of Definition 1.1 with underlying score hypothesis* $\boldsymbol{m} : \{0,1\}^d \to [0,1]^k$*, where* $k \in \mathbb{N}$ *is the number of labels. Given i.i.d. data* $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R$*, the following hold for any* $\varepsilon > 0$ *and* $\delta > 0$*:*

1. *There exists an algorithm (Decision Trees via Level-Splits - Algorithm 5) with set of splits* S_n *that computes an estimate* $h^{(n)}(\cdot\,; S_n) : \{0,1\}^d \to \mathbb{S}_k$ *which satisfies*

$$\mathop{\mathbf{Pr}}_{(\boldsymbol{x}^1,\sigma^1),...,(\boldsymbol{x}^n,\sigma^n)\sim\mathcal{D}_R^n}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[d_2(h(\boldsymbol{x}), h^{(n)}(\boldsymbol{x}; S_n))\right] > \epsilon \cdot k^2\right] \leq \delta\,,$$

and for the number of samples n *and the number of splits* $\log(t)$*, we have that:*

   (a) *If* $\boldsymbol{m}$ *is* $\boldsymbol{r}$*-sparse as per Definition A.1 and under the* $\boldsymbol{C}$*-submodularity condition (*m_i *and* $\mathcal{D}_x$ *satisfy Condition 3 for each alternative* $i \in [k]$*), it suffices to draw*

$$n = \widetilde{O}\left(\log(dk/\delta) \cdot k^{Cr+2} \cdot (Cr/\varepsilon)^{Cr+2}\right)$$

   samples and set the number of splits to be $\log(t) = \frac{Cr}{Cr+2}(\log(n) - \log(\log(d/\delta)))$.

   (b) *If, additionally to 1.(a), the distribution* $\mathcal{D}_x$ *is a Boolean product distribution, it suffices to draw*

$$n = \widetilde{O}\left(\log(dk/\delta) \cdot 2^r \cdot k^2 \cdot (C/\varepsilon)^2\right)$$

   samples and set the number of splits to be $\log(t) = r$.

2. *There exists an algorithm (Decision Tress via Breiman - Algorithm 5) with P_n that computes an estimate $h^{(n)}(\cdot\,; P_n) : \{0,1\}^d \to \mathbb{S}_k$ which satisfies*

$$\Pr_{(\boldsymbol{x}^1,\sigma^1),\ldots,(\boldsymbol{x}^n,\sigma^n)\sim\mathcal{D}_R^n}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[d_2(h(\boldsymbol{x}), h^{(n)}(\boldsymbol{x}; P_n))\right] > \epsilon\cdot k^2\right] \le \delta\,,$$

and for the number of samples n and the number of splits $\log(t)$, we have that:

   (a) *If $\boldsymbol{m}$ is $\boldsymbol{r}$-sparse and under the $\boldsymbol{C}$-approximate diminishing returns condition (m_i and $\mathcal{D}_x$ satisfy Condition 4 for each alternative $i \in [k]$), it suffices to draw*

$$n = \widetilde{O}\left(\log(dk/\delta)\cdot k^{Cr+3}\cdot (Cr/\varepsilon)^{C\cdot r+3}\right)$$

   samples and set $\log(t) \ge \frac{Cr}{Cr+3}(\log(n) - \log(\log(d/\delta)))$.

   (b) *If, additionally to 2.(a), the distribution $\mathcal{D}_x$ is a Boolean product distribution, it suffices to draw*

$$n = \widetilde{O}\left(\log(dk/\delta)\cdot k^3\cdot C^2\cdot 2^r/\varepsilon^3\right)$$

   samples and set $\log(t) \ge r$.

The running time of the algorithms is $\mathrm{poly}_{C,r}(d, k, 1/\epsilon)$.

Algorithm 5 Algorithms for Label Ranking with Complete Rankings

```
1:  Input: Access to i.i.d. examples of the form $(\boldsymbol{x},\sigma) \sim \mathcal{D}_R$.
2:  Model: Oracle $\mathrm{Ex}(\boldsymbol{m})$ with $\boldsymbol{m} : \{0,1\}^d \to [0,1]^k$ and $h(\boldsymbol{x}) = \mathrm{argsort}(\boldsymbol{m}(\boldsymbol{x}))$. (Definition 1.1)
3:  Output: An estimate $h^{(n)}(\cdot; S_n)$.
4:  LabelRank($\varepsilon,\delta$):
5:  Level-Splits Case
6:  Draw $n = \widetilde{\Theta}(\log(dk/\delta)(rk/\varepsilon)^r)$ samples from $\mathcal{D}_R$ {Under sparsity and Condition 3.}
7:  For any $j \in [n]$, compute $\boldsymbol{y}^{(j)} \leftarrow \boldsymbol{m}_C(\sigma^{(j)})$ {See Equation (6).}
8:  $D^{(n)} \leftarrow \{\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}\}_{j\in[n]}$
9:  output argsort(LearnScore-LS($D^{(n)}$)) {Call Algorithm 3.}
10: Breiman Case
11: Draw $n = \widetilde{\Theta}\,(\log(dk/\delta)(rk/\varepsilon)^r)$ samples from $\mathcal{D}$ {Under sparsity and Condition 4.}
12: For any $j \in [n]$, compute $\boldsymbol{y}^{(j)} \leftarrow \boldsymbol{m}_C(\sigma^{(j)})$ {See Equation (6).}
13: $D^{(n)} \leftarrow \{\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}\}_{j\in[n]}$
14: output argsort( LearnScore-Breiman($D^{(n)}$)) {Call Algorithm 4.}
```

Proof. (of Theorem A.4) We let the mapping $\boldsymbol{m}_C : \mathbb{S}_k \to [0,1]^k$ be the canonical representation of a ranking, i.e., for $\sigma \in \mathbb{S}_k$, we define

$$\boldsymbol{m}_C(\sigma) = (\sigma(i)/k)_{i\in[k]}\,. \tag{6}$$

We reduce this problem to a score problem: for any sample $(\boldsymbol{x},\sigma) = (\boldsymbol{x}, \mathrm{argsort}(\boldsymbol{m}(\boldsymbol{x}))) \sim \mathcal{D}_R$, we create the tuple $(\boldsymbol{x},\boldsymbol{y}') = (\boldsymbol{x}, \boldsymbol{m}_C(\sigma))$, where $\boldsymbol{m}_C$ is the canonical representation of the ranking σ. Hence, any permutation of length k is mapped to a vector whose entries are integer multiples of $1/k$. Let us fix $\boldsymbol{x} \in \{0,1\}^d$.

So, the tuple $(\boldsymbol{x},\boldsymbol{y}')$ falls under the setting of the score variant of the regression setting of Definition D.1 with regression function equal to $\boldsymbol{m}'$ (which is equal to $\boldsymbol{m}_C \circ \mathrm{argsort} \circ \boldsymbol{m}$) and noise vector $\boldsymbol{\xi}' = 0$, i.e., $\boldsymbol{y}' = \boldsymbol{m}'(\boldsymbol{x}) + \boldsymbol{\xi}'$. Recall that our goal is to use the transformed samples $(\boldsymbol{x},\boldsymbol{y}')$ in order to estimate the true label ranking mapping $h : \mathbb{X} \to \mathbb{S}_k$. Let us set $h'^{(n)}$ be the label ranking estimate using n samples. We will show that $h'^{(n)} = \mathrm{argsort}(\boldsymbol{m}'^{(n)})$ is close to $h' = \mathrm{argsort}(\boldsymbol{m}')$ in Spearman's distance, where $\boldsymbol{m}'^{(n)}$ is the estimation of $\boldsymbol{m}'$ using Theorem A.3. We have that

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left\|\boldsymbol{m}'(\boldsymbol{x}) - \boldsymbol{m}'^{(n)}(\boldsymbol{x}; S_n^1, \ldots, S_n^k)\right\|_2^2 \le \varepsilon$$

with high probability using the vector-valued tools developed in Theorem A.3. By choosing an appropriate method, we obtain each one of the items $1(a), 1(b), 2(a)$ and $2(b)$ (each sample complexity result is in full correspondence with Theorem A.3). Hence, our estimate is, by definition, close to $\boldsymbol{m}_C \circ \mathrm{argsort} \circ \boldsymbol{m}$, i.e.,

$$\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x} \left\| \boldsymbol{m}_C(\mathrm{argsort}(\boldsymbol{m}(\boldsymbol{x}))) - \boldsymbol{m}'^{(n)}(\boldsymbol{x}; S_n^1, ..., S_n^k) \right\|_2^2 \leq \varepsilon\,,$$

thanks to the structure of the samples $(\boldsymbol{x}, \boldsymbol{y}')$. We can convert our estimate $\boldsymbol{m}'^{(n)}$ to a ranking by setting $h' = \mathrm{argsort}(\boldsymbol{m}'^{(n)})$. For any $i \in [k]$, let us set m_i and $\widehat{m}_i$ for the true and the estimation quantities for simplicity; intuitively (without the expectation operator), a gap of order ϵ/k to the estimate $(m_i - \widehat{m}_i)^2$ yields a bound $|m_i - \widehat{m}_i| \leq \sqrt{\varepsilon/k}$. Recall that $m_i = \sigma(i)/k$ and so this implies that, in integer scaling, $|\sigma(i) - k \cdot \widehat{m}_i| \leq O(\sqrt{\varepsilon \cdot k})$. We now have to compute $\widehat{\sigma}(i)$, that is the rounded value of $k \cdot \widehat{m}_i$. When turning the values $k \cdot \widehat{m}_i$ into a ranking, the distortion of the i-th element from the correct value $\sigma(i)$ is at most the number of indices $j \neq i$ that lie inside the estimation radius. So, any term of the Spearman distance is on expectation of order $O(\varepsilon \cdot k)$. This is due to the fact that $\mathbf{E}[|m_i - k \cdot \widehat{m}_i|] \leq \sqrt{\mathbf{E}[(m_i - k \cdot \widehat{m}_i)^2]} = O(\sqrt{\varepsilon \cdot k})$. To conclude, we get that $\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}_x}\, d_2(\mathrm{argsort}(\boldsymbol{m}(\boldsymbol{x}))), h'(\boldsymbol{x})) \leq O(\epsilon) \cdot k^2$. □

A.2. Noiseless Oracle with Complete Rankings and Random Forests (Level Splits & Breiman)

In this section, we provide similar algorithmic results for Fully Grown Honest Forests based on the Level-Splits and the Breiman's criteria.

A.2.1. Definition of Properties for Random Forests

For algorithms that use Random Forests via Level Splits, we need the following condition.

Condition 5 (Strong Sparsity). A target function $f : \{0,1\}^d \to [-1/2, 1/2]$ is (β, r)-strongly sparse if f is r-sparse with relevant features R (see Definition A.1) and the function $\widetilde{V}$ (see Equation (4)) satisfies

$$\widetilde{V}(T \cup \{j\}) - \widetilde{V}(T) + \beta \leq \widetilde{V}(T \cup \{i\}) - \widetilde{V}(T)\,,$$

for all $i \in R, j \in [d] \setminus R$ and $T \subset [d] \setminus \{i\}$. Moreover, a vector-valued function $\boldsymbol{m} : \{0,1\}^d \to [-1/2, 1/2]^k$ is $(\boldsymbol{\beta}, \boldsymbol{r})$-strongly sparse if each m_j is (β, r)-strongly sparse for any $j \in [k]$.

For algorithms that use Random Forests via Breiman's criterion, we need the following condition.

Condition 6 (Marginal Density Lower Bound). We say that the density $\mathcal{D}_x$ is (ζ, q)-lower bounded if, for every set $Q \subset [d]$ with size $|Q| = q$, for every $\boldsymbol{w} \in \{0,1\}^q$, it holds that

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\mathcal{D}_x}[\boldsymbol{x}_Q = \boldsymbol{w}] \geq \zeta/2^q\,.$$

A.2.2. Main Result for Noiseless LR with Random Forests

Our theorem both for Score Learning and Label Ranking for Random Forests with Level-Splits follows.

Theorem A.5 (Label Ranking with Fully Grown Honest Forests via Level-Splits)**.** *Let $\varepsilon, \delta > 0$. Let $H > 0$. Under Definition A.2 with underlying score hypothesis $\boldsymbol{m} : \{0,1\}^d \to [0,1]^k$, where $k \in \mathbb{N}$ is the number of labels and given access to i.i.d. data $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R$, the following hold. For any $i \in [k]$, we have that: let $m_i^{(n,s)}$ be the forest estimator for alternative i that is built with sub-sampling of size s from the training set and where every tree $m_i(\boldsymbol{x}, D_s)$ is built using Algorithm 9, with inputs: $\log(t)$ large enough so that every leaf has two or three samples and $h = 1$. Under the strong sparsity condition (see Condition 5) for any $i \in [k]$, if R is the set of relevant features and for every $\boldsymbol{w} \in \{0,1\}^r$, it holds for the marginal probability that $\mathbf{Pr}_{\boldsymbol{z}\sim\mathcal{D}_x}(\boldsymbol{z}_R = \boldsymbol{w}) \notin (0, \zeta/2^r)$ and if $s = \widetilde{\Theta}(2^r \cdot (\log(dk/\delta)/\beta^2 + \log(k/\delta)/\zeta))$, then it holds that*

$$\mathop{\mathbf{Pr}}_{(\boldsymbol{x}^1,\sigma^1),\ldots,(\boldsymbol{x}^n,\sigma^n)\sim\mathcal{D}_R^n} \left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x} \left[\left\| \boldsymbol{m}(\boldsymbol{x}) - \boldsymbol{m}^{(n,s)}(\boldsymbol{x}) \right\|_2^2 \right] \geq \varepsilon \right] \leq \delta$$

using a training set of size $n = \widetilde{O}\left(\frac{2^r k \log(k/\delta)}{\varepsilon}\left(\frac{\log(d)}{\beta^2} + \frac{1}{\zeta}\right)\right)$. Moreover, under the generative process of $\mathrm{Ex}(\boldsymbol{m})$ of Definition 1.1, it holds that there exists a $\mathrm{poly}(d, k, 1/\varepsilon)$-time algorithm with the same sample complexity that computes an estimate $h^{(n,s)} : \{0,1\}^d \to \mathbb{S}_k$ which satisfies

$$\mathop{\mathbf{Pr}}_{(\boldsymbol{x}^1,\sigma^1),\ldots,(\boldsymbol{x}^n,\sigma^n)\sim\mathcal{D}_R^n} \left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x} \left[d_2(h(\boldsymbol{x}), h^{(n,s)}(\boldsymbol{x})) \right] > \epsilon \cdot k^2 \right] \leq \delta\,.$$

Proof. (of Theorem A.5) Recall that each sample of Definition A.2 can be equivalently generated as follows:

1. $\boldsymbol{x} \in \{0,1\}^d$ is drawn from $\mathcal{D}_x$,
2. For each $i \in [k]$:
   (a) Draw $\xi \in [-1/4, 1/4]$ from the zero mean distribution marginal $\mathcal{E}_i$.
   (b) Compute $y_i = m_i(\boldsymbol{x}) + \xi$.
3. Output $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{y} = (y_i)_{i \in [k]}$.

In order to estimate the coordinate $i \in [k]$, i.e., the function $m_i : \{0,1\}^d \to [1/4, 3/4]$, we have to make use of the samples $(\boldsymbol{x}, y_i) \in \{0,1\}^d \times [0,1]$. We have that

$$\mathbf{Pr}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left\|\boldsymbol{m}(\boldsymbol{x}) - \boldsymbol{m}^{(n,s)}(\boldsymbol{x})\right\|_2^2\right] > \varepsilon\right] = \mathbf{Pr}\left[\sum_{i\in[k]}\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[(m_i(\boldsymbol{x}) - m_i^{(n,s)}(\boldsymbol{x}))^2\right] > \varepsilon\right]$$
$$\leq \mathbf{Pr}[\exists i \in [k] : B_i]\,,$$

where we consider the events

$$B_i = \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left(m_i(\boldsymbol{x}) - m_i^{(n,s)}(\boldsymbol{x})\right)^2\right] > \varepsilon/k\,,$$

for any $i \in [k]$, whose randomness lies in the random variables used to construct the empirical estimate $m_i^{(n,s)} = m_i^{(n,s)}(\cdot; (\boldsymbol{x}^{(1)}, y_i^{(1)}), \ldots, (\boldsymbol{x}^{(n)}, y_i^{(n)}))$, where $m_i^{(n,s)}$ is the forest estimator for the alternative i using subsampling of size s. We are ready to apply the result of Syrgkanis & Zampetakis (2020) for random forest with Level-Splits (Theorem 3.4) with accuracy ε/k and confidence δ/k. Fix $i \in [k]$. Let $m_i^{(n,s)}$ be the forest estimator that is built with sub-sampling of size s from the training set and where every tree $m_i(\boldsymbol{x}, D_s)$ is built using Algorithm 9, with inputs: $\log(t)$ large enough so that every leaf has two or three samples and $h = 1$. Under the strong sparsity condition for m_i (see Condition 5), if R is the set of relevant features and for every $\boldsymbol{w} \in \{0,1\}^r$, it holds for the marginal probability that $\mathbf{Pr}_{\boldsymbol{z}\sim\mathcal{D}_x}(\boldsymbol{z}_R = \boldsymbol{w}) \notin (0, \zeta/2^r)$ and if $s = \widetilde{\Theta}(2^r \cdot (\log(dk/\delta)/\beta^2 + \log(k/\delta)/\zeta))$, then it holds that

$$\mathop{\mathbf{Pr}}_{D_n\sim\mathcal{D}^n}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left(m_i(\boldsymbol{x}) - m_i^{(n,s)}(\boldsymbol{x})\right)^2\right] \geq \varepsilon/k\right] \leq \delta/k\,,$$

using a training set of size $n = \widetilde{O}\left(\frac{2^r k \log(k/\delta)}{\varepsilon}\left(\frac{\log(d)}{\beta^2} + \frac{1}{\zeta}\right)\right)$. Aggregating the k random forests, we get the desired result using the union bound. The Spearman's distance result follows using the canonical vector representation, as in Theorem A.4. □

The result for the Breiman's criterion is the following.

Theorem A.6 (Label Ranking with Fully Grown Honest Forests via Breiman)**.** *Let $\varepsilon, \delta > 0$. Under Definition A.2 with underlying score hypothesis $\boldsymbol{m} : \{0,1\}^d \to [0,1]^k$, where $k \in \mathbb{N}$ is the number of labels and given access to i.i.d. data $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R$, the following hold. Suppose that $\mathcal{D}_x$ is (ζ, r)-lower bounded (see Condition 6). For any $i \in [k]$, let $m_i^{(n,s)}$ be the forest estimator for the i-th alternative that is built with sub-sampling of size s from the training set and where every tree $m_i(\boldsymbol{x}, D_s)$ is built using the Algorithm 10, with inputs: $\log(t)$ large enough so that every leaf has two or three samples, training set D_s and $h = 1$. Then, using $s = \widetilde{\Theta}(\frac{2^r \log(dk/\delta)}{\zeta\beta^2})$ and under Condition 5 for any $i \in [k]$, we have that*

$$\mathop{\mathbf{Pr}}_{(\boldsymbol{x}^1,\sigma^1),\ldots,(\boldsymbol{x}^n,\sigma^n)\sim\mathcal{D}_R^n}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left\|\boldsymbol{m}(\boldsymbol{x}) - \boldsymbol{m}^{(n,s)}(\boldsymbol{x})\right\|_2^2\right] \geq \varepsilon\right] \leq \delta\,,$$

using $n = \widetilde{O}\left(\frac{2^r k \log(dk/\delta)}{\varepsilon\zeta\beta^2}\right)$. Moreover, under the generative process of $\mathrm{Ex}(\boldsymbol{m})$ of Definition 1.1, it holds that there exists a $\mathrm{poly}(d, k, 1/\varepsilon)$-time algorithm with the same sample complexity that computes an estimate $h^{(n,s)} : \{0,1\}^d \to \mathbb{S}_k$ which satisfies

$$\mathop{\mathbf{Pr}}_{(\boldsymbol{x}^1,\sigma^1),\ldots,(\boldsymbol{x}^n,\sigma^n)\sim\mathcal{D}_R^n}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[d_2(h(\boldsymbol{x}), h^{(n,s)}(\boldsymbol{x}))\right] > \epsilon \cdot k^2\right] \leq \delta\,.$$

Proof. (of Theorem A.6) The proof is similar as in Theorem A.5 (by modifying the sample complexity and the value of subsampling s) and is omitted. □

A.3. Noisy Oracle with Incomplete Rankings

In this section, we study the Label Ranking problem with incomplete rankings and focus on Problem 2. In Definition 2.4, we describe how a noisy score vector $\boldsymbol{y}$ and its associated ranking $\mathrm{argsort}(\boldsymbol{y})$ is generated. In order to resolve Problem 2, we consider an One-Versus-One (OVO) approach. In fact, we consider a VC class $\mathcal{G}$ of binary class and our goal is to use the incomplete observations and output a collection of $\binom{k}{2}$ classifiers from $\mathcal{G}$ so that, for a testing example $\boldsymbol{x} \sim \mathcal{D}_x$ with $\boldsymbol{x} \in \mathbb{R}^d$, the estimated ranking $\widehat{\sigma}_x$, based on our selected hypotheses from $\mathcal{G}$, will be close to the optimal one with high probability. We propose the following algorithm (Algorithm 7).

Algorithm 6 Algorithm for Estimation and Aggregation for a VC class

```
1: Input: A collection of training sets $D_{i,j}$ for $1 \leq i < j \leq k$, VC class $\mathcal{G}$.
2: Output: An estimate $\widehat{\boldsymbol{s}} : \mathbb{X} \to \mathbb{N}^k$ for the optimal score vector $\boldsymbol{s}^\star$.
3: EstimateAggregate($D_{i,j}$ for all $i < j, \mathcal{G}$):
4: for $1 \leq i < j \leq k$ do
5:     Find $\widehat{g}_{i,j} = \mathrm{argmin}_{g \in \mathcal{G}} \frac{1}{|D_{i,j}|} \sum_{(\boldsymbol{x},y) \in D_{i,j}} \mathbf{1}\{g(\boldsymbol{x}) \neq y\}$
6: endfor
7: for $1 \leq i \leq k$ do
8:     $\widehat{s}(\boldsymbol{x}; i) = 1 + \sum_{j \neq i} \mathbf{1}\{\widehat{g}_{i,j}(\boldsymbol{x}) = -1\}$ {Due to the Strict Stochastic Transitivity property (see Condition 2).}
9: endfor
10: Break ties randomly
11: output $\widehat{\boldsymbol{s}}(\cdot) = (\widehat{s}(\cdot;1), ..., \widehat{s}(\cdot;k))$
```

Algorithm 7 Algorithm for Label Ranking with Incomplete Rankings

```
1: Input: Sample access to i.i.d. examples of the form $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R^{\boldsymbol{q}}$, VC class $\mathcal{G}$.
2: Model: Incomplete rankings are generated as in Definition 2.4.
3: Output: An estimate $\widehat{\sigma} : \mathbb{R}^d \to \mathbb{S}_k$ of the optimal classifier $\sigma^\star$ that satisfies
       $$\underset{\boldsymbol{x} \sim \mathcal{D}_x}{\mathbf{Pr}} [\widehat{\sigma}(\boldsymbol{x}) \neq \sigma^\star(\boldsymbol{x})] \leq \frac{C_{a,B}}{\phi^2} \cdot \mathrm{OPT}(\mathcal{G}) + \varepsilon \,.$$
4: LabelRankIncomplete($\varepsilon, \delta$):
5: Set $n = \widetilde{\Theta}\left(k^{\frac{4(1-a)}{a}} \max\{\log(k/\delta), \mathrm{VC}(\mathcal{G})\}/\mathrm{poly}_a(\phi \cdot \varepsilon)\right)$ {See Theorem A.7.}
6: Draw a training set $D$ of $n$ independent samples from $\mathcal{D}_R^{\boldsymbol{q}}$
7: For any $i \neq j$, set $D_{i,j} = \emptyset$
8: for $1 \leq i < j \leq k$ do
9:     if $(\boldsymbol{x}, \sigma) \in D$ and $\sigma \ni \{i, j\}$ then
10:        Add $(\boldsymbol{x}, \mathrm{sgn}(\sigma(i) - \sigma(j)))$ to $D_{i,j}$
11:    endif
12: endfor
13: Training Phase: $\widehat{\boldsymbol{s}} \leftarrow$ EstimateAggregate($D_{i,j}$ for all $i < j, \mathcal{G}$) {See Algorithm 6.}
14: Testing Phase: On input $\boldsymbol{x} \in \mathbb{R}^d$, output $\mathrm{argsort}(\widehat{\boldsymbol{s}}(\boldsymbol{x}))$
```

In order to resolve Problem 2 under Condition 2, we will make use of the Kemeny embedding and the OVO approach. Let D be the training set with labeled examples of the form $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R^{\boldsymbol{q}}$, where σ corresponds to an incomplete ranking generated as in Definition 2.4. Our algorithm proceeds as follows:

1. As a first step, for any pair of alternatives $i < j$ with $i, j \in [k]$, we create a dataset $D_{i,j} = \emptyset$.

2. For any $i < j$ and for any feature $\boldsymbol{x} \in D$ whose incomplete ranking σ contains both i and j, we add in the dataset $D_{i,j}$ the example $(\boldsymbol{x}, y) := (\boldsymbol{x}, \mathrm{sgn}(\sigma(i) - \sigma(j)))$.

3. For any $i < j$, we compute the ERM solution (see Algorithm 6) to the binary classification problem $\widehat{g}_{i,j} = \mathrm{argmin}_{g \in \mathcal{G}} \widehat{L}_{i,j}(g)$ where
$$\widehat{L}_{i,j}(g) = \frac{1}{|D_{i,j}|} \sum_{(\boldsymbol{x},y) \in D_{i,j}} \mathbf{1}\{g(\boldsymbol{x}) \neq y\}\,.$$

4. We aggregate the binary classifiers (see Algorithm 6) using the score function:
$$\widehat{s}(\boldsymbol{x}; i) = 1 + \sum_{j \neq i} \mathbf{1}\{\widehat{g}_{i,j}(\boldsymbol{x}) = -1\}\,.$$
The structure of this score function comes from the SST property.

5. Break the possible ties randomly and output the prediction $\mathrm{argsort}(\widehat{s}(\boldsymbol{x}))$.

Let us consider a binary classification problem with labels $-1, +1$. Let the regression function be $\eta(\boldsymbol{x}) = \mathbf{Pr}_{(\boldsymbol{x},y)}[y = +1 | \boldsymbol{x}]$ and define the mapping $g^\star(\boldsymbol{x}) = \mathbf{1}\{\eta(\boldsymbol{x}) \geq 1/2\}$. If the distribution over (x, y) were known, the problem of finding an optimal classifier would be solved by simply outputting the Bayes classifier $g^\star$, since it is known to minimize the misclassication probability $\mathbf{Pr}_{(x,y)}[y \neq g(x)]$ over the collection of all classifiers. In particular, for any $g \in \mathcal{G}$, it holds that $L(g) - L(g^\star) = 2\,\mathbf{E}_{\boldsymbol{x} \sim \mathcal{D}_x}\left[|\eta(\boldsymbol{x}) - 1/2| \cdot \mathbf{1}\{g(\boldsymbol{x}) \neq g^\star(\boldsymbol{x})\}\right]$.

In Problem 2, our goal is to estimate the solution of the ranking median regression problem with respect to the KT distance
$$\sigma^\star = \underset{h : \mathbb{X} \to \mathbb{S}_k}{\mathrm{argmin}} \underset{(\boldsymbol{x},\sigma) \sim \mathcal{D}_R}{\mathbf{E}} [d_{KT}(h(\boldsymbol{x}), \sigma)]\,.$$

When the probabilities $p_{ij}(\boldsymbol{x}) = \mathbf{Pr}[\sigma(i) > \sigma(j) | \boldsymbol{x}]$ satisfy the SST property, the solution is unique almost surely and has a closed form (see (2)). Hence, we can use estimate the $O(k^2)$ binary optimal classifiers in order to estimate it. This is exactly what we will do in Algorithm 7. Our main result is the following.

Theorem A.7 (Label Ranking with Incomplete Rankings). *Let $\varepsilon, \delta \in (0,1)$ and assume that Condition 2 holds, i.e., the Stochastic Transitivity property holds, the Tsybakov's noise condition holds with $a \in (0,1), B > 0$ and the deletion tolerance condition holds for the survival probability vector with parameter $\phi \in (0,1)$. Set $C_{a,B} = B^{1-a}/((1-a)^{1-a} a^a)$ and consider a hypothesis class $\mathcal{G}$ of binary classifiers with finite VC dimension. There exists an algorithm (Algorithm 7) that computes an estimate $\widehat{\sigma} : \mathbb{R}^d \to \mathbb{S}_k$ so that*
$$\underset{\boldsymbol{x} \sim \mathcal{D}_x}{\mathbf{Pr}} [\widehat{\sigma}(\boldsymbol{x}) \neq \sigma^\star(\boldsymbol{x})] \leq \frac{C_{a,B}}{\phi^2} \left(2 \sum_{i<j} \left(\inf_{g \in \mathcal{G}} L_{i,j}(g) - L^\star_{i,j} \right)^a \right) + \varepsilon\,,$$
with probability at least $1 - \delta$, where $\sigma^\star : \mathbb{R}^d \to \mathbb{S}_k$ is the mapping (see Equation (2)) induced by the aggregation of the $\binom{k}{2}$ Bayes classifiers $g^\star_{i,j}$ with loss $L^\star_{i,j}$, using n independent samples from $\mathcal{D}_R^{\boldsymbol{q}}$ (see Definition 2.4), with
$$n = O\left(\frac{C_{a,B}}{\phi^{4-2a} \cdot \binom{k}{2}} \cdot \left(\frac{C_{a,B} \binom{k}{2}}{\varepsilon \cdot \phi} \right)^{\frac{2-a}{a}} \cdot M \right)\,,$$
where
$$M = \max\left\{ \log(k/\delta), \mathrm{VC}(\mathcal{G}) \cdot \log\left(\frac{C_{a,B} \mathrm{VC}(\mathcal{G})}{\phi^{3-2a}} \cdot \left(\frac{C_{a,B}\binom{k}{2}}{\varepsilon \cdot \phi} \right)^{\frac{2-a}{a}} \right) \right\}\,.$$

In Table 1, we present our sample complexity results (concerning Theorem 2.5 (and Theorem A.7)) for various natural candidate VC classes, including halfspaces and neural networks. We let $a \vee b := \max\{a, b\}$.

Table 1. The table depicts the sample complexity for Problem 2 and Theorem A.7 for various concept classes. In the sample complexity column, we set $N_0 = \text{poly}_{a,B}\left(\frac{k}{\phi \cdot \varepsilon}\right)$. The VC dimension bounds for halfspaces and axis-aligned rectangles can be found in Shalev-Shwartz & Ben-David (2014) and the VC dimension of L_2-balls can be found in Dudley (1979). For the Neural Networks cases, M and N are the number of parameters and of neurons respectively and the corresponding VC dimension bounds are from Baum & Haussler (1989) and Karpinski & Macintyre (1997).

Concept Class	VC Dimension	Sample Complexity
Halfspaces in $\mathbb{R}^d$	$d+1$	$N_0 \cdot O(\log(k/\delta) \vee d\log(d))$
Axis-aligned Rectangles in $\mathbb{R}^d$	$2d$	$N_0 \cdot O(\log(k/\delta) \vee d\log(d))$
L_2-balls in $\mathbb{R}^d$	$d+1$	$N_0 \cdot O(\log(k/\delta) \vee d\log(d))$
NN with Sigmoid Activation	$O(M^2N^2)$	$N_0 \cdot O(\log(k/\delta) \vee M^2N^2\log(M \cdot N))$
NN with Sign Activation	$O(M\log(M))$	$N_0 \cdot O(\log(k/\delta) \vee M\log^2(M))$

Remark A.8. We remark that, in the above results for the noisy nonparametric regression, we only focused on the sample complexity of our learning algorithms. Crucially, the runtime depends on the complexity of the Empirical Risk Minimizer and this depends on the selected VC class. Hence, the choice of the VC class involves a trade-off between computational complexity and expressivity/flexibility.

We continue with the proof. In order to obtain fast learning rates for general function classes, the well-known Talagrand's inequality (see Fact 1 and Boucheron et al., 2005) is used combined with an upper bound on the variance of the loss (which is given by the noise condition) and convergence bounds on Rademacher averages (see e.g., Bartlett et al., 2005).

Proof. (of Theorem A.7) We decompose the proof into a series of claims. Consider the binary hypothesis class $\mathcal{G}$ consisting of mappings $g : \mathbb{R}^d \to \{-1, +1\}$ of finite VC dimension. We let $g^\star$ be the Bayes classifier. We consider $\binom{k}{2}$ copies of this class, one for each unordered pair (i, j) and let $\mathcal{G}_{i,j} = \{g_{i,j} : \mathbb{R}^d \to \{-1, +1\}\}$ be the corresponding class. We let $\widehat{g}_{i,j}$ and $g^\star_{i,j}$ be the algorithm's empirical classifier and the Bayes classifier respectively for the pair (i, j).

Claim 1. It holds that

$$\Pr_{\boldsymbol{x}\sim\mathcal{D}_x}[\widehat{\sigma}(\boldsymbol{x}) \neq \sigma^\star(\boldsymbol{x})] \leq \sum_{i<j} \Pr_{\boldsymbol{x}\sim\mathcal{D}_x}[\widehat{g}_{i,j}(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x})]\,.$$

Proof. The following hold due to the SST condition (Condition 2.i), which implies (2). Let $\widehat{g}_{i,j} = \widehat{g}_{i,j}(D_n)$ be the output estimator for the pair (i, j). We have that

$$\bigcap_{i<j}\{\boldsymbol{x} \in \mathbb{R}^d : \widehat{g}_{i,j}(\boldsymbol{x}) = g^\star_{i,j}(\boldsymbol{x})\} \subset \{\boldsymbol{x} \in \mathbb{R}^d : \widehat{\sigma}(\boldsymbol{x}) = \sigma^\star(\boldsymbol{x})\}\,,$$

where $\widehat{\sigma}, \sigma^\star : \mathbb{R}^d \to \mathbb{S}_k$ are the mappings generated by aggregating the estimators $\{\widehat{g}_{i,j}\}, \{g^\star_{i,j}\}$ respectively. Hence, we get that

$$\{\boldsymbol{x} \in \mathbb{R}^d : \widehat{\sigma}(\boldsymbol{x}) \neq \sigma^\star(\boldsymbol{x})\} \subset \bigcup_{i<j}\{\boldsymbol{x} \in \mathbb{R}^d : \widehat{g}_{i,j}(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x})\}\,.$$

So, we have that the desired probability is controlled by

$$\Pr_{\boldsymbol{x}\sim\mathcal{D}_x}[\widehat{\sigma}(\boldsymbol{x}) \neq \sigma^\star(\boldsymbol{x})] \leq \sum_{i<j} \Pr_{\boldsymbol{x}\sim\mathcal{D}_x}[\widehat{g}_{i,j}(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x})]\,,$$

where the above probabilities also depend on the input training set D_n. □

Thanks to the union bound, it suffices to control the error probability of a single binary classifier. Note that the empirical estimator $\widehat{g}_{i,j} : \mathbb{R}^d \to \{-1, 1\}$ is built from a random number of samples $(\boldsymbol{x}, \sigma)$ (those that satisfy $\sigma \ni \{i, j\}$, see also Algorithm 7). Let us fix a pair (i, j) and, for $\sigma \ni \{i, j\}$, we set $y_{i,j} = \text{sgn}(\sigma(i) - \sigma(j))$. For each classifier $g \in \mathcal{G}_{i,j}$, we introduce the risk

$$L_{i,j}(g) = \mathop{\mathbf{E}}_{(\boldsymbol{x},\sigma)\sim\mathcal{D}_R^q}[\mathbf{1}\{g(\boldsymbol{x}) \neq y_{i,j}\}|\sigma \ni \{i, j\}] = \frac{\mathbf{E}_{(\boldsymbol{x},\sigma)\sim\mathcal{D}_R^q}[\mathbf{1}\{g(\boldsymbol{x}) \neq y_{i,j} \cap \sigma \ni \{i, j\}\}]}{\mathbf{E}_{(\boldsymbol{x},\sigma)\sim\mathcal{D}_R^q}[\mathbf{1}\{\sigma \ni \{i, j\}\}]}\,,$$

and the empirical risk that is obtained using n i.i.d. samples

$$\widehat{L}_{i,j}(g) = \frac{\sum_{i\in[n]} \mathbf{1}\{g(\boldsymbol{x}) \neq y_{i,j} \cap \sigma \ni \{i,j\}\}}{\sum_{i\in[n]} \mathbf{1}\{\sigma \ni \{i,j\}\}} \,.$$

We can control these quantities using the following result, which is a modification of a result that appears in Clémençon & Vogel (2020). In our case, the estimator for the pair (i,j) is built from the samples $(\boldsymbol{x},\sigma) \sim \mathcal{D}_R^{\boldsymbol{q}}$ which contain both i and j.

Claim 2. Let $\delta > 0$ and $i \neq j$ with $i,j \in [k]$. Assume that the Tsybakov condition (Condition 2.ii) holds with parameters a, B for the pair (i,j) and that the survival probabilities vector $\boldsymbol{q}$ satisfies Condition 2.iii with parameter $\phi \in (0,1)$. Set $C_{a,B} = \frac{B^{1-a}}{(1-a)^{1-a}a^a}$. Then, for a training set T_n with elements $(\boldsymbol{x},y)$ with $y = \mathrm{sgn}(\sigma(i)-\sigma(j))$ where $(\boldsymbol{x},\sigma) \sim \mathcal{D}_R^{\boldsymbol{q}}$ conditioned that $\sigma \ni \{i,j\}$, it holds that

$$L_{i,j}(\widehat{g}_{i,j}) - L_{i,j}(g^\star) \leq 2 \cdot \left(\inf_{g\in\mathcal{G}} L_{i,j}(g) - L_{i,j}(g^\star) \right) + r_n(\delta)\,,$$

with probability at least $1-\delta$, where $\widehat{g}_{i,j} = \mathrm{argmin}_{g\in\mathcal{G}} \widehat{L}_{i,j}(g; T_n)$ and $g^\star$ is the Bayes classifier and

$$r_n(\delta) = \max\left\{ 2\left(\frac{16 C_{a,B}}{n\cdot\phi^{3-2a}}\right)^{\frac{1}{2-a}} \cdot \left((C\cdot \mathrm{VC}(\mathcal{G})\cdot\log(n))^{\frac{1}{2-a}} + (32\log(2/\delta))^{\frac{1}{2-a}} \right), \frac{C'\cdot\log(2/\delta)}{\phi\cdot n} \right\},$$

where C, C' are constants. Moreover, if the size of the training set T_n is at least

$$n \geq n_{\mathrm{PC}} := \max\left\{ (2/\delta)^{\frac{1}{2C^2\cdot \mathrm{VC}(\mathcal{G})}}, \log(2/\delta)\left(\frac{6^{2-a}\phi^{1-a}}{C_{a,B}}\right)^{\frac{1}{1-a}} \right\},$$

we have that

$$r_n(\delta) = 2\left(\frac{16 C_{a,B}}{n\cdot\phi^{3-2a}}\right)^{\frac{1}{2-a}} \cdot \left((C\cdot \mathrm{VC}(\mathcal{G})\cdot\log(n))^{\frac{1}{2-a}} + (32\log(2/\delta))^{\frac{1}{2-a}} \right).$$

Proof. Let us fix $i \neq j$. Consider the binary class $\mathcal{G} = \mathcal{G}_{i,j}$ and let us set $L_{i,j}^\star = L_{i,j}(g^\star)$, where $g^\star$ is the Bayes classifier. Let us set $y_{i,j} = \mathrm{sgn}(\sigma(i)-\sigma(j)) \in \{-1,+1\}$. Consider the loss function for the classifier $g \in \mathcal{G}$:

$$c_{i,j}(g;\boldsymbol{x},\sigma) = \mathbf{1}\{g(\boldsymbol{x}) \neq y_{i,j} \cap \sigma \ni \{i,j\}\}\,.$$

We introduce the class of loss functions $\mathcal{F}_{i,j}$ associated with $\mathcal{G}_{i,j}$, where

$$\mathcal{F}_{i,j} = \left\{ (\boldsymbol{x},\sigma) \mapsto \mathbf{1}\{\sigma \ni \{i,j\}\} \cdot (c_{i,j}(g;\boldsymbol{x},\sigma) - \mathbf{1}\{g^\star_{i,j}(\boldsymbol{x}) \neq y_{i,j}\}) : g \in \mathcal{G}_{i,j} \right\}.$$

Let $\mathcal{F}_{i,j}^\star$ be the star-hull of $\mathcal{F}_{i,j}$ with $\mathcal{F}_{i,j}^\star = \{a\cdot f : a \in [0,1], f \in \mathcal{F}_{i,j}\}$. For any $f \in \mathcal{F}_{i,j}$, we introduce the function T which controls the variance of the function f as follows: First, we have that

$$\mathbf{Var}(f) \leq \mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\mathcal{D}_x}[g(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x})] \leq \frac{C_{a,B}}{\phi}(L_{i,j}(g) - L_{i,j}^\star)^a\,.$$

We remark that the first inequality follows from the binary structure of f and the second inequality follows from the fact that the Tsybakov's noise condition implies (see Fact 2) that

$$\mathop{\mathbf{Pr}}_{(\boldsymbol{x},\sigma)}[g(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x}) | \sigma \ni \{i,j\}] \leq C_{a,B}(L_{i,j}(g) - L_{i,j}^\star)^a\,,$$

and since $\mathbf{Pr}[\sigma \ni \{i,j\}|\boldsymbol{x}] > \phi$ for all $i \neq j$ and $\boldsymbol{x} \in \mathbb{R}^d$. Next, we can write that

$$\frac{C_{a,B}}{\phi}(L_{i,j}(g) - L_{i,j}^\star)^a = \frac{C_{a,B}}{\phi\,\mathbf{Pr}[\sigma \ni \{i,j\}]^a}(\mathbf{E}\,f)^a =: T^2(f)\,.$$

We will make use of the following result of Boucheron et al. (2005). In order to state this result, we have to define the functions ψ and w, (we also refer the reader to Boucheron et al. (2005) for further intuition on the definition of these crucial functions). We set

$$\psi(r) = \mathbf{E}\, R_n\{f \in \mathcal{F}^\star_{i,j} : T(f) \leq r\}\,,$$

where R_n is the Rademacher average[2] of a subset of the star-hull of the loss class $\mathcal{F}_{i,j}$ whose variance is controlled by r^2 and

$$w(r) = \sup_{f \in \mathcal{F}^\star_{i,j} : \mathbf{E}\, f \leq r} T(f)\,,$$

which captures the largest variance (i.e., the value $\sqrt{\mathbf{Var}(f)}$) among all loss functions f in the star hull of the loss class with bounded expectation.

Theorem A.9 (Theorem 5.8 in (Boucheron et al., 2005))**.** *Consider the class $\mathcal{G}$ of classifiers $g : \mathbb{X} \to \{-1, +1\}$. For any $\delta > 0$, let $r^\star(\delta)$ denote the solution of*

$$r = 4\psi(w(r)) + 2w(r)\sqrt{\frac{2\log(2/\delta)}{n}} + \frac{16\log(2/\delta)}{3n}$$

and $\varepsilon^\star$ the positive solution of the equation $r = \psi(w(r))$. Then, for any $\theta > 0$, with probability at least $1-\delta$, the empirical risk minimizer g_n satisfies

$$L(g_n) - \inf_{g\in\mathcal{G}} L(g) \leq \theta\left(\inf_{g\in\mathcal{G}} L(g) - L(g^\star)\right) + \frac{(1+\theta)^2}{4\theta} r^\star(\delta)\,.$$

In our binary setting with $i \neq j$, the risk $L_{i,j}$ is conditioned on the event $\sigma \ni \{i,j\}$. Hence, with probability at least $1-\delta$, we get that

$$L_{i,j}(g_n) - \inf_{g\in\mathcal{G}} L_{i,j}(g) \leq \theta\left(\inf_{g\in\mathcal{G}} L_{i,j}(g) - L_{i,j}(g^\star)\right) + \frac{(1+\theta)^2}{4\theta}\frac{r^\star(\delta)}{\mathbf{Pr}[\sigma \ni \{i,j\}]}\,,$$

where $\mathcal{G} = \mathcal{G}_{i,j}$. We set $\theta = 1$ and, by adding and subtracting the Bayes error $L^\star_{i,j} = L_{i,j}(g^\star)$ in the left hand side, we obtain

$$L_{i,j}(\widehat{g}_{i,j}) - L^\star_{i,j} \leq 2\left(\inf_{g\in\mathcal{G}} L_{i,j}(g) - L^\star_{i,j}\right) + \frac{r^\star(\delta)}{\mathbf{Pr}[\sigma \ni \{i,j\}]}\,.$$

The result follows by using the fact that $\mathbf{Pr}[\sigma \ni \{i,j\}] = \Omega(\phi)$ and by the analysis of Clémençon & Vogel (2020) for $r^\star(\delta)$ (see the proof of Lemma 14 by Clémençon & Vogel (2020) and replace ε by ϕ). Finally, we set $r_n(\delta) = r^\star(\delta)/\mathbf{Pr}[\sigma \ni \{i,j\}]$. □

Crucially, observe that the above result does not depend on k, since it focuses on the pairwise comparison i, j.

Claim 3. For $C_{a,B}$ and ϕ as defined in Claim 2, it holds that

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\mathcal{D}_x}[\widehat{\sigma}(\boldsymbol{x}) \neq \sigma^\star(\boldsymbol{x})] \leq \frac{C_{a,B}}{\phi}\left(2\sum_{i<j}\left(\inf_{g\in\mathcal{G}} L_{i,j}(g) - L^\star_{i,j}\right)^a + \binom{k}{2} r_n^a\left(\delta/\binom{k}{2}\right)\right)\,,$$

with probability at least $1-\delta$.

Proof. Recall that $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R^{\boldsymbol{q}}$ and fix the training examples given that $\sigma \ni \{i,j\}$. The Tsybakov's noise condition over the marginal over $\boldsymbol{x}$ given that $\sigma \ni \{i,j\}$ implies (see Fact 2) that

$$\mathop{\mathbf{Pr}}_{(\boldsymbol{x},\sigma)}[g_{i,j}(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x}) | \sigma \ni \{i,j\}] \leq C_{a,B}(L_{i,j}(g) - L^\star_{i,j})^a\,.$$

[2]The Rademacher average of a set A is $R_n(A) := \frac{1}{n}\mathbf{E}\sup_{a\in A}|\sum_{i=1}^n \sigma_i a_i|$, where $\sigma_1, ..., \sigma_n$ are independent Rademacher random variables.

We have that

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\mathcal{D}_x}[g_{i,j}(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x}) | \sigma \ni \{i,j\}] = \int_{\mathbb{R}^d} \mathbf{1}\{g_{i,j}(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x})\}\mathcal{D}_x(\boldsymbol{x}|\sigma \ni \{i,j\})d\boldsymbol{x} =$$
$$= \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\frac{\mathcal{D}_x(\boldsymbol{x}|\sigma \ni \{i,j\})}{\mathcal{D}_x(\boldsymbol{x})}\mathbf{1}\{g_{i,j}(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x})\}\right],$$

where $\mathcal{D}_x(\cdot|\sigma \ni \{i,j\})$ is the conditional distribution of $\boldsymbol{x}$ given that the label permutation contains both i and j. Then, using the deletion tolerance property, we can obtain that $\mathcal{D}_x(\boldsymbol{x}|\sigma \ni \{i,j\}) = \Omega(\phi \cdot \mathcal{D}_x(\boldsymbol{x}))$. So, we have that

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}}[\widehat{g}_{i,j}(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x})] \leq \frac{C_{a,B}}{\phi}(L_{i,j}(\widehat{g}_{i,j}) - L^\star_{i,j})^a .$$

Using Claim 2 for the pair $i \neq j$ and Minkowski inequality, we have that

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}}[\widehat{g}_{i,j}(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x})] \leq \frac{C_{a,B}}{\phi}\left(2\left(\inf_{g\in\mathcal{G}} L_{i,j}(g) - L^\star_{i,j}\right)^a + r_n^a(\delta)\right) .$$

Hence, via the union bound, we have that

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\mathcal{D}_x}[\widehat{\sigma}(\boldsymbol{x}) \neq \sigma^\star(\boldsymbol{x})] \leq \frac{C_{a,B}}{\phi}\left(2\sum_{i<j}\left(\inf_{g\in\mathcal{G}} L_{i,j}(g) - L^\star_{i,j}\right)^a + \binom{k}{2} r_n^a\left(\frac{\delta}{\binom{k}{2}}\right)\right),$$

with probability at least $1-\delta$. □

Claim 4. Let $C_{a,B}$ and ϕ as defined in *Claim* 2. For any $\varepsilon > 0$, it suffices to draw

$$n = O\left(\frac{n_{\mathrm{PC}}}{\phi\binom{k}{2}}\right)$$

samples from $\mathcal{D}_R^{\boldsymbol{q}}$ in order to get

$$\frac{C_{a,B}}{\phi}\binom{k}{2} r^a_{n_{\mathrm{PC}}}\left(\delta/\binom{k}{2}\right) \leq \varepsilon .$$

Proof. Note that each pair (i,j) requires a training set of size at least $n_{\mathrm{PC}}(\delta)$ in order to make the failure probability at most $\delta/\binom{k}{2}$. These samples correspond the rankings σ so that $\sigma \ni \{i,j\}$. Hence, the sample complexity of the problem is equal to the number of (incomplete) permutations drawn from $\mathcal{D}_R^{\boldsymbol{q}}$ in order to obtain the desired number of pairwise comparisons. With high probability, the sample complexity is

$$n = O\left(\frac{n_{\mathrm{PC}}}{\binom{k}{2}\min_{i\neq j}\mathbf{Pr}[\sigma \ni \{i,j\}]}\right) = O\left(\frac{n_{\mathrm{PC}}}{\phi\binom{k}{2}}\right),$$

since the random variable that corresponds to the number of pairwise comparisons provided by each sample $(\boldsymbol{x},\sigma) \sim \mathcal{D}_R^{\boldsymbol{q}}$ is at least $\phi \cdot \binom{k}{2}$, with high probability. The desired sample complexity bound requires a number of pairwise comparisons n_{PC} so that

$$\frac{C_{a,B}}{\phi}\binom{k}{2} r^a_{n_{\mathrm{PC}}}\left(\delta/\binom{k}{2}\right) \leq \varepsilon .$$

The number of pairwise comparisons should be

$$r_{n_{\mathrm{PC}}}\left(\delta/\binom{k}{2}\right) \leq \left(\frac{\varepsilon\phi}{C_{a,B}\binom{k}{2}}\right)^{1/a} .$$

Let us set $m = n_{\mathrm{PC}}$. It remains to control the function r_m. Recall that (see Claim 2)

$$r_m(\delta) = \max\left\{2\left(\frac{16C_{a,B}}{m\cdot\phi^{3-2a}}\right)^{\frac{1}{2-a}} \cdot \left((C\cdot\mathrm{VC}(\mathcal{G})\cdot\log(m))^{\frac{1}{2-a}} + (32\log(2/\delta))^{\frac{1}{2-a}}\right), \frac{C'\cdot\log(2/\delta)}{\phi\cdot m}\right\},$$

If the second term is larger in the above maximum operator, we get that

$$\frac{c_1 \cdot \log(k/\delta)}{\phi \cdot m} \leq \left(\frac{\varepsilon \phi}{C_{a,B}\binom{k}{2}} \right)^{1/a} ,$$

and so

$$m \geq \Omega \left(\frac{\log(k/\delta)}{\phi} \cdot \left(\frac{C_{a,B}\binom{k}{2}}{\varepsilon \phi} \right)^{1/a} \right) =: N_1 .$$

On the other side, we get that

$$\left(\frac{c_1 C_{a,B} \cdot \mathrm{VC}(\mathcal{G}) \cdot \log(m)}{m \cdot \phi^{3-2a}} \right)^{\frac{1}{2-a}} + \left(\frac{c_2 C_{a,B} \log(k/\delta)}{m \cdot \phi^{3-2a}} \right)^{\frac{1}{2-a}} \leq \frac{1}{2} \left(\frac{\varepsilon \phi}{C_{a,B}\binom{k}{2}} \right)^{1/a} ,$$

and so we should take the maximum between the terms

$$m \geq \Omega \left(\frac{C_{a,B} \log(k/\delta)}{\phi^{3-2a}} \cdot \left(\frac{C_{a,B}\binom{k}{2}}{\varepsilon \cdot \phi} \right)^{\frac{2-a}{a}} \right) =: N_2 ,$$

and

$$\frac{m}{\log(m)} \geq \Omega \left(\frac{C_{a,B} \mathrm{VC}(\mathcal{G})}{\phi^{3-2a}} \cdot \left(\frac{C_{a,B}\binom{k}{2}}{\varepsilon \cdot \phi} \right)^{\frac{2-a}{a}} \right) =: M_3 .$$

Let $m = e^y$ and set $ye^{-y} = 1/M_3$. So, we have that $-ye^{-y} = -1/M_3$ and hence $-y = W(-1/M_3)$, where W is the Lambert W function. Let N_3 be the value of m that corresponds to the Lambert equation, i.e., $-\log(m) = W(-1/M_3)$ and so $N_3 \approx e^{-W(-1/M_3)}$ with $1/M_3 \in [0, 1/e]$. Hence, the number of samples that suffice to draw from $\mathcal{D}_R^{\boldsymbol{q}}$ is

$$n \geq \frac{1}{\phi\binom{k}{2}} \max\{N_1, N_2, N_3\} = \frac{1}{\phi\binom{k}{2}} \max\{N_2, N_3\} .$$

We remark that, in the third case, it suffices to take

$$m \geq N_3 \approx 2M_3 \log(M_3) = \widetilde{\Omega} \left(\frac{C_{a,B} \mathrm{VC}(\mathcal{G})}{\phi^{3-2a}} \cdot \left(\frac{C_{a,B}\binom{k}{2}}{\varepsilon \cdot \phi} \right)^{\frac{2-a}{a}} \right) ,$$

since $2M \log(M) \geq M \log(2M \log(M))$ for all M sufficiently large. □

These claims complete the proof. □

B. Additional Experimental Results for Noisy Oracle with Complete Rankings

Experimental Setting and Evaluation Metrics. We follow the setting that was proposed by Cheng & Hüllermeier (2008) (it has been used for the empirical evaluation of LR since then). For each data set, we run five repetitions of a ten-fold cross-validation process. Each data set is divided randomly into ten folds five times. For every division, we repeat the following process: every fold is used exactly one time as the validation set, while the other nine are used as the training set (i.e., ten iterations for every repetition of the ten-fold cross-validation process) (see James et al., 2013, p.181). For every test instance, we compute the Kendall tau coefficient between the output and the given ranking. In every iteration, we compute the mean Kendall tau coefficient of all the test instances. Finally, we compute the mean and standard deviation of every iteration's aggregated results. This setting is used for the evaluation of both our synthetic data sets and the LR standard benchmarks.

Algorithm's Implementation. The algorithm's implementation was in Python. We decided to use the version of our algorithms with Breiman's criterion. Therefore there was no need to implement decision trees and random forests from scratch. We used `scikit-learn` implementations. The code for reproducibility of our results can be found in: `https://github.com/pseleni/LR-nonparametric-regression`.

Data sets. The code for the creation of the Synthetic bencmarks, the Synthetic benchmarks and the standard LR benchmarks that were used in the experimental evaluation can be found in: `https://github.com/pseleni/LR-nonparametric-regression`.

Experimental Results for Different Noise Settings. In our noisy nonparametric regression setting, the input noise acts additively to the vector $\boldsymbol{m}(\boldsymbol{x})$ for some input $\boldsymbol{x}$ and the output is computed by permuting the labels. We aim to understand if the proposed algorithms' performance differs in case the added noise is applied directly to the output ranking, by a parameterized noise operator, instead of being added to the vector $\boldsymbol{m}(\boldsymbol{x})$.

Hence, we resort to a popular distance-based probability model introduced by Mallows (Mallows, 1957). The standard Mallows model $\mathcal{M}(\sigma_0, \theta)$ is a two-parameter model supported on $\mathbb{S}_k$ with density function for a permutation σ equal to

$$\mathop{\mathbf{Pr}}_{\sigma \sim \mathcal{M}(\sigma_0,\theta)}[\sigma | \theta, \sigma_0] = \frac{e^{-\theta D(\sigma,\sigma_0)}}{Z_k(\theta, \sigma_0)} \,.$$

The ranking σ_0 is the central ranking (location parameter) and $\theta > 0$ is the spread (or dispersion) parameter. In this dispersion regime, the ranking σ_0 is the mode of the distribution. The probability of any other permutation decays exponentially as the distance from the center permutation increases. The spread parameter controls how fast this occurs. Finally, $Z_k(\theta, \sigma_0) := \sum_{\sigma \in \mathbb{S}_k} \exp(-\theta D(\sigma, \sigma_0))$ is the partition function of the Mallows distribution. In what follows, we work with the KT distance (when $D = d_{KT}$, the partition function only depends on the dispersion θ and k and not on the central ranking i.e., $Z_k = Z_k(\theta)$).

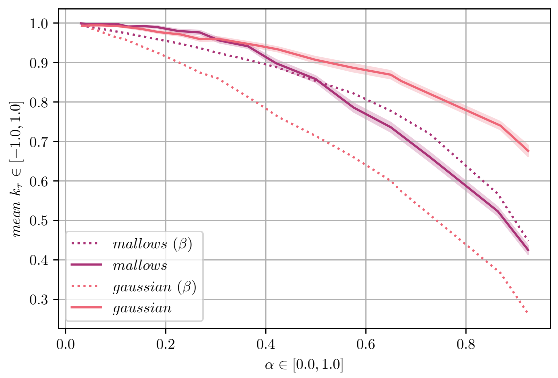
We once again use the noiseless data sets of the two families, namely LFN and SFN. We create 20 noisy data sets for each family using the Mallows Model as a noise operator (we will refer to these data sets as MN - Mallows Noisy). For each noiseless sample, we generate the corresponding output ranking by drawing a permutation from a Mallows distribution with mode equal to the noiseless ranking and dispersion parameter $\theta \in [0.6, 4.4]$[3] .

We also generate 20 additional noisy data sets for each family (following the generative process $\mathrm{Ex}(\boldsymbol{m}, \mathcal{E})$) with different zero mean Gaussian noise distributions (we will refer to these data sets as GN - Gaussian Noisy), matching the α-inconsistency property of the MN data sets. Since the corresponding MN and GN data sets do not also share the β-k_τ property, the results are not directly comparable. Therefore we compare the MN and GN data sets, for each algorithm and data set family separately, with respect to α-inconsistency and using β-k_τ gap as a reference point. The results are obtained again in terms of mean k_τ from five repetitions of a ten-fold cross validation, using the noisy output as training data set and the noiseless output data as validation set, and are visualized in Figure 2. The mean k_τ values are shown as solid lines, the shaded area corresponds to the standard deviation, and the dotted lines reveal the β-k_τ gap.

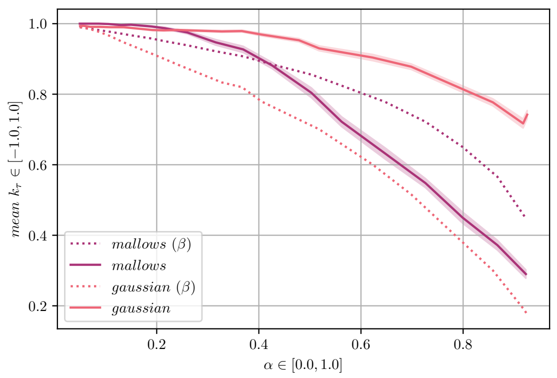
We come to similar conclusions regarding the noise tolerance of each algorithm, e.g., shallow trees and fully grown random forests are noisy tolerant while fully grown decision trees' performance decays almost linearly. What is really interesting, is that in spite of the β-k_τ gap being higher in the MN data sets, i.e., the mean error in the input is smaller, the performance of the algorithms is worse, in comparison to the GN data sets. This reveals that neither α-inconsistency nor β-k_τ gap can be strictly correlated to the ability of the models to interpolate the correct underlying function.

Some comments are in order. The geometry change of the input noise is not the one we should blindly blame for the performance drop. It has to do more with the generative process used for the creation of the data sets. Specifically, in the GN, for every $\boldsymbol{x}$, k samples from a zero mean Gaussian distribution truncated to $[-0.25, 0.25]$ were drawn ($\boldsymbol{\xi} \in [-1/4, 1/4]^k$) and subsequently resulted to a permutation of the elements. The permutation of the elements depends on the noiseless regression values and the additive noise of each label. This results to irregular changes, which in the presence of a big number of samples 'cancel' each other. On the other hand, the MN data sets were created by drawing a sample from a Mallows distribution, which means that certain permutations are more probable for each ranking than others. The noise is no longer unbiased and it is highly likely that the same ranking almost constantly appears as output for a given center permutation. This could also be linked to strategic addition of noise. With these in mind, we can safely conclude that the noise tolerance of our models is directly affected not only by the error in terms of permutation distance, but also, quite naturally, by the frequency of the repetitive errors, i.e., the frequency of the same erroneous noisy output ranking that arises from the noiseless output ranking, as we have to observe the correct permutation at some frequency to achieve meaningful results.

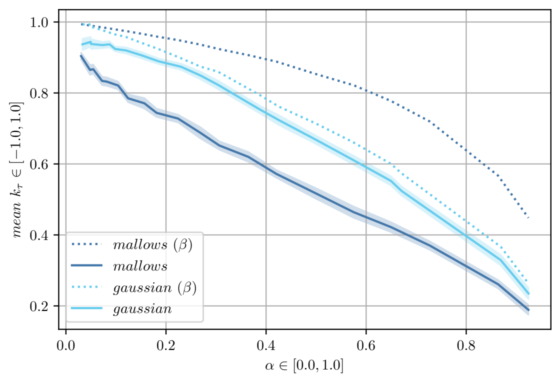
[3]We use a different dispersion parameter for each MN dataset. Specifically we use $\theta \in [0.6, 4.4]$ with 0.2 step.



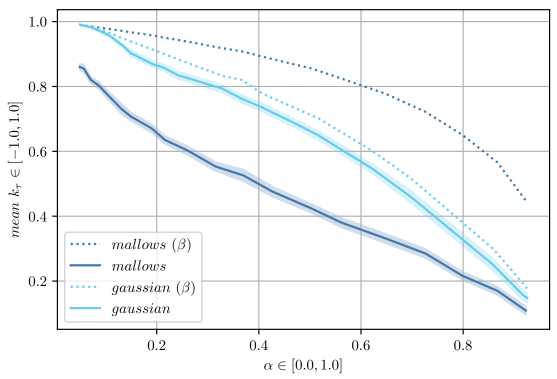
(a) SFN - fully grown random forests



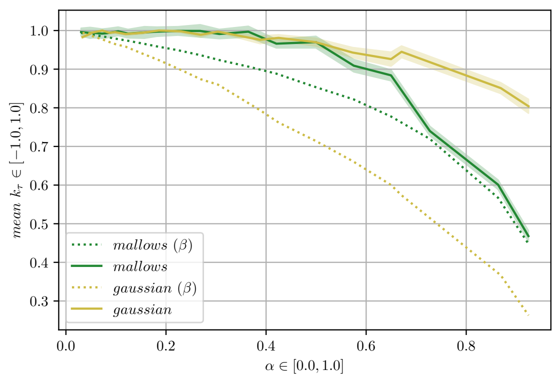
(b) LFN - fully grown random forests



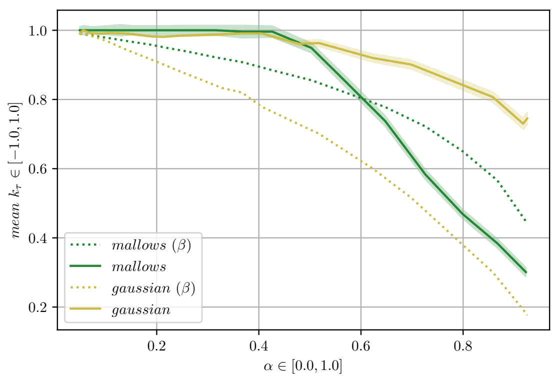
(c) SFN - fully grown decision trees



(d) LFN - fully grown decision trees



(e) SFN - shallow decision trees



(f) LFN - shallow decision trees

Figure 2. Illustration of the experimental results in terms of mean k_τ for different noise operators $\mathcal{E}$ with respect to α-inconsistency and with β-k_τ gap as a reference point. The Gaussian operator gives a ranking $\sigma = \text{argsort}(\boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi})$ with $\boldsymbol{\xi} \in [-1/4, 1/4]^k$ being a zero mean truncated Gaussian random variable. The Mallows operator gives a permutation $\sigma \sim \mathcal{M}(\text{argsort}(\boldsymbol{m}(\boldsymbol{x})), \theta)$ where $\theta \in [0.6, 4.4]$.

Evaluation on LR Standard Benchmarks. We also evaluate our algorithms on standard Label Ranking Benchmarks. Specifically on sixteen semi-synthetic data sets and on five real world LR data sets. The semi-synthetic ones are considered standard benchmarks for the evaluation of LR algorithms, ever since they were proposed in Cheng & Hüllermeier (2008). They were created from the transformation of multi-class (Type A) and regression (Type B) data sets from UCI repository and the Statlog collection into Label Ranking data (see Cheng & Hüllermeier, 2008). A summary of these data sets and their characteristics are given in Table 2. The real-world ones are genetic data, where the genome consists of 2465 genes and each gene is represented by an associated phylogenic profile of length, i.e., 24 features. In these data sets we aim to predict a 'qualitative' representation of an expression profile. The expression profile of a gene is an ordered sequence of real-valued measurements, converted into a ranking (e.g., $(2.1, 3.5, 0.7, -2.5)$ is converted to $(2, 1, 3, 4)$) (Hüllermeier et al., 2008). A summary of the real-world data sets is given in Table 3.

Table 2. Properties of the Semi-Synthetic Benchmarks

BENCHMARK	TYPE	NUMBER OF INSTANCES	NUMBER OF ATTRIBUTES	NUMBER OF LABELS
AUTHORSHIP	A	841	70	4
BODYFAT	B	4522	7	7
CALHOUSING	B	37152	4	4
CPU-SMALL	B	14744	6	5
ELEVATORS	B	29871	9	9
FRIED	B	73376	9	5
GLASS	A	214	9	6
HOUSING	B	906	6	6
IRIS	A	150	4	3
PENDIGITS	A	10992	16	10
SEGMENT	A	2310	18	7
STOCK	B	1710	5	5
VEHICLE	B	846	18	14
VOWEL	A	528	10	11
WINE	A	178	13	3
WISCONSIN	B	346	16	16

Table 3. Properties of the Real-Word Benchmarks

BENCHMARK	NUMBER OF INSTANCES	NUMBER OF FEATURES	NUMBER OF LABELS
SPO	2465	24	11
HEAT	2465	24	6
DDT	2465	24	4
COLD	2465	24	4
DIAU	2465	24	7

We follow the same experimental setting as before, i.e., five repetitions of a ten-fold cross validation process for each data set. In Table 4 we summarize our results. RF stands for the algorithm using Random Forest, DT for Decision Trees, SDT for Shallow Decision Trees. These are the vanilla versions of the proposed algorithms, i.e., the parameters of the regressors were note tuned (only MSE criterion and maximum depth were defined, while the other parameters were the default). RFT and DTT stand for Random Forests Tuned and Decision Trees Tuned (each decision tree was tuned on the training data). The parameters of the regressor were tuned in a five folds inner c.v. for each training set. The parameter grids are reported in the anonymized repository.

Random Forests have the best result overall. Interestingly the tuning does not always lead to better results. In the majority of the benchmarks RF and RFT have comparable results.

In Table 6, we compare our Random Forest results (RF and RFT) with other previously proposed methods, as in Cheng et al. (2013) Labelwise Decomposition (LWD), Hüllermeier et al. (2008) Ranking with Pairwise Comparisons (RPC), Cheng & Hüllermeier (2008) Label Ranking Trees (LRT), Zhou & Qiu (2018) Label Ranking Random Forests (LR-RF), Korba et al. (2018) k-NN Kemeny regressor (kNN Kemeny) and Dery & Shmueli (2020) Boosting-based Learning Ensemble for LR (BoostLR).

For the semi-synthetic data sets, our results are competitive in comparison to Cheng et al. (2013) LWD, Hüllermeier et al. (2008) RPC and Cheng & Hüllermeier (2008) LRT results, but with no systematic improvements. As expected, in comparison to the most recent and state-of-the-art results, the experimental results are comparable but cannot compete the

Table 4. Performance in terms of Kendall's tau coefficient - Semi Synthetic Benchmarks

BENCHMARK	RF	DT	DTS	RFT	DTT
AUTHORSHIP	0.85±0.04	0.78±0.05	0.81±0.05	**0.87±0.04**	0.77±0.05
BODYFAT	**0.12±0.05**	0.05±0.06	0.09±0.07	0.11±0.06	0.07±0.07
CALHOUSING	0.32±0.01	0.24±0.01	0.17±0.02	**0.33±0.01**	0.16±0.03
CPU-SMALL	0.29±0.01	0.21±0.02	0.28±0.01	**0.30±0.02**	0.28±0.01
ELEVATORS	0.60±0.01	0.47±0.01	0.50±0.01	**0.61±0.01**	0.55±0.02
FRIED	**0.96±0.00**	0.90±0.00	0.65±0.01	**0.96±0.00**	0.90±0.00
GLASS	**0.88±0.06**	0.80±0.06	0.79±0.06	0.80±0.07	0.80±0.07
HOUSING	**0.44±0.07**	0.40±0.06	0.39±0.06	0.37±0.07	0.31±0.08
IRIS	**0.95±0.07**	0.91±0.09	0.92±0.08	**0.95±0.07**	0.90±0.10
PENDIGITS	0.86±0.01	0.77±0.018	0.63±0.01	**0.86±0.01**	0.78±0.01
SEGMENT	0.90±0.02	0.87±0.02	0.82±0.02	**0.91±0.02**	0.87±0.02
STOCK	**0.80±0.03**	0.76±0.04	0.76±0.04	0.78±0.03	0.73±0.05
VEHICLE	**0.84±0.03**	0.78±0.04	0.79±0.04	0.83±0.03	0.78±0.04
VOWEL	0.67±0.04	0.63±0.05	0.55±0.04	**0.68±0.04**	0.58±0.06
WINE	0.90±0.09	0.84±0.12	0.86±0.10	**0.91±0.08**	0.84±0.11
WISCONSIN	0.14±0.04	0.08±0.04	0.1±0.04	**0.14±0.05**	0.09±0.04

Table 5. Performance in terms of Kendall's tau coefficient- Real World Benchmarks

BENCHMARK	RF	DT	DTS	RFT	DTT
COLD	**0.10±0.03**	0.06±0.03	0.07±0.03	0.09±0.03	0.07±0.04
DIAU	**0.15±0.03**	0.12±0.02	0.12±0.02	0.14±0.03	0.12±0.04
DTT	**0.13±0.04**	0.10±0.04	0.09±0.04	**0.13±0.03**	0.10±0.03
HEAT	0.07±0.02	0.05±0.02	0.05±0.02	**0.08±0.03**	0.05±0.02
SPO	**0.05±0.02**	0.05±0.02	0.04±0.02	0.01±0.01	0.01±0.01

highly optimized applied algorithms. But we believe that the insights gained using this technique may be valuable to a variate of other Label Ranking methods.

For the real-world data sets there are not so many methods to compare our results with. Therefore we compare them with the RPC method and BoostLR. The results are summarized in Table 7. The performance of our algorithm in the real-word benchmarks is worse than in the other data sets. We suspect that the "non-sparsity" of these data sets (genome data) is one of the main reasons that this pattern in the performance is observed. A more thorough investigation is left for future work.

C. Results on the Noisy Oracle with Partial Rankings

In this setting, we consider a distribution of partitions of the interval (of positive integers) $[1..k]$, which depends on the feature $\boldsymbol{x} \in \mathbb{R}^d$. Before a formal definition, we provide an intuitive example: for some feature $\boldsymbol{x}$, let the noisy score vector be equal to $\boldsymbol{y} = [0.2, 0.4, 0.1, 0.3, 0.5]$ (where $\boldsymbol{y} = \boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi}$, as in Definition 2.4). Then, it holds that $\sigma = \mathrm{argsort}(\boldsymbol{y}) = (e \succ b \succ d \succ a \succ c)$. In the partial setting, we additionally draw an *increasing*[4] partition I of the label space $[k]$ from the distribution $\boldsymbol{p}(\boldsymbol{x})$. Assume that $I = [1...2][3...3][4...5]$, whose size is 3. Then, the partial ranking is defined as $\mathrm{PartialRank}(\sigma; I)$ and is equal to $e = b \succ d \succ a = c$.

Definition C.1 (Generative Process for Partial Data)**.** Consider an instance of the Label Ranking problem with underlying score hypothesis $\boldsymbol{m} : \mathbb{X} \to [0,1]^k$ and let $\mathcal{D}_x$ be a distribution over features. Consider the partial partition distribution $\boldsymbol{p}$. Each sample is generated as follows:

1. Draw $\boldsymbol{x} \in \mathbb{X}$ from $\mathcal{D}_x$ and $\boldsymbol{\xi} \in [-1/4, 1/4]^k$ from the distribution $\mathcal{E}$.
2. Compute $\boldsymbol{y} = \boldsymbol{m}(\boldsymbol{x}) + \boldsymbol{\xi}$.

[4]A partition I of the space $[k]$ is called increasing if there exists an increasing sequence $i_1 < i_2 < ... < i_m$ of indices of $[k]$ so that $I = [1..i_1][i_1..i_2]...[i_m + 1..k]$. For instance, the partition $[1,2][3,4][5]$ is increasing, but the partition $[1,4][2,3][5]$ is not.

Table 6. Evaluation in terms of Kendall's tau coefficient - Semi-Synthetic Benchmarks

BENCHMARK	RF	RFT	LWD	RPC	LRT	LR-RF	kNN KEMENY	BOOSTLR
AUTHORSHIP	0.86±0.04	0.87±0.04	0.91±0.01	0.91	0.88	0.92	0.94±0.02	0.92
BODYFAT	0.12±0.05	0.11±0.06	-	0.28	0.11	0.19	0.23±0.06	0.20
CALHOUSING	0.32±0.01	0.33±0.01	-	0.24	0.36	0.37	0.33±0.01	0.44
CPU-SMALL	0.29±0.01	0.3±0.02	-	0.45	0.42	0.52	0.51±0.00	0.50
ELEVATORS	0.60±0.01	0.61±0.01	-	0.75	0.76	0.76	-	0.77
FRIED	0.96±0.00	0.96±0.00	-	1.00	0.89	1.00	0.89±0.00	0.94
GLASS	0.83±0.06	0.80±0.07	0.88±0.4	0.88	0.88	0.89	0.85±0.06	0.89
HOUSING	0.44±0.07	0.37±0.07	-	0.67	0.80	0.80	-	0.83
IRIS	0.95±0.07	0.95±0.07	0.93±0.06	0.89	0.95	0.97	0.95±0.04	0.83
PENDIGITS	0.86±0.01	0.86±0.01	-	0.93	0.94	0.94	0.94±0.00	0.94
SEGMENT	0.90±0.02	0.91±0.02	0.94±0.01	0.93	0.95	0.96	0.95±0.01	0.96
STOCK	0.80±0.03	0.78±0.03	-	0.78	0.90	0.92	-	0.93
VEHICLE	0.84±0.03	0.83±0.03	0.87±0.02	0.85	0.83	0.86	0.85±0.03	0.86
VOWEL	0.67±0.04	0.68±0.04	0.67±0.02	0.65	0.79	0.97	0.85±0.03	0.84
WINE	0.90±0.09	0.91±0.08	0.91±0.06	0.92	0.88	0.95	0.94±0.06	0.95
WISCONSIN	0.14±0.04	0.14±0.05	-	0.63	0.34	0.48	0.49±0.04	0.45

Table 7. Evaluation in terms of Kendall's tau coefficient - Real Word Benchmarks

BENCHMARK	RF	RFT	RPC	BOOSTLR
SPO	0.05±0.02	0.01±0.01	0.14±0.02	0.14
HEAT	0.07±0.02	0.08±0.03	0.13±0.2	0.13
DTT	0.13±0.04	0.13±0.03	0.17±0.3	0.17
COLD	0.10±0.03	0.09±0.0	0.22±0.03	0.21
DIAU	0.15±0.03	0.14±0.03	0.33±0.02	0.33

3. Set $\widetilde{\sigma} = \mathrm{argsort}(\boldsymbol{y})$.
4. Draw a partition $I = [1...i_1][i_1+1...i_2]...[i_L+1...k]$ of size $L+1$ from a distribution $\boldsymbol{p}(\boldsymbol{x})$, for some $L \in [0..k]$.
5. Set $\sigma = \mathrm{PartialRank}(\widetilde{\sigma}; I)$
6. Output $(\boldsymbol{x}, \sigma)$.

We let $(\boldsymbol{x}, \sigma) \sim \mathcal{D}_R^{\boldsymbol{p}}$.

Note that when $L = 0$, we observe a complete ranking σ. We remark that our generative model is quite general: It allows arbitrarily complex distributions over partitions, which depend on the feature space instances. We aim to address Problem 2 when dealing with partial observations. We are going to address this question in a similar fashion as in the incomplete regime. Specifically, we assume that Condition 2 still holds, but instead of the Item 3 (Deletion tolerance), we assume that the property of *partial tolerance* holds.

Condition 7. For any $1 \leq i < j \leq k$, we assume that the following hold: The Stochastic Transitivity and the Tsybakov's noise condition with parameters a, B (see Condition 2) are satisfied and the following condition holds:

1. (Partial tolerance): There exists $\xi \in (0,1)$ so that $p_{i,j}(\boldsymbol{x}) \geq \xi$, where $p_{i,j}(\boldsymbol{x})$ is the probability that the pair $i < j$ is not in the same subset of the partial partition for $\boldsymbol{x} \in \mathbb{R}^d$.

Using similar technical tools as in Theorem 2.5, we obtain the following result.

Theorem C.2 (Label Ranking with Partial Permutations)**.** *Let $\varepsilon, \delta \in (0,1)$ and assume that Condition 7 holds, i.e., the Stochastic Transitivity property holds and both the Tsybakov's noise condition holds with $a \in (0,1), B > 0$ and the partial tolerance condition holds for the partition probability vector* $\mathbf{p}$ *with parameter $\xi \in (0,1)$. Set $C_{a,B} = B^{1-a}/((1-a)^{1-a}a^a)$*

and consider a hypothesis class $\mathcal{G}$ of binary classifiers with finite VC dimension. There exists an algorithm (Algorithm 8) that computes an estimate $\widehat{\sigma} : \mathbb{R}^d \to \mathbb{S}_k$ so that

$$\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\mathcal{D}_x}[\widehat{\sigma}(\boldsymbol{x}) \neq \sigma^\star(\boldsymbol{x})] \leq \frac{C_{a,B}}{\xi^2}\left(2\sum_{i<j}\left(\inf_{g\in\mathcal{G}} L_{i,j}(g) - L^\star_{i,j}\right)^a\right) + \varepsilon\,,$$

with probability at least $1-\delta$, where $\sigma^\star : \mathbb{R}^d \to \mathbb{S}_k$ is the mapping (see Equation (2)) induced by the aggregation of the $\binom{k}{2}$ Bayes classifiers $g^\star_{i,j}$ with loss $L^\star_{i,j}$, using n independent samples from $\mathcal{D}_R^{\boldsymbol{p}}$ (see Definition C.1), with

$$n = O\left(\frac{C_{a,B}}{\xi^{4-2a}\cdot\binom{k}{2}}\cdot\left(\frac{C_{a,B}\binom{k}{2}}{\varepsilon\cdot\xi}\right)^{\frac{2-a}{a}}\cdot M\right)\,,$$

where

$$M = \max\left\{\log(k/\delta), \mathrm{VC}(\mathcal{G})\cdot\log\left(\frac{C_{a,B}\mathrm{VC}(\mathcal{G})}{\xi^{3-2a}}\cdot\left(\frac{C_{a,B}\binom{k}{2}}{\varepsilon\cdot\xi}\right)^{\frac{2-a}{a}}\right)\right\}.$$

Algorithm 8 Algorithm for Label Ranking with Partial Rankings

```
 1: Input: Sample access to i.i.d. examples of the form $(\boldsymbol{x},\sigma)\sim\mathcal{D}_R^{\boldsymbol{p}}$, VC class $\mathcal{G}$.
 2: Model: Partial rankings are generated as in Definition C.1.
 3: Output: An estimate $\widehat{\sigma}:\mathbb{R}^d\to\mathbb{S}_k$ of the optimal estimator $\sigma^\star$ that satisfies
        $\mathop{\mathbf{Pr}}_{\boldsymbol{x}\sim\mathcal{D}_x}[\widehat{\sigma}(\boldsymbol{x})\neq\sigma^\star(\boldsymbol{x})] \leq \frac{C_{\alpha,B}}{\xi^2}\cdot\mathrm{OPT}(\mathcal{G})+\varepsilon\,.$
 4: LabelRankPartial($\varepsilon,\delta$):
 5: Set $n=\widetilde{\Theta}\left(k^{\frac{4(1-\alpha)}{\alpha}}\max\{\log(k/\delta),\mathrm{VC}(\mathcal{G})\}/\mathrm{poly}_\alpha(\xi\cdot\varepsilon)\right)$ {See Theorem C.2.}
 6: Draw a training set $D$ of $n$ independent samples from $\mathcal{D}_R^{\boldsymbol{p}}$
 7: For any $i\neq j$, set $D_{i,j}=\emptyset$
 8: for $1\leq i<j\leq k$ do
 9:     if $(\boldsymbol{x},\sigma)\in D$ and $\sigma(i)\neq\sigma(j)$ then {$i$ and $j$ do not lie in the same partition.}
10:         Add $(\boldsymbol{x},\mathrm{sgn}(\sigma(i)-\sigma(j)))$ to $D_{i,j}$
11:     endif
12: endfor
13: Training Phase: $\widehat{\boldsymbol{s}}\leftarrow$ EstimateAggregate($D_{i,j}$ for all $i<j,\mathcal{G}$) {See Algorithm 6.}
14: Testing Phase: On input $\boldsymbol{x}\in\mathbb{R}^d$, output argsort($\widehat{\boldsymbol{s}}(\boldsymbol{x})$)
```

Proof. The proof is similar to the incomplete rankings case and the difference lies in the following steps

1. For any $i\neq j$, the conditioning on the event $\sigma\ni\{i,j\}$ should be replaced with the event that i does not lie in the same partition ($\sigma(i)\neq\sigma(j)$). This implies that any ϕ term (which corresponds to a lower bound on the probability $\mathbf{Pr}[\sigma\ni\{i,j\}]$) should be replaced by the term $\xi\in(0,1)$.

2. For the final sample complexity, the following holds: With high probability, the number of pairwise comparisons induced by a partial ranking is at least $\xi\cdot\binom{k}{2}$. Hence, with high probability, a number of $n_{\mathrm{PC}}/(\xi\cdot\binom{k}{2})$ independent draws from the distribution $\mathcal{D}_R^{\boldsymbol{p}}$ suffices to obtain the desired bound.

$\square$

D. Background on Regression with Trees and Forests

In this section, we provide a discussion on decision trees and forests. We refer to Shalev-Shwartz & Ben-David (2014), Syrgkanis & Zampetakis (2020) and Breiman et al. (1984) for further details.

D.1. Further Previous Work

From an information-theoretic viewpoint, sample complexity bounds of decision trees and other data-adaptive partitioning estimators have been established (Nobel, 1996; Lugosi & Nobel, 1996; Mansour & McAllester, 2000). However, from a computational aspect, the problem of choosing the optimal tree is NP-complete (e.g., Laurent & Rivest, 1976) and, hence, from a practical standpoint, trees and forests are built greedily (e.g., Level-Splits, Breiman) by identifying the most empirically informative split at each iteration (Breiman et al., 1984; Breiman, 2001). Advances have shown that such greedily constructed trees are asymptotically consistent (Biau, 2012; Denil et al., 2014; Scornet et al., 2015) in the low-dimensional regime. On the other side, the high-dimensional regime, where the number of features can grow exponentially with the number of samples, is studied by Syrgkanis & Zampetakis (2020). The literature related to the CART criterion (Breiman et al., 1984) is vast; however, there are various other strands of research dealing with the problem of sparse nonparametric regression (that we consider in the noiseless regression settings of our work). On the one side, several heuristic methods have been proposed (Friedman et al., 2001; Friedman, 1991; George & McCulloch, 1997; Smola & Bartlett, 2001). On the other side, various works, such as the ones of Lafferty & Wasserman (2008); Liu & Chen (2009); Comminges & Dalalyan (2012); Yang & Tokdar (2015), design and theoretically analyze greedy algorithmic approaches that exploit the sparsity of the regression function in order to get around with the curse of dimensionality of the input feature data.

D.2. Preliminaries on Regression Trees

Decision Trees. A decision tree is a predictor $h : \mathbb{X} \to \mathcal{Y}$, which, on the input feature $\boldsymbol{x}$, predicts the label associated with the instance by following a decision path from a root node of a tree to a leaf. At each node on the root-to-leaf path, the successor child is chosen on the basis of a splitting of the input space. The splitting may be based on a specific feature of $\boldsymbol{x}$ or on a predefined set of splitting rules and a leaf always contains a specific label or value, depending on the context (classification or regression).

Nonparametric Regression. In the nonparametric regression problem, we consider that we observe independent samples $(\boldsymbol{x}, y)$, generated as $y = f(\boldsymbol{x}) + \xi$, where ξ corresponds to bounded zero mean noise. Specifically, we have the following generative process:

Definition D.1 (Standard Nonparametric Regression)**.** Consider the underlying regression function $f : \mathbb{X} \to [1/4, 3/4]$ and let $\mathcal{D}_x$ be a distribution over features. Each sample is generated as follows:

1. Draw $\boldsymbol{x} \in \mathbb{X}$ from $\mathcal{D}_x$.
2. Draw $\xi \in [-1/4, 1/4]$ from the zero mean distribution $\mathcal{E}$.
3. Compute $y = f(\boldsymbol{x}) + \xi$.

We let $(\boldsymbol{x}, y) \sim \mathcal{D}$.

Note that the noise random variable does not depend on the feature $\boldsymbol{x}$. In the high-dimensional regime, we assume that the target function is sparse (recall Definition A.1).

Regression Tree Algorithms. Let us briefly describe how regression tree algorithms work in an abstract fashion. We will focus on binary trees. In general, the regression tree algorithms operate in two phases, which are the following: first the algorithm finds a partition $\mathcal{P}$ of the hypercube $\{0, 1\}^d$. Afterwards, it assigns a single value to every cell of the partition $\mathcal{P}$, which defines the estimation function $f^{(n)}$. Finally, the algorithm outputs this estimate. More concretely, we have that:

Phase 1 (Partitioning the space). First, a depth 0 tree contains a single cell $\{\{0, 1\}^d\}$. If this single node splits based on whether $x_1 = 0$ or $x_1 = 1$, we obtain a depth 1 tree with two cells $\{\{0\} \times \{0, 1\}^{d-1}, \{1\} \times \{0, 1\}^{d-1}\}$. In general, this procedure generates a partition $\mathcal{P}$ of the space $\{0, 1\}^d$. We let $\mathcal{P}(\boldsymbol{x})$ denote the unique cell of $\mathcal{P}$ that contains $\boldsymbol{x}$.

Phase 2 (Computing the estimation). Let D denote the training set that contains examples of the form $(\boldsymbol{x}, y)$, generated as $y = f(\boldsymbol{x}) + \xi$. For any cell $c \in \mathcal{P}$, we create the dataset D_c of all the training examples $(\boldsymbol{x}, y) \in D$ that are contained in the cell c, i.e., $\boldsymbol{x} \in c$. Then, we compute the value of the cell as

$$f^{(n)}(c; \mathcal{P}) := \frac{1}{|D_c|} \sum_{(\boldsymbol{x},y)\in D_c} y \,.$$

The main question not covered in the above discussion is the following:

How is the split of Phase 1 chosen?

There are various splitting rules in order to partition the space in Phase 1. We discuss two such rules: the Breiman's Algorithm and the Level-Splits Algorithm. In Breiman's algorithm, every node can choose a different direction to split, by using the following greedy criterion: every node chooses the direction that minimizes its own empirical mean squared error. On the other side, in the Level-Splits algorithm, every node at the same level has to split in the same direction, by using the greedy criterion: at every level, we choose the direction that minimizes the total empirical mean squared error. In the upcoming sections, we elaborate on the algorithms based on the Level-Splits (Appendix D.3) and Breiman's (Appendix D.4) criterion.

D.3. Level-Splits Algorithm

We define $\boldsymbol{x}_S$ as the sub-vector of $\boldsymbol{x}$, where we observe only the coordinates with indices in $S \subseteq [d]$. Recall that in the Level-Splits algorithm with set of splits S, any level has to split at the same direction. Hence, each level provides a single index to the set S and the size of the set S is the depth of the decision tree. Given a set of splits S, we define the expected mean squared error of S as follows:

$$\widetilde{L}(S) = \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left(f(\boldsymbol{x}) - \mathop{\mathbf{E}}_{\boldsymbol{w}\sim\mathcal{D}_x}[f(\boldsymbol{w})|\boldsymbol{w}_S = \boldsymbol{x}_S]\right)^2\right] = \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[f^2(\boldsymbol{x})\right] - \mathop{\mathbf{E}}_{\boldsymbol{z}_S\sim\mathcal{D}_{x,S}}\left(\mathop{\mathbf{E}}_{\boldsymbol{w}\sim\mathcal{D}_x}[f(\boldsymbol{w})|\boldsymbol{w}_S = \boldsymbol{z}_S]\right)^2 .$$

This function quantifies the population version of the mean squared error between the actual value of f at $\boldsymbol{x}$ and the mean value of f constrained at the cell of $\mathcal{P}$ that contains $\boldsymbol{x}$, i.e., the subspace of $\{0,1\}^d$ that is equal to $\mathcal{P}(\boldsymbol{x})$. Observe that $\widetilde{L}$ depends only on f and $\mathcal{D}_x$. We set

$$\widetilde{V}(S) := \mathop{\mathbf{E}}_{\boldsymbol{z}_S\sim\mathcal{D}_{x,S}}\left(\mathop{\mathbf{E}}_{\boldsymbol{w}\sim\mathcal{D}_x}[f(\boldsymbol{w})|\boldsymbol{w}_S = \boldsymbol{z}_S]\right)^2 . \tag{7}$$

The function $\widetilde{V}$ can be seen as a measure of heterogeneity of the within-leaf mean values of the target function f, from the leafs created by split S. The following condition is required.

Condition 8 (Approximate Submodularity). Let $C \geq 1$. We say that the function $\widetilde{V}$ is C-approximate submodular if and only if for any $T, S \subseteq [d]$, such that $S \subseteq T$ and any $i \in [d]$, it holds that

$$\widetilde{V}(T \cup \{i\}) - \widetilde{V}(T) \leq C \cdot (\widetilde{V}(S \cup \{i\})) - \widetilde{V}(S)).$$

We can equivalently write this condition as (this is the formulation we used in Condition 1):

$$\widetilde{L}(T) - \widetilde{L}(T \cup \{i\}) \leq C \cdot (\widetilde{L}(S) - \widetilde{L}(S \cup \{i\})).$$

The reduction of the mean squared error when the coordinate i is added to a set of splits T is upper bounded by the reduction when adding i to a subset of T, i.e., *if adding i does not decrease the mean squared error significantly at some point (when having the set S), then i cannot decrease the mean squared error significantly in the future either (for any superset of S).* We remark that this condition is necessary for any greedy algorithm to work (see Syrgkanis & Zampetakis, 2020).

Empirical MSE. For the algorithm, we will use the empirical version of the mean square error. Provided a set of splits S, we have that

$$L_n(S) = \frac{1}{n}\sum_{j\in[n]}\left(y^{(j)} - f^{(n)}(\boldsymbol{x}^{(j)};S)\right)^2 = \frac{1}{n}\sum_{j\in[n]}(y^{(j)})^2 - \frac{1}{n}\sum_{j\in[n]} f^{(n)}(\boldsymbol{x}^{(j)};S)^2 \tag{8}$$

$$=: \frac{1}{n}\sum_{j\in[n]}(y^{(j)})^2 - V_n(S)\,. \tag{9}$$

Algorithm 9 Level-Splits Algorithm (see (Syrgkanis & Zampetakis, 2020))

```
1: Input: honesty flag $h$, training dataset $D_n$, maximum number of splits $\log(t)$.
2: Output: Tree approximation of $f$.
3: LevelSplits-Algo($h, D_n, \log(t)$):
4: $\mathcal{V} \leftarrow D_{n,x}$ {Keep only training features $\boldsymbol{x}$.}
5: if $h = 1$ then Split randomly $D_n$ in half; $D_{n/2}, D'_{n/2}, n \leftarrow n/2, \mathcal{V} \leftarrow D'_{n,x}$
6: Set $\mathcal{P}_0 = \{\{0,1\}^d\}$ {The partition that corresponds to the root.}
7: $\mathcal{P}_\ell = \emptyset$ for any $\ell \in [n]$
8: level $\leftarrow -1, S \leftarrow \emptyset$
9: while level $< \log(t)$ do
10:     level $\leftarrow$ level $+ 1$
11:     Select the direction $i \in [d]$ that maximizes $V_n(S \cup \{i\})$ {For $V_n$, see Equation (9).}
12:     for all $C \in \mathcal{P}_{level}$ do
13:         Partition the cell $C$ into the cells $C_k^i = \{\boldsymbol{x} : \boldsymbol{x} \in C \wedge x_i = k\}, k \in 0, 1$
14:         if $|\mathcal{V} \cap C_0^i| \geq 1 \wedge |\mathcal{V} \cap C_1^i| \geq 1$ then
15:             $\mathcal{P}_{level+1} \leftarrow \mathcal{P}_{level+1} \cup \{C_0^i, C_1^i\}$
16:         else
17:             $\mathcal{P}_{level+1} \leftarrow \mathcal{P}_{level+1} \cup \{C\}$
18:         endif
19:     endfor
20:     $S \leftarrow S \cup \{i\}$
21: endwhile
22: Output $(\mathcal{P}_n, f^{(n)}) = (\mathcal{P}_{level+1}, \boldsymbol{x} \mapsto f^{(n)}(\boldsymbol{x}; S))$
```

The following result summarizes the theoretical guarantees for algorithms with Decision Trees via the Level-Splits criterion (see Syrgkanis & Zampetakis, 2020).

Theorem D.2 (Learning with Decision Trees via Level-Splits (see (Syrgkanis & Zampetakis, 2020))). *Let $\varepsilon, \delta > 0$. Let $H > 0$. Let D_n be i.i.d. samples from the nonparametric regression model $y = f(\boldsymbol{x}) + \xi$, where $f(\boldsymbol{x}) \in [-1/2, 1/2], \xi \sim \mathcal{E}, \mathbf{E}_{\xi\sim\mathcal{E}}[\xi] = 0$ and $\xi \in [-1/2, 1/2]$. Let also S_n be the set of splits chosen by the Level-Splits algorithm (see Algorithm 9), with input $h = 0$. The following statements hold.*

1. *Given $n = \widetilde{O}\left(\log(d/\delta) \cdot (Cr/\varepsilon)^{Cr+2}\right)$ samples, if f is r-sparse as per Definition A.1 and under the submodularity Condition 8, and if we set the number of splits to be $\log(t) = \frac{Cr}{Cr+2}(\log(n) - \log(\log(d/\delta)))$, then it holds that*

$$\Pr_{\mathrm{D}_n\sim\mathcal{D}^n}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[(f(\boldsymbol{x}) - f^{(n)}(\boldsymbol{x};S_n))^2\right] > \varepsilon\right] \leq \delta\,.$$

2. *If f is r-sparse as per Definition A.1 and under the submodularity Condition 8 and the independence of features condition, given $n = \widetilde{O}\left(\log(d/\delta) \cdot 2^r \cdot (C/\varepsilon)^2\right)$ samples and if we set the number of splits to be $\log(t) = r$, then it holds that*

$$\Pr_{\mathrm{D}_n\sim\mathcal{D}^n}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[(f(\boldsymbol{x}) - f^{(n)}(\boldsymbol{x};S_n))^2\right] > \varepsilon\right] \leq \delta\,.$$

Fully Grown Honest Forests with Level-Splits Algorithm. We first explain the term Fully Grown Honest Forests: The term *Fully Grown* or (deep) means that we split every node until every leaf has exactly 1 training sample. The term *Honest* (see Wager & Athey, 2018) corresponds to the following: the regression tree algorithms operate in two phases where in both stages we use the same set of training examples. In honest trees, we split randomly the training set and use half of the dataset ($D_{n/2}$) in find a partition of $\{0,1\}^d$ and the other half ($D'_{n/2}$) to assign the values in the cells. Finally, the term *Forest* is used when we subsample s out of n samples and use them in order to build independent trees; we then output the average of these trees and this function is denoted by $f^{(n,s)}$.

In the work of Syrgkanis & Zampetakis (2020), a result about Fully Grown Forests via the Level-Splits criterion is provided under Condition 5. Shortly, it holds that, using a training set of size $n = \widetilde{O}\left(\frac{2^r \log(1/\delta)}{\varepsilon}\left(\frac{\log(d)}{\beta^2} + \frac{1}{\zeta}\right)\right)$ and if for every $\boldsymbol{w} \in \{0,1\}^r$, it holds for the marginal probability that $\mathbf{Pr}_{\boldsymbol{z}\sim\mathcal{D}_x}(\boldsymbol{z}_R = \boldsymbol{w}) \notin (0, \zeta/2^r)$ and if $s = \widetilde{\Theta}(2^r \cdot (\log(d/\delta)/\beta^2 + \log(1/\delta)/\zeta))$, then it holds that $\mathbf{Pr}_{D_n\sim\mathcal{D}^n}\left(\mathbf{E}_{\boldsymbol{x}\sim\mathcal{D}_x}[(f(\boldsymbol{x}) - f^{(n,s)}(\boldsymbol{x}))^2] \geq \varepsilon\right) \leq \delta$. We remark that every tree $f(\boldsymbol{x}, D_s)$ is built using Algorithm 9, with inputs: $\log(t)$ large enough so that every leaf has two or three samples and $h = 1$.

D.4. Breiman's Algorithm

We now turn our attention to the Breiman's criterion. We define the total expected mean square error that is achieved by a partition $\mathcal{P}$ of $\{0,1\}^d$ in the population model as follows:

$$\widetilde{L}(\mathcal{P}) = \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left(f(\boldsymbol{x}) - \mathop{\mathbf{E}}_{\boldsymbol{z}\sim\mathcal{D}_x}[f(\boldsymbol{z})|\boldsymbol{z}\in\mathcal{P}(\boldsymbol{x})]\right)^2\right] = \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}[f^2(\boldsymbol{x})] - \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left(\mathop{\mathbf{E}}_{\boldsymbol{z}\sim\mathcal{D}_x}[f(\boldsymbol{z})|\boldsymbol{z}\in\mathcal{P}(\boldsymbol{x})]\right)^2 .$$

As in the Level-Splits criterion, we set

$$\widetilde{V}(\mathcal{P}) = \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left(\mathop{\mathbf{E}}_{\boldsymbol{z}\sim\mathcal{D}_x}[f(\boldsymbol{z})|\boldsymbol{z}\in\mathcal{P}(\boldsymbol{x})]\right)^2 .$$

In order to define the splitting criterion of the algorithm (due to the local nature of Breiman), one has to introduce the local version of the expected MSE for the cell A:

$$\begin{aligned}\widetilde{L}_\ell(A,\mathcal{P}) &= \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left(f(\boldsymbol{x}) - \mathop{\mathbf{E}}_{\boldsymbol{z}\sim\mathcal{D}_x}[f(\boldsymbol{z})|\boldsymbol{z}\in\mathcal{P}(\boldsymbol{x})]\right)^2 \Big| \boldsymbol{x}\in A\right] \\ &= \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}[f^2(\boldsymbol{x})|\boldsymbol{x}\in A] - \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left(\mathop{\mathbf{E}}_{\boldsymbol{z}\sim\mathcal{D}_x}[f(\boldsymbol{z})|\boldsymbol{z}\in\mathcal{P}(\boldsymbol{x})]\right)^2 \Big| \boldsymbol{x}\in A\right] .\end{aligned}$$

We set

$$\widetilde{V}_\ell(A,\mathcal{P}) = \mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[\left(\mathop{\mathbf{E}}_{\boldsymbol{z}\sim\mathcal{D}_x}[f(\boldsymbol{z})|\boldsymbol{z}\in\mathcal{P}(\boldsymbol{x})]\right)^2 \Big| \boldsymbol{x}\in A\right] .$$

The following condition is required for decision tree-based algorithms that use the Breiman's criterion.

Condition 9 (Approximate Diminishing Returns). For $C \geq 1$, we say that the function $\widetilde{V}$ has the C-approximate diminishing returns property if for any cells A, A', any $i \in [d]$ and any $T \subseteq [d]$ such that $A' \subseteq A$, it holds that

$$\widetilde{V}_\ell(A', T\cup\{i\}) - \widetilde{V}_\ell(A', T) \leq C \cdot (\widetilde{V}_\ell(A, i) - \widetilde{V}_\ell(A)) .$$

For the algorithm, we need the empirical mean squared error, conditional on a cell A and a potential split direction i, which is defined as follows: let $N_n(A)$ be the number of training points in the cell A. Recall that $A^i_z = \{\boldsymbol{x}\in A | x_i = z\}$ for

$z \in \{0,1\}$. Also, set $f^{(n)}(\boldsymbol{x};\mathcal{P}) = g^{(n)}(\mathcal{P}(\boldsymbol{x}))$. Then, we have that

$$L_n^\ell(A,i) = \sum_{z\in\{0,1\}} \frac{N_n(A_z^i)}{N_n(A)} \sum_{j:\boldsymbol{x}^{(j)}\in A_z^i} \frac{1}{N_n(A_z^i)} (y^{(j)} - f^{(n)}(\boldsymbol{x}^{(j)};\mathcal{P}(\boldsymbol{x}^{(j)})))^2 \tag{10}$$

$$= \frac{1}{N_n(A)} \sum_{j:\boldsymbol{x}^{(j)}\in A} (y^{(j)})^2 - \sum_{z\in\{0,1\}} \frac{N_n(A_z^i)}{N_n(A)} (g^{(n)}(A_z^i))^2 \tag{11}$$

$$=: \frac{1}{N_n(A)} \sum_{j:\boldsymbol{x}^{(j)}\in A} (y^{(j)})^2 - V_n^\ell(A,i)\,. \tag{12}$$

Algorithm 10 Breiman's Algorithm (see (Syrgkanis & Zampetakis, 2020))

```
1: Input: honesty flag $h$, training dataset $D_n$, maximum number of splits $t$.
2: Output: Tree approximation of $f$.
3: Breiman-Algo($h, D_n, t$):
4: $\mathcal{V} \leftarrow D_{n,x}$
5: if $h = 1$ then Split randomly $D_n$ in half; $D_{n/2}, D'_{n/2}, n \leftarrow n/2, \mathcal{V} \leftarrow D'_{n,x}$
6: Set $\mathcal{P}_0 = \{\{0,1\}^d\}$ {The partition that corresponds to the root.}
7: $\mathcal{P}_\ell = \emptyset$ for any $\ell \in [n]$
8: level $\leftarrow 0, n_{nodes} \leftarrow 1$, queue $\leftarrow \mathcal{P}_0$
9: while $n_{nodes} < t$ do
10:     if queue $= \emptyset$ do
11:         level $\leftarrow$ level $+ 1$, queue $\leftarrow \mathcal{P}_{level}$
12:     endif
13:     Pick $A$ the first element in queue
14:     if $|\mathcal{V} \cap A| \leq 1$ then
15:         queue $\leftarrow$ queue $\setminus \{A\}, \mathcal{P}_{level+1} \leftarrow \mathcal{P}_{level+1} \cup \{A\}$
16:     else
17:         Select $i \in [d]$ that maximizes $V_n^\ell(A,i)$ {See Equation (12).}
18:         Cut the cell $A$ to cells $A_k^i = \{\boldsymbol{x} | \boldsymbol{x} \in A \wedge x_i = k\}, k = 0, 1$
19:         queue $\leftarrow$ queue $\setminus \{A\}, \mathcal{P}_{level+1} \leftarrow \mathcal{P}_{level+1} \cup \{A_0^i, A_1^i\}$
20:     endif
21: endwhile
22: $\mathcal{P}_{level+1} \leftarrow \mathcal{P}_{level+1} \cup$ queue
23: Output $(\mathcal{P}_n, f^{(n)}) = (\mathcal{P}_{level+1}, \boldsymbol{x} \mapsto f^{(n)}(\boldsymbol{x};\mathcal{P}_{level+1}))$
```

Theorem D.3 (Learning with Decision Trees via Breiman (see (Syrgkanis & Zampetakis, 2020))). *Let $\varepsilon, \delta > 0$. Let $H > 0$. Let D_n be i.i.d. samples from the nonparametric regression model $y = f(\boldsymbol{x}) + \xi$, where $f(\boldsymbol{x}) \in [-1/2, 1/2], \xi \sim \mathcal{E}, \mathbf{E}_{\xi\sim\mathcal{E}}[\xi] = 0$ and $\xi \in [-1/2, 1/2]$. Let also P_n be the partition that the algorithm (see Algorithm 10) returns with input $h = 0$. The following statements hold.*

1. *If f be r-sparse as per Definition A.1 and if the approximate diminishing returns Condition 9 holds, then given $n = \widetilde{O}\left(\log(d/\delta)(C \cdot r/\varepsilon)^{C\cdot r+3}\right)$ samples and if we set $\log(t) \geq \frac{Cr}{Cr+3}(\log(n) - \log(\log(d/\delta)))$, then it holds that*

$$\Pr_{\mathrm{D}_n \sim \mathcal{D}^n}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[(f(\boldsymbol{x}) - f^{(n)}(\boldsymbol{x};P_n))^2\right] > \varepsilon\right] \leq \delta\,.$$

2. *If f is r-sparse as per Definition A.1, if the approximate diminishing returns Condition 9 holds and the distribution $\mathcal{D}_x$ is a product distribution, given $n = \widetilde{O}\left(C^2 2^r \log(d/\delta)/\varepsilon^3\right)$ samples and if we set $\log(t) \geq r$, then it holds that*

$$\Pr_{\mathrm{D}_n \sim \mathcal{D}^n}\left[\mathop{\mathbf{E}}_{\boldsymbol{x}\sim\mathcal{D}_x}\left[(f(\boldsymbol{x}) - f^{(n)}(\boldsymbol{x};P_n))^2\right] > \varepsilon\right] \leq \delta\,.$$

Finally, in the work of Syrgkanis & Zampetakis (2020), a result about Fully Grown Forests via the Breiman's criterion is provided under Condition 6. Shortly, it holds that, using a training set of size $n = \frac{2^r \log(d/\delta)}{\varepsilon \zeta \beta^2}$ and if $s = \widetilde{\Theta}(\frac{2^r \log(d/\delta)}{\zeta \beta^2})$, then it holds that $\mathbf{Pr}_{D_n \sim \mathcal{D}^n}\left(\mathbf{E}_{\boldsymbol{x} \sim \mathcal{D}_x}[(f(\boldsymbol{x}) - f^{(n,s)}(\boldsymbol{x}))^2] \geq \varepsilon\right) \leq \delta$. Note that every tree $f(\boldsymbol{x}, D_s)$ is built using the Algorithm 10, with inputs: $\log(t)$ large enough so that every leaf has two or three samples, training set D_s and $h = 1$.

E. Background on Statistical Learning Theory

For a detailed exposition of a statistical learning theory perspective to binary classification, we refer to Bousquet et al. (2003); Boucheron et al. (2005).

Talagrand's Inequality. Let $Pf = \mathbf{E}\, f$ and $P_n f$ be the corresponding empirical functional. Talagrand's inequality provides a concentration inequality for the random variable $\sup_{f \in \mathcal{F}}(Pf - P_n f)$, which depends on the maximum variance attained by any function over the class $\mathcal{F}$.

Fact 1 (Theorem 5.4 in Boucheron et al. (2005)). Let $b > 0$ and $\mathcal{F}$ be a set of functions from $\mathbb{X}$ to $\mathbb{R}$. Assume that all functions in $\mathcal{F}$ satisfy $Pf - f \leq b$. Then, with probability at least $1 - \delta$, it holds that

$$\sup_{f \in \mathcal{F}}(Pf - P_n f) \leq 2\,\mathbf{E}[\sup_{f \in F}(Pf - P_n f)] + \sqrt{\frac{2 \sup_{f \in \mathcal{F}} \mathbf{Var}(f) \log(1/\delta)}{n}} + \frac{4b \log(1/\delta)}{3n} \,.$$

On Tsybakov's Condition. The following property holds for the Tsybakov's noise condition.

Fact 2 (Tsybakov's Condition). Let $\mathcal{G}$ be a class of binary classifiers. Under the Tsybakov's noise condition (see Condition 2.(ii)) with $a, B > 0$ and for $i \neq j$, it holds that

$$\mathop{\mathbf{Pr}}_{(\boldsymbol{x},\sigma) \sim \mathcal{D}_R^{\boldsymbol{q}}}[g(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x}) | \sigma \ni \{i,j\}] \leq C_{a,B}(L_{i,j}(g) - L_{i,j}(g^\star))^a \,,$$

where $C_{a,B} = \frac{B^{1-a}}{(1-a)^{1-a} a^a}$, $g^\star$ is the Bayes classifier and the loss function is defined as

$$L_{i,j}(g) := \mathop{\mathbf{E}}_{(\boldsymbol{x},\sigma) \sim \mathcal{D}_R^{\boldsymbol{q}}} \mathbf{1}\{g(\boldsymbol{x}) \neq \mathrm{sgn}(\sigma(i) - \sigma(j)) \cap \sigma \ni \{i,j\}\} \,.$$

Proof. Let us set $L^\star_{i,j} = L_{i,j}(g^\star)$. Define the quantity

$$\eta(\boldsymbol{x}) = \mathop{\mathbf{E}}_{(X,\sigma)}[\mathrm{sgn}(\sigma(i) - \sigma(j)) = +1 | X = \boldsymbol{x}] \,.$$

The loss of the classifier is equal to

$$L_{i,j}(g) - L^\star_{i,j} = \mathop{\mathbf{E}}_{(\boldsymbol{x},\sigma) \sim \mathcal{D}_R^{\boldsymbol{q}}}[|2\eta(\boldsymbol{x}) - 1| \cdot \mathbf{1}\{g(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x})\} | \sigma \ni \{i,j\}] \,,$$

and so

$$L_{i,j}(g) - L^\star_{i,j} \geq t\,\mathbf{E}[\mathbf{1}\{g(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x})\} \cdot \mathbf{1}\{|2\eta(\boldsymbol{x}) - 1| \geq t\} | \sigma \ni \{i,j\}] \,.$$

Using Markov's inequality, we have that for all $t \geq 0$:

$$\begin{aligned} L_{i,j}(g) - L^\star_{i,j} \geq\ & t\,\mathbf{Pr}[|2\eta(\boldsymbol{x}) - 1| \geq t | \sigma \ni \{i,j\}] \\ & - t\,\mathbf{E}[\mathbf{1}\{g(\boldsymbol{x}) = g^\star_{i,j}(\boldsymbol{x})\}\mathbf{1}\{|2\eta(\boldsymbol{x}) - 1| \geq t\} | \sigma \ni \{i,j\}] \,. \end{aligned}$$

The Tsybakov's condition implies that

$$L_{i,j}(g) - L^\star_{i,j} \geq t(1 - Bt^{\frac{a}{1-a}}) - t\,\mathbf{Pr}[g(\boldsymbol{x}) = g^\star_{i,j}(\boldsymbol{x}) | \sigma \ni \{i,j\}] \,.$$

Hence,

$$L_{i,j}(g) - L^\star_{i,j} \geq t(\mathbf{Pr}[g(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x}) | \sigma \ni \{i,j\}] - Bt^{\frac{a}{1-a}}) \,.$$

Choosing t appropriately, one gets that

$$\mathbf{Pr}[g(\boldsymbol{x}) \neq g^\star_{i,j}(\boldsymbol{x}) | \sigma \ni \{i,j\}] \leq \frac{B^{1-a}}{(1-a)^{1-a} a^a}(L_{i,j}(g) - L^\star_{i,j})^a \,.$$

The proof is concluded by setting $C_{a,B} = \frac{B^{1-a}}{(1-a)^{1-a} a^a}$. $\square$