On the Consistency Rate of Decision Tree Learning Algorithms

Anonymous Author
Anonymous Institution

Abstract

Decision tree learning algorithms such as CART are generally based on heuristics that maximizes the impurity gain greedily. Though these algorithms are practically successful, theoretical properties such as consistency are far from clear. In this paper, we disclose that the most serious obstacle encumbering consistency analysis for decision tree learning algorithms lies in the fact that the worst-case impurity gain, i.e., the core heuristics for tree splitting, can be zero. Based on this recognition, we present a new algorithm, named Grid Classification And Regression Tree (GridCART), with a provable consistency rate $\mathcal{O}(n^{-1/(d+2)})$, which is the first consistency rate proved for heuristic tree learning algorithms.

1 Introduction

Decision trees are among the most popular methods in machine learning and data mining. The widely used CART [Breiman et al., 1984] is a heuristic tree-based learning algorithm and is usually selected to be a base learner in ensemble learning such as the popular random forest [Breiman, 2001], gradient boosting decision tree [Chen and Guestrin, 2016; Ke et al., 2017] and deep forest [Zhou and Feng, 2019]. At each step, CART chooses a dimension and a cut point to maximize the impurity gain, and then splits every leaf t into two children $t \cap \{\mathbf{x} \mid x^{(k)} < s\}$ and $t \cap \{\mathbf{x} \mid x^{(k)} \geqslant s\}$. The process will continue recursively until all the leaves contain samples with the same label. The algorithm is highly heuristic, but it succeeds in various types of tasks. Thus, it is crucial to understand the mysteries behind the great success.

Consistency describes whether a learning algorithm can eventually learn the optimal classifier from a large amount of training data with high probability [Devroye et al., 1997], and thus a successful algorithm should be provably consistent. Nevertheless, despite having been proposed for more than 30 years [Breiman et al., 1984], whether this type of heuristic algorithm for tree learning is generally consistent still remains mysterious. There have been a lot of efforts on this issue. Devroye et al. [1997], Biau et al. [2008], and Gao and Zhou [2020] showed the consistency of pure random trees, which splits every leaf independently on samples, whereas the label-dependent analysis of tree generating is instead the most challenging. Scornet et al. [2015] was the first to analyze the label-dependent node splitting and showed that CART is consistent under the assumptions of an additive target function and uniformly randomly distributing features. Klusowski [2021] eliminated all the assumptions on the feature distribution and presented that CART is consistent in the high dimension for additive target functions. However, we usually have no side information about the target function when one practically uses heuristic algorithms in real-world tasks. Thus, these assumptions are still far from mild.

In this paper, we revisit this issue by exploring a new impurity measure, called Influence [Kahn et al., 1988]. Our study point is from some seminal works [Blanc et al., 2020b; Fiat and Pechyony, 2004], in which they showed the possibility that using Influence as the impurity measure contributes to the convergence of error, which is a key ingredient for consistency, without any assumptions on the target function. Nevertheless, previous studies related to Influence cannot give consideration to both consistency and practiced effectiveness, since an Influence oracle is required for tree generating. We bridge this gap by proposing GridCART, which not only can run practically but also is consistent without assumptions on the target function. The contributions of this work are summarized as follows:

1. We revisit the challenge for the consistency analysis of heuristic tree learning algorithms and disclose a serious obstacle that the worst-case gain in impurity equals to zero when proving consistency.
2. We propose Grid Classification And Regression Tree (GridCART), whose gain in impurity at each node splitting can be lower bounded nontrivially, making it feasible for the consistency analysis.

Preliminary work. Under review by AISTATS 2023. Do not distribute.

3. We present a consistency rate $\mathcal{O}(n^{-1/(d+2)})$ for GridCART, which is the first consistency rate for heuristic tree learning algorithms, even under weaker assumptions for consistency in previous studies.

The rest of this paper is organized as follows. Section 2 reviews some related work. Section 3 introduces some essential background knowledge and notations. Section 4 shows the difficulty in proving convergence of CART. Section 5 presents our proposed GridCART for binary classification, its consistency rate, and the time complexity. Section 6 extends GridCART to more general tasks. Section 7 presents some experimental results to show the effectiveness of our theory. Section 8 concludes our work with prospects.

2 Related Work

The history of decision trees dates back to the 1970s. Quinlan [1979, 1986] first proposed ID3 for classification in discrete feature space, one of the most popular tree learning algorithms till now. C4.5 for continuous features and CART for both classification and regression were then proposed by Quinlan [1993] and Breiman et al. [1984], respectively. Brodley and Utgoff [1995] thought about multivariate decision trees, which have no restriction on the orthogonality of the split. Mingers [1989] studied how the choice of splitting criterion impacted the generalization performance. Utgoff [1989] proposed ID5R, which enabled incremental learning for decision trees. Geurts et al. [2006] introduced the extremely randomized trees whose structures are independent of the labels of the learning samples. Tree-based ensemble algorithms such as random forest [Breiman, 2001] XGBoost [Chen and Guestrin, 2016], LightGBM [Ke et al., 2017] and deep forest [Zhou and Feng, 2019] are also popular and effective methods.

There are great efforts on the consistency of decision trees. Biau et al. [2008] investigated the connection between decision trees and tree ensemble methods, and they proved that purely randomized trees are consistent. Gao and Zhou [2020]; Gao et al. [2022] then presented the convergence rates of purely randomized trees and a simplified variant of Breiman's original CART [Breiman et al., 1984]. The growth of random trees they analyzed are label-independent, whereas heuristic algorithms for tree learning are usually label-dependent. Scornet et al. [2015] showed that CART is consistent under the assumptions of uniformly distributing features and additive target functions with Gaussian noise. Klusowski [2021] provided universal consistency of CART in the high dimensions assuming the target function is additive too. It is worth mentioning that Blanc et al. [2020b]; Fiat and Pechyony [2004] studied a variant of CART which uses the well-known Influence [Kahn et al., 1988; O'Donnell, 2014] as the impurity measure. Their works are mostly related to ours, but they studied the training error of the tree for Boolean functions, while this work focuses on the consistency rate for real-valued feature spaces.

3 Preliminary

Setting Let $\mathcal{X} = [0,1]^p$ and $\mathcal{Y} = \{0,1\}$ be the feature space and label space respectively. Let $\mathbf{p} : \mathcal{X} \to \mathbb{R}^+$ be an underlying probability density on $\mathcal{X}$ and $\eta(\mathbf{x}) = \mathbb{E}[Y \mid \mathbf{X} = \mathbf{x}]$ is the conditional probability function. We observe data $D_n = \{(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), ..., (\mathbf{X}_n, Y_n)\}$ drawn i.i.d. with $\mathbf{X}_i \sim \mathbf{p}$ and $Y_{i|\mathbf{X}_i} \sim \text{Bernoulli}(\eta(\mathbf{X}_i))$, where $\mathbf{X}_i = (\mathbf{X}_i^{(1)}, \mathbf{X}_i^{(2)}, ..., \mathbf{X}_i^{(p)})$ is a p-dimension vector. $R(h) \triangleq \Pr[h(\mathbf{X}) \neq Y]$ denotes the generalization error of hypothesis h. We write Bayes error and the target function (or Bayes optimal classifer) as $R^\star$ and f, respectively, where $R^\star = \min_{h:\mathcal{X}\to\mathcal{Y}} R(h)$ and $f \in \arg\max_{h:\mathcal{X}\to\mathcal{Y}} R(h)$.

As the underlying distribution $\mathbf{p}$ and the target function f are unknown, we run a learning algorithm and obtain estimators $\hat{\mathbf{p}}$ and $\hat{f}$. For simplicity, we write $\mathbb{E}_{\mathbf{X}\sim\mathbf{p}}$ and $\Pr_{\mathbf{X}\sim\mathbf{p}}$ as $\mathbb{E}_{\mathbf{p}}$ and $\Pr_{\mathbf{p}}$, respectively, so as to $\mathbb{E}_{\hat{\mathbf{p}}}$ and $\Pr_{\hat{\mathbf{p}}}$.

We say a distribution $\mathbf{p}$ is a product distribution if $\mathbf{p}_{\mathbf{X}} \equiv \prod_{k=1}^{p} \mathbf{p}_{\mathbf{X}^{(k)}}$, i.e., $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(d)}$ are independent on each others. The convergence rate we analyze in this paper is about the excess error, which is defined by excess error $\triangleq R(T) - R^\star$. If the generalization errors of a sequence of classifier $\{T_n\}_{n=1}^{\infty}$ converge to Bayes error or equivalently the excess error converges to 0, as $n \to \infty$, we say the sequence of tree classifiers is consistent.

Tree Denote a tree, that consists of many leaf nodes by T and let leaves$(T) \triangleq \{t \mid t \text{ is a leaf of } T\}$ be the leaves set. Every leaf, which is a subset of $\mathcal{X}$, is the intersection of sets like $\{\mathbf{x} \mid \mathbf{x}^{(i)} < s_i\}$ or $\{\mathbf{x} \mid \mathbf{x}^{(j)} \geqslant s_j\}$, where $\mathbf{x}$ is a p-dimension vector in $\mathcal{X}$ and $\mathbf{x}^{(k)}$ is its k-th dimension. For all $\mathbf{x} \in t$, the prediction $T(\mathbf{x})$ is decided as follows:

$$T(\mathbf{x}) = \arg\max_{y\in\{0,1\}} \sum_{i=1}^{n} \mathbb{I}\{Y_i = y, \mathbf{X}_i \in t\} .$$

An impurity function $G : [0,1] \to [0,1]$ is a strongly concave function that satisfies $G(0) = G(1) = 0, G(1/2) = 1$, and $G(x) = G(1-x)$ for any x. The impurity function commonly used in famous heuristic algorithms comprises entropy, Gini-index, etc. The impurity function provides an efficient and effective way to build a tree greedily. At each step, the learning algorithm chooses a dimension k and a cut point s splitting t into two children $t_L \triangleq t \cap \{\mathbf{x} \mid \mathbf{x}^{(k)} < s\}$ and $t_R \triangleq t \cap \{\mathbf{x} \mid \mathbf{x}^{(k)} \geqslant s\}$. The choices of k and s maximize the impurity gain

$$p_t\Big[G(E_t) - w_{t_L} G(E_{t_L}) - w_{t_R} G(E_{t_R})\Big] ,$$

where $p_t \triangleq \Pr_{\hat{\mathbf{p}}}[\mathbf{X} \in t]$ and $E_t \triangleq \mathbb{E}_{\hat{\mathbf{p}}}\big[\hat{f}(\mathbf{X}) \mid \mathbf{X} \in t\big]$. $w_{t_L} \triangleq p_t/p_{t_L}$ and $w_{t_R} \triangleq p_t/p_{t_R}$ denote the ratios of sample numbers falling into the two new leaf nodes.

Assumptions We introduce assumptions in this paper.

Assumption 1. *Assume that $\eta(\mathbf{x})$ is L-Lipschitz, i.e., $\exists L \geqslant 0, |\eta(\mathbf{x}_1) - \eta(\mathbf{x}_2)| \leqslant L\|\mathbf{x}_1 - \mathbf{x}_2\|, \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$.*

Assumption 2. *The probability density function is continuously differentiable, i.e., the derivative $\mathrm{d}\mathbf{p}_{\mathbf{X}^{(k)}}/\mathrm{d}\mathbf{x}^{(k)}$ is a continuous function for all k.*

Assumption 3. *The underlying feature distribution is a product distribution, i.e., $\mathbf{p}(\mathbf{x}) \equiv \prod_{k=1}^{p} \mathbf{p}_{\mathbf{X}^{(k)}}(\mathbf{x}^{(k)})$.*

It is worth mentioning that Assumption 1 and 2 are mild to derive a consistency rate [Audibert and Tsybakov, 2007]. Assumption 3 looks strong, but it is usually employed by seminal studies [Brutzkus et al., 2020; Kalai and Teng, 2008; Takimoto and Maruoka, 2003], which is also a relaxation of uniform distribution in [Scornet et al., 2015]. Notice that our results in this paper do not depend on any assumptions on the target function used in [Blanc et al., 2020a; Klusowski, 2021; Scornet et al., 2015].

4 Difference Between CART and InfCART

In this section, we will show the difficulty in achieving a converging impurity which is one of the main ingredients for consistency. Section 4.2 shows that the zero impurity gain of CART in the worst case leads to a serious obstacle for the consistency analysis. Section 4.3 introduces Influence-based Classification And Regression Trees (InfCART), a variant of CART that uses Influence as the impurity measure, which can always get impurity gain even under no assumptions on the target function.

4.1 Roadmap for Bounding the Error

For any $K \in N^+$, CART builds a tree T_K with depth K. We define the potential C_K of T_K as follows:

$$C_K \triangleq \sum_{t \in \text{leaves}(T_K)} p_t \, \mathcal{G}(t) \, ,$$

where $p_t \triangleq \Pr[\mathbf{X} \in t]$ equals to the ratio of samples dropped in leaf t, and $\mathcal{G}(t) = G\Big(\mathbb{E}_{\hat{\mathbf{p}}}\big[\hat{f}(\mathbf{X}) \mid \mathbf{X} \in t\big]\Big)$ is the impurity of leaf t. Here, we use $\hat{\mathbf{p}}$ instead of $\mathbf{p}$ because the true $\mathbf{p}$ is unknown. Thus, C_K measures the average impurity of the tree T_K. We then show that the potential C_K of T_K is an upper bound of the error $\hat{R}(T_K)$ in Proposition 1.

Proposition 1. *For any tree T_K with depth K, and any impurity function G, we have $\hat{R}(T_K) \leqslant C_K$.*

One of the main ingredients of proving consistency is to show that the error converges to zero. Thus, it suffices to ensure a vanishing potential C_K from Proposition 1, which will be detailed in Section 4.2

4.2 Conventional Impurity Measures Fail

Let Δ_K be the potential gain for a tree T_K growing from depth K to $K+1$, then we have the following conclusion.

Proposition 2. *There exists a probability density $\mathbf{p}$ and a conditional probability function $\eta : \mathcal{X} \to [0,1]$, such that $\Delta_K = 0$ for any possible cut in every leaf.*

Proposition 2 shows that in the worst case, the potential gain Δ_K can always be zero, leading to a failure of proving a vanishing error, as discussed in Section 4.1. The proof of Proposition 2 is by simply giving a counterexample in which the maximal potential gain is zero but the error is not. More details will be shown in Appendix B.

4.3 Influence as an Impurity Measure

From the above discussion, the potential gain using conventional impurity measure, which is used in CART, tends to be zero in the worst case. In this section, we will present a new impurity measure called Influence, which never fails to achieve a nonzero impurity gain. To begin with, we introduce the definition of Influence as follows:

Definition 1. *(G-Influence) Let $\mathbf{p}$ and f be a product distribution and a mapping from $\mathcal{X}$ to $\mathcal{Y} = \{0,1\}$ respectively. G-Influence of function f on variable $\mathbf{X}^{(k)}$ is defined by*

$$\mathrm{Inf}_k^G[f] \triangleq \mathbb{E}_{\mathbf{p}^{(j \neq k)}}\left[G\Big(\mathbb{E}_{\mathbf{p}^{(k)}}\big[f(\mathbf{X})\big]\Big)\right] ,$$

where

$$\mathbb{E}_{\mathbf{p}^{(j\neq k)}}[\cdot] \triangleq \mathbb{E}_{\mathbf{X}^{(1)}\sim\mathbf{p}^{(1)}} \ldots \mathbb{E}_{\mathbf{X}^{(k-1)}\sim\mathbf{p}^{(k-1)}} \mathbb{E}_{\mathbf{X}^{(k+1)}\sim\mathbf{p}^{(k+1)}} \ldots \mathbb{E}_{\mathbf{X}^{(p)}\sim\mathbf{p}^{(p)}}[\cdot] ,$$

or equivalently taking expectation over all the coordinates except $\mathbf{X}^{(k)}$ over product distribution $\mathbf{p}$.

It is natural to define the conditional Influence as follows:

Definition 2. *(Conditional G-Influence) With the same notations in Definition 1, the conditional Influence of function f at leaf t is defined by*

$$\mathrm{Inf}_k^G[f] \triangleq \mathbb{E}_{\mathbf{p}^{(j \neq k)}}\left[G\Big(\mathbb{E}_{\mathbf{p}^{(k)}}\big[f(\mathbf{X}) \mid \mathbf{X} \in t\big]\Big)\right] .$$

With Definition 2 at hand, we can write the impurity gain induced by Influence as follows:

$$\Delta^{\mathrm{Inf}}(t,k,s) = p_t\Big(\mathcal{G}^{\mathrm{Inf}}(t) - w_{t_L}\mathcal{G}^{\mathrm{Inf}}(t_L) - w_{t_R}\mathcal{G}^{\mathrm{Inf}}(t_R)\Big) ,$$

where we re-define $\mathcal{G}(t) \triangleq \mathrm{Inf}_k^G[f \mid t]$ for simplicity. Then, InfCART splits every leaf t by maximizing

$$\max_{k,s} \Delta^{\mathrm{Inf}}(t,k,s) .$$

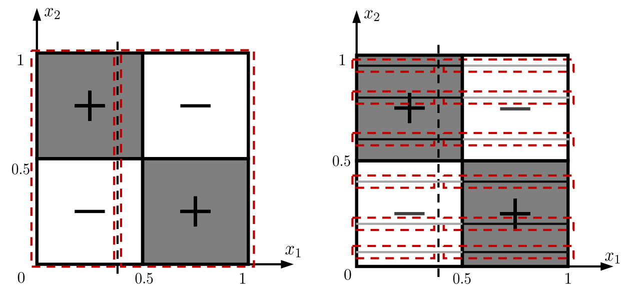
We compare CART with InfCART in Table 1, where we write the impurity measure of CART redundantly for comparison. Different from CART, InfCART takes an expectation taken later than the impurity function G. We will then show that the slight difference makes great help for lower bounding the impurity gain nontrivially.

Table 1: Comparison of CART and InfCART

	Impurity Measure
CART	$\mathcal{G}(t) = \frac{1}{p}\sum_{k=1}^{p}\left[G\left(\mathbb{E}_{\mathbf{p}^{(j\neq k)}}\mathbb{E}_{\mathbf{p}^{(k)}}\left[f(\mathbf{X}) \mid \mathbf{X}\in t\right]\right)\right]$
InfCART	$\mathcal{G}(t) = \frac{1}{p}\sum_{k=1}^{p}\mathbb{E}_{\mathbf{p}^{(j\neq k)}}\left[G\left(\mathbb{E}_{\mathbf{p}^{(k)}}\left[f(\mathbf{X}) \mid \mathbf{X}\in t\right]\right)\right]$

Proposition 3. *For any product distribution* $\mathbf{p}$ *on* $\mathcal{X}$*, and any target function* $\hat{f}:\mathcal{X}\to\mathcal{Y}$*. InfCART gets no impurity gain if and only if it reaches zero error.*

Remark 1. Proposition 3 shows the advantage of InfCART compared with CART; the former never fails to obtain impurity gain without any assumptions on the target functions [Blanc et al., 2020a; Klusowski, 2021; Scornet et al., 2015], while the latter can not.



(a) CART takes the average along the whole children, and thus gets no impurity gain.

(b) InfCART takes the average along the impurity gain on every line and gets impurity gain.

Figure 1: Comparison of the calculation of impurity gain between CART and InfCART.

As shown in Figure 1, CART calculates the impurity by taking the average of labels in each child into impurity function G, leading to impurities equal to 1 in both the parent and the two children, and thus gets no impurity gain for any cut in this example. Nevertheless, InfCART treats the leaf as many horizontal lines and calculates the impurity gain by taking an average over the impurity gain at every line, which always yields a nonzero impurity gain.

Unfortunately, InfCART requires an Influence oracle to calculate the impurity gain, as the expectations in Definition 2 are taken over the population, which is impractical for real-world tasks. The definition of Influence depends on Assumption 3 or independence between features [Kahn et al., 1988; O'Donnell, 2014]. However, it is not a strong assumption as discussed in Section 3.

5 GridCART: A Refined CART with Consistency Guarantee

This section presents GridCART, whose key idea is to estimate the underlying distribution, and then Influence can be obtained from the estimated distribution. Therefore, GridCART can be not only a practical but also a provably consistent heuristic algorithm. It consists of three steps: estimating the feature distribution, estimating the target function, and tree building, which is detailed in Algorithm 1-4.

5.1 Estimate the Feature Distribution

Algorithm 1 Histogram Density Estimation (**HDE**)

Input: Training dataset D, grid size h

Output: An estimated probability density function $\hat{\mathbf{p}}_{\mathbf{X}}$

```
1: for $k \in [p]$ do
2:     $\mathcal{B} \leftarrow \{[0,h],[0,2h],\dots[0,1]\}$
3:     for $b$ in $\mathcal{B}$ do
4:         $\omega(b) \leftarrow 1/(nh)\sum_{i=1}^{n}\mathbb{I}\{\mathbf{X}_i \in b\}$
5:     $\hat{\mathbf{p}}_{\mathbf{X}^{(k)}}(x) \leftarrow \omega(b)$, for $x \in b, b \in \mathcal{B}$
6: $\hat{\mathbf{p}}_{\mathbf{X}} \leftarrow \prod_{k=1}^{p}\mathbf{p}_{\mathbf{X}^{(k)}}$
7: return $\hat{\mathbf{p}}_{\mathbf{X}}$
```

In this step, we apply the well-known histogram method to estimate the feature distribution. We estimate a univariate distribution for each dimension and the estimated density of the joint distribution is the multiplication of them. The reason is that the notorious curse of dimensionality in nonparametric density estimation [Devroye et al., 1997] may be avoided as shown in Lemma 4. The process of histogram density estimation is detailed in Algorithm 1.

Lemma 4. *Suppose that* $\mathbb{L}_1$*-error of estimators* $\hat{\mathbf{p}}_{\mathbf{X}^{(k)}}$ *has upper bound* $\mathbb{E}\left[\|\hat{\mathbf{p}}_{\mathbf{X}^{(k)}} - \mathbf{p}_{\mathbf{X}^{(k)}}\|_1\right] \leqslant e_k$*. Then, under Assumption 3, we have*

$$\mathbb{E}\left[\left\|\prod_{k=1}^{p}\hat{\mathbf{p}}_{\mathbf{X}^{(k)}} - \mathbf{p}_{\mathbf{X}}\right\|_1\right] \leqslant \sum_{k=1}^{p} e_k \ .$$

Generally, it is hard to estimate the distribution in high dimensions. However, Lemma 4 reduces the p-dimension case to many 1-dimension cases, making the estimation much easier. The total loss can be bounded by the summation of error in each dimension which is linear in p in the worst case but not depends on p exponentially in general cases [Tsybakov, 2009]. Combining it with the error bound of the histogram density estimation, we obtain Lemma 5.

Lemma 5. *Under Assumption 2 and 3, the* $\mathbb{L}_1$*-error of histogram density estimator can be upper bounded by*

$$\mathbb{E}\left[\left\|\prod_{k=1}^{p}\hat{\mathbf{p}}_{\mathbf{X}^{(k)}} - \mathbf{p}_{\mathbf{X}}\right\|_1\right] \leqslant \mathcal{O}\left(p\left(h_n + \sqrt{\frac{1}{nh_n}}\right)\right).$$

Lemma 5 provides an error bound for the density estimator in Algorithm 1. Based on this lemma, we have the expected $\mathbb{L}_1$-error has order $\mathcal{O}(n^{-1/3})$, by selecting the window size $h_n = \Theta(n^{-1/3})$. The error bound does not depend on p exponentially, implying that the curse of dimensionality is avoided under Assumption 3.

5.2 Estimate the Target Function

We use the well-known histogram method to estimate the target function, which is detailed in Algorithm 2. We split the feature space $\mathcal{X} = [0,1]^p$ into many length-h_n disjoint cubes $[0,1]^p = \bigcup_{j\in\mathcal{J}} \mathcal{C}_j$. Every cube $\mathcal{C}_j$ has form $\times_{k=1}^{p}[m_k h_n, (m_k+1)h_n)$, where $m_k \in \mathbb{N}$, and we have the number of cubes $|\mathcal{J}| \propto \frac{1}{h_n^p}$ by simple calculations. The endpoint $\mathbf{1}_{1\times p}$ is neglected for simplicity. Let $\mathcal{C}(\mathbf{x})$ be the cube that contains $\mathbf{x}$ for any $\mathbf{x} \in \mathcal{X}$, then we can define

Algorithm 2 Learn Histogram Classifier (**LHC**)

Input: Training dataset D, grid size h
Output: A histogram classifier g_n^{hist}

```
1: // Cartesian product of intervals
2: $\mathfrak{C} \leftarrow \{ \times_{k=1}^{p}[m_k h, (m_k+1)h) \mid m_k \in [1/h], \forall j\}$
3: for $\mathcal{C}$ in $\mathfrak{C}$ do
4:     $\hat{\eta}(\mathcal{C}) \leftarrow \frac{\sum_{i=1}^n \mathbb{I}\{\mathbf{X}_i \in \mathcal{C}\}Y_i}{\sum_{i=1}^n \mathbb{I}\{\mathbf{X}_i \in \mathcal{C}\}}$
5: $g_n^{\text{hist}}(x) \leftarrow \mathbb{I}\{\hat{\eta}(c) > 1/2\}$, for $x \in \mathcal{C}, \mathcal{C} \in \mathfrak{C}$
6: return $g_n^{\text{hist}}$
```

$$\hat{\eta}_n(\mathbf{x}) \triangleq \frac{\sum_{i=1}^n Y_i \mathbb{I}\{\mathbf{X}_i \in \mathcal{C}(\mathbf{x})\}}{\sum_{i=1}^n \mathbb{I}\{\mathbf{X}_i \in \mathcal{C}(\mathbf{x})\}}$$

as the estimated conditional probability. If there are no samples falling into the cube $\mathcal{C}(\mathbf{x})$, we choose a uniformly random label as the output. The histogram classifier g_n^{hist} can eventually be defined by

$$g_n^{\text{hist}}(\mathbf{x}) = \mathbb{I}\{\hat{\eta}_n(\mathbf{x}) > 1/2\} ,$$

or equivalently predict the label by voting in every cube. Then we have the following conclusion.

Lemma 6. *Under Assumption 1, suppose that the histogram rule satisfies $h_n \to 0$ and $nh_n^p \to \infty$ as $n \to \infty$. For any $\mathbf{p}, \eta$, $n > 0$ we have*

$$\mathbb{E}\Big[R(g_n^{hist})\Big] - R^\star \leqslant \mathcal{O}\left(h_n + \sqrt{\frac{1}{nh_n^p}}\right) .$$

Exclusively provided that $h_n = \Theta(n^{-1/(d+2)})$, we obtain a consistency rate of order $\mathcal{O}(n^{-1/(d+2)})$ for g_n^{hist}.

Lemma 6 shows that the expected excess error of the histogram classifier can converge to zero in order related to n and h_n. Intuitively, as sample size $n \to \infty$ and $h_n \to 0$, the number of cubes $|\mathcal{J}| \to \infty$, which helps capture more detail in feature space; and the number of samples dropping into every cube $|\{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_n\} \cap \mathcal{C}| \to \infty$ in probability, which makes the voting more and more accurate.

5.3 Tree Constructing

With estimator $\hat{\mathbf{p}}$ and g_n^{hist} at hand, we can construct a tree by running Algorithm 3.

Algorithm 3 Tree Building (**TB**)

Input: A histogram density estimator $\hat{\mathbf{p}}$, a histogram classifier g_n^{hist}, maximal tree depth K_{max}, grid size h
Output: A learned tree $T_{K_{\text{max}}} : \mathbb{R}^p \to \{0,1\}$

```
1: Pass $\mathbf{p} \leftarrow \hat{\mathbf{p}}$ and $f \leftarrow g_n^{\text{hist}}$ to the definition of conditional Influence in Definition 2
2: // Initialize the leaves set by the whole feature space
3: Leaves set $\mathcal{L} \leftarrow \{[0,1]^p\}$
4: for $K = 1 \dots K_{\text{max}}$ do
5:     $\mathcal{L}' \leftarrow \mathcal{L}$ // Copy a leaves set
6:     for $t \in \mathcal{L}'$ do
7:         if All the samples in $t$ have the same label then
8:             continue
9:         else
10:            $(k_0, s_0) \leftarrow \arg\max_{k,s} \Delta^{\text{Inf}}(t,k,s)$
11:            $t_L \leftarrow t \cap \{\mathbf{x} \mid \mathbf{x}^{(k_0)} \leqslant s_0\}$
12:            $t_R \leftarrow t \cap \{\mathbf{x} \mid \mathbf{x}^{(k_0)} > s_0\}$
13:            $\mathcal{L} \leftarrow (\mathcal{L} - t) \cup t_L \cup t_R$
14: $T(\mathbf{x}) \leftarrow \arg\max_{y\in\{0,1\}} \sum_{i=1}^n \mathbb{I}\{\mathbf{X}_i \in t, Y_i = y\}$, for $\mathbf{x} \in t$
15: return $T$
```

To show the convergence guarantee of the error, we begin with the following definition:

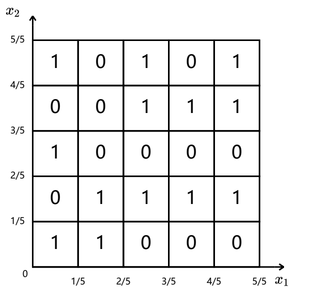
Definition 3 (N-piece function)**.** *A function f is called an N-piece function if $f_k(z)$ is a piecewise constant function with at most N pieces for any $k \in [p]$, where $f_k(z) \triangleq f\left(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(k-1)}, z, \mathbf{x}^{(k+1)}, \dots, \mathbf{x}^{(p)}\right)$ with $\mathbf{x} \in \mathbb{R}^p$.*

Definition 3 is an extension of piecewise-constant function in univariate cases. An example of a 5-piece function is given in Figure 2, in which given either x_1 or x_2, the univariate function is piece-wise constant with 5 pieces. It is not difficult to show that the histogram classifier learned by Algorithm 2 is an $1/h_n$-piece function.

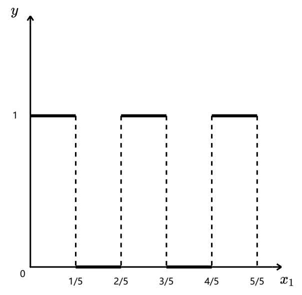
Theorem 7. *Suppose that the target function g is N-piece. Then, under Assumption 1-3, for the tree T generated by Algorithm 3 with depth K, we have*

$$\mathbb{E}\Big[\mathbb{I}\{T(\mathbf{X}) \neq g_n^{hist}(\mathbf{X})\}\Big] \leqslant \mathcal{O}\left(N^3/K\right) .$$

Remark 2. Theorem 7 shows that Algorithm 3 can fit any product distribution with N-piece function well. Comparing our result with the previous work [Blanc et al., 2020b], they considered binary features and size-s tree target functions, while we consider the setting where the feature space is real-valued and the target function is N-piece.



(a) An example of a 5-piece function from $[0,1]^2$ to $\{0,1\}$.



(b) The 5-piece in Figure 2a evaluating on $x_2 = 0.9$.

Figure 2: An example of a 5-piece function

5.4 Consistency Rate of GridCART

Algorithm 4 Grid-based Classification And Regression Tree (**GridCART**) for binary classification

Input: Training dataset $D = \{(\mathbf{X}_1, Y_1), ..., (\mathbf{X}_n, Y_n)\}$, maximal tree depth $K_{\max}$, grid size h
Output: A learned tree $T_{K_{\max}} : \mathbb{R}^p \to \{0,1\}$

```
1:  // Estimating the distribution and the target function
2:  for $k = 1 \dots p$ do
3:      // Learn the distribution $\mathbf{p}_{\mathbf{X}^{(k)}}$ by applying kernel
4:        density estimation with window size $h$
5:      $\hat{\mathbf{p}}_{\mathbf{X}^{(k)}} \leftarrow \mathbf{HDE}(\{\mathbf{X}_1^{(k)}, ..., \mathbf{X}_n^{(k)}\}, h)$
6:  $\hat{\mathbf{p}}_{\mathbf{X}} \leftarrow \prod_{k=1}^{p} \hat{\mathbf{p}}_{\mathbf{X}^{(k)}}$
7:
8:  // Learn the histogram classifier with grid size $h$
9:  $g_n^{\text{hist}} \leftarrow \mathbf{LHC}(D, h)$
10:
11: // Build a tree to fit the histogram classifier
12: $T \leftarrow \mathbf{TB}(\hat{\mathbf{p}}_{\mathbf{X}}, g_n^{\text{hist}})$
13: return $T$
```

In this subsection, we will show the consistency rate of GridCART. We start with the following decomposition:

$$R(T) - R^\star = \underbrace{\left(R(T) - R(g_n^{\text{hist}})\right)}_{\text{estimation error}} + \underbrace{\left(R(g_n^{\text{hist}}) - R^\star\right)}_{\text{approximation error}}, \tag{1}$$

where we define $R(T) - R(g_n^{\text{hist}})$ and $R(g_n^{\text{hist}}) - R^\star$ as the estimation error and the approximation error, respectively. Note that the definition may be slightly different from the known $R(T) - \inf_{h\in\mathcal{H}} R(h)$ and $\inf_{h\in\mathcal{H}} R(h) - R^\star$, since in other learning algorithms the learning objective is to fit the best classifier in the given hypothesis set, while the objective here is to fit the histogram classifier by a tree. We then decompose the estimation error as follows:

$$\begin{aligned} &R(T) - R(g_n^{\text{hist}}) \\ &= \mathbb{E}_{\mathbf{p},Y}\left[\mathbb{I}\{T(\mathbf{X}) \neq Y\}\right] - \mathbb{E}_{\mathbf{p},Y}\left[\mathbb{I}\{g_n^{\text{hist}}(\mathbf{X}) \neq Y\}\right] \\ &\leqslant \underbrace{\mathbb{E}_{\mathbf{p}}\left[\mathbb{I}\{T(\mathbf{X}) \neq g_n^{\text{hist}}(\mathbf{X})\}\right]}_{\text{term (a)}} + \underbrace{\mathbb{E}_{\mathbf{p}}\left\| \prod_{k=1}^{p} \hat{\mathbf{p}}_{\mathbf{X}^{(k)}} - \mathbf{p}_{\mathbf{X}} \right\|_1}_{\text{term (b)}}, \end{aligned} \tag{2}$$

where term (a) measures how the learned tree fits the histogram classifier, and term (b) measures the difference between the estimated distribution $\hat{\mathbf{p}}$ and the ground truth distribution $\mathbf{p}$. We learn a tree on an estimated distribution; if the estimation is accurate enough, and the tree fits well, then the learned tree can perform as well as the histogram classifier. Taking the error bounds in Lemma 5, Lemma 6, and Theorem 7 into Eq. (1) and Eq. (2), we obtain the consistency rate of GridCART in Theorem 8.

Theorem 8 (Consistency rate for binary classification)**.** *Under Assumption 1-3, the expected excess error of tree T_{K_n} learned by Algorithm 4 has the following upper bound*

$$\mathbb{E}_{D_n}\left[R(T_{K_n})\right] - R^\star \leqslant \mathcal{O}\left(\frac{1}{K_n h_n^3} + h_n + \sqrt{\frac{1}{n h_n^p}}\right).$$

Choosing $h_n = \Theta\left(n^{-1/(d+2)}\right)$, $K_n = \Omega(n^{4/(d+2)})$, we obtain a consistency rate of order $\mathcal{O}\left(n^{-1/(d+2)}\right)$.

Remark 3. The consistency rate in Theorem 8 is the first nontrivial consistency rate for heuristic tree learning algorithms. The previous consistency rate focused on the randomized tree, which generates independently on the labels, whereas previous works for label-dependent generating relied on strong assumptions and presented no consistency rate. More details are shown in Table 2.

Table 2: Comparison of our results with previous works

Algorithm	**Feature Distribution**	**Target Function**	**Result**
Pure Random Tree	any	any	$\mathcal{O}(n^{-1/(8d+2)})$ [Gao and Zhou, 2020]
Mid-point Random Tree	any	any	$\mathcal{O}((\ln n/n)^{1/(d+2)})$ [Gao and Zhou, 2020]
CART	uniformly random in $[0,1]^p$	additive model	$o(1)$ [Scornet et al., 2015]
CART	any	additive model	$\mathcal{O}(1/\log n)$ [Klusowski, 2021]
GridCART	product distribution	any	$\mathcal{O}(n^{-1/(d+2)})$ **(our result)**

The parameter K_n in Theorem 8 controls the maximal depth the tree can grow with. Choosing $K_n = \Omega(n^{4/(d+2)})$ reaches the best convergence rate in the theorem, whereas this may be inefficient in running time. Note that we can choose $K_n = \omega(\log^3(n))$ and $h_n = \Theta(1/\log n)$ to achieve both an efficient running time and the consistency.

5.5 Discussions about Time Complexity

In this section, we compare the time complexity of CART and our proposed GridCART. Note that the comparison focuses on the complexity of splitting at each leaf node, i.e., we neglect the comparison of the structures of trees, as the total time spent in building a tree is a simple summation of the splitting complexity at all leaves.

CART At each leaf node, CART traverses all the dimensions and sorts the samples falling into the leaf according to the dimension. After that, CART calculates the impurity gains for all the possible cuts. The time complexity of traversing all possible cuts is dominated by the complexity of sorting. For each dimension, the sorting may spend a time of order $\mathcal{O}(n\log n)$, resulting in a total time of order $\mathcal{O}(pn\log n)$ at each leaf node.

GridCART GridCART first estimates the underlying distribution using observed samples with time complexity of order $\mathcal{O}(n)$. When splitting a node, there are d/h_n possible cuts as there are $1/h_n$ possible cuts along each dimension. To calculate the impurity gain of a specific cut, we only need one traverse over all the cubes, resulting in time complexity of order $\mathcal{O}(1/h_n^p)$. Therefore, the total complexity of one split is $\mathcal{O}\big(d/h_n^{(d+1)}\big)$. We can achieve $\mathcal{O}\big(n + pn^{(d+1)/(d+2)}\big)$, provided $h_n = \Theta(n^{-1/(d+2)})$ as shown in Theorem 8, being slightly faster than CART.

From the above analysis, we conclude that: GridCART is faster than CART when splitting. This is because GridCART converts data to cubes with the number of order $o(n)$. In addition, GridCART stores data in an ordered form; thus, no sorting is required, which eliminates a log.

6 Beyond Binary Classification

In this section, we extend our GridCART to multi-class classifications and regression settings.

6.1 Multiclass Classification

We here consider the multi-class classification, e.g., $\mathcal{Y} = \{1, 2, \dots, m\}$ for $m \in \mathbb{N}^+$. To begin with, we first define the conditional G-Influence for multi-class classification.

Definition 4 (Conditional G-Influence for class c)**.** *Suppose that* $\mathbf{X}$ *is drawn from a product distribution. Conditional Influence of function* $f : \mathcal{X} \to \mathcal{Y}$ *on variable* $\mathbf{X}^{(k)}$ *w.r.t. class* c *at leaf* t *is defined by*

$$\mathrm{Inf}_k^G[f, c \mid t] \triangleq \mathbb{E}_{\hat{\mathbf{p}}^{(j\neq k)}}\left[G\Big(\Pr_{\hat{\mathbf{p}}^{(k)}}\big[\hat{f}(\mathbf{X}) = c \mid t\big]\Big)\right],$$

where we here abbreviate $\mathbb{E}_{\hat{\mathbf{p}}^{(j\neq k)}}\Big[\mathbb{I}\{\hat{f}(\mathbf{X}) = c\} \mid \mathbf{X} \in t\Big]$ *as* $\Pr_{\hat{\mathbf{p}}^{(j\neq k)}}\big[f(\mathbf{X}) = c \mid t\big]$ *for simplicity.*

With the definition of Influence at hand, the extension from binary classification to multi-class classification is simple. We first define the impurity measure as follows:

$$\mathcal{G}(t) = \frac{1}{pm}\sum_{k=1}^{p}\sum_{c=1}^{m}\mathrm{Inf}_k^G[f, c \mid t] .$$

Then, every leaf node is split by solving

$$\max_{k,s} \mathcal{G}(t) - w_{t_L}\mathcal{G}(t_L) - w_{t_R}\mathcal{G}(t_R) .$$

Note that when $m = 2$, by using the fact that $G(x) = G(1-x)$ for all impurity function G, we have

$$\begin{aligned}
\mathrm{Inf}_k^G[f, 1 \mid t] &= \mathbb{E}_{\hat{\mathbf{p}}^{(j\neq k)}}\left[G\Big(\Pr_{\hat{\mathbf{p}}^{(k)}}\big[f(\mathbf{X}) = 1 \mid t\big]\Big)\right] \\
&= \mathbb{E}_{\hat{\mathbf{p}}^{(j\neq k)}}\left[G\Big(1 - \Pr_{\hat{\mathbf{p}}^{(k)}}\big[f(\mathbf{X}) = 2 \mid t\big]\Big)\right] \\
&= \mathbb{E}_{\hat{\mathbf{p}}^{(j\neq k)}}\left[G\Big(\Pr_{\hat{\mathbf{p}}^{(k)}}\big[f(\mathbf{X}) = 2 \mid t\big]\Big)\right] \\
&= \mathrm{Inf}_k^G[f, 2 \mid t] ,
\end{aligned}$$

which implies that the definition of impurity measure recovers the binary classification setting.

Similarly, the consistency rate for binary classification can be extended to multi-class classification cases. The key step is to pre-define the Bayes error and the target function, and then the extension is natural, which is detailed in Appendix F for anyone interested.

6.2 Regression

We here consider the regression task, e.g., $\mathcal{Y} = [-M, M]$ for some $M > 0$. Similarly, we first formulate the conditional Influence for regression tasks.

Definition 5 (Conditional Influence for regression)**.** *Let* $\hat{p}$ *and* f *be a product distribution and a mapping from* $\mathcal{X}$ *to* $\mathcal{Y} = \mathbb{R}$*, respectively. The conditional Influence of function* f *on variable* $\mathbf{X}^{(k)}$ *is defined by*

$$\mathrm{Inf}_k[f \mid t] \triangleq \mathbb{E}_{\hat{\mathbf{p}}^{(j\neq k)}}\left[\mathrm{Var}_{\hat{\mathbf{p}}^{(k)}}\left[\hat{f}(\mathbf{X}) \mid \mathbf{X} \in t\right]\right],$$

where we choose variance as the impurity measure, which is widely used in regression tasks [Breiman et al., 1984]. For regression tasks, we replace the criterion in Algorithm 3 [line 10] as follows:

$$(k_0, s_0) \leftarrow \arg\max_{k,s} \Delta_{\mathrm{reg}}^{\mathrm{Inf}}(t, k, s) ,$$

where $\Delta_{\mathrm{reg}}^{\mathrm{Inf}}(t, k, s) \triangleq \mathrm{Inf}_k[f \mid t] - w_{t_L}\mathrm{Inf}_k[f \mid t_L] - w_{t_R}\mathrm{Inf}_k[f \mid t_R]$. Besides, the histogram method output $g_n^{\mathrm{hist}} \leftarrow \hat{\eta}(c)$ in Algorithm 2 [line 5].

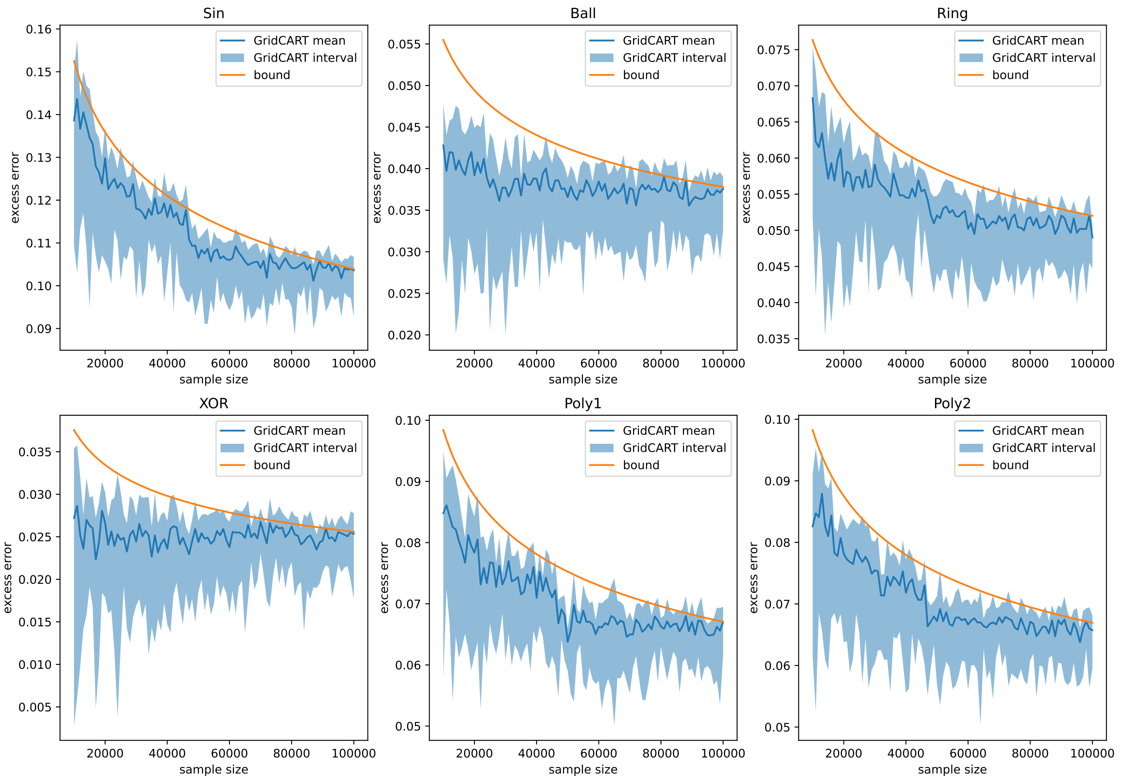


Figure 3: Comparison of excess errors for GridCART trained with increasing training sample sizes and our error bounds. The blue curves are the mean empirical excess errors (test error minus Bayes error), and the blue regions are filled between the minimal the maximal values among multiple repeats. The orange curves are the error bound we presented in Theorem 8. The order of excess error nearly matches our presented upper bound in many cases.

Theorem 9 (Consistency rate for regression)**.** *Suppose that Assumption 1-3 hold and $\mathcal{Y} = [-M, M]$ for some $M > 0$. The excess error of tree T_{K_n} generated by GridCART for regression has the following upper bound:*

$$\mathbb{E}_{D_n}[R(T_{K_n})] - R^\star = \mathcal{O}\left(M\left(\frac{1}{K_n h_n^3} + h_n + \sqrt{\frac{1}{nh_n^p}}\right)\right).$$

Theorem 9 shows that GridCART for regression is consistent with order $\mathcal{O}(Mn^{-1/(d+2)})$. In contrast to Theorem 8 for binary classification, Theorem 9 shows that an extra constant M would be suffered for regression tasks, which coincides with the fact that the regression task becomes more difficult if the range of label space $\mathcal{Y}$ is wider.

7 Numerical Simulation

To corroborate our theoretical results, we present some numerical simulations. We choose many synthetic datasets and plot the decreasing errors as sample sizes increase. For each pair of datasets and sample sizes, we repeat the experiment ten times. The curves of excess errors as well as our error bound about training sample sizes are shown in Figure 3. Note that the upper bound we present is relevant to unknown constants so we have to rescale the upper bound to fit the mean values for the comparison.

As shown in Figure 3, the blue lines represent the mean value and the blue regions are filled between the minimal and maximal values. The shape of the two curves in most of the subplots can nearly match, implying that the order of our upper bound is not so far away from the worst-case error in practice. In the problems of Ball and XOR, the bound curve can not match the so much. This may be because our error bound is obtained based on the worst-case analysis, which can be too pessimistic in some cases. In another word, the two problems are not difficult for GridCART.

8 Conclusions

In this work, we first investigated the difficulty in proving the consistency of CART and disclosed that no impurity gain in the worst case leads to the hardness to obtain a nontrivial result. Motivated by this recognition, we proposed the GridCART, a slightly modified CART, and provided a consistency rate of order $\mathcal{O}(n^{-1/(d+2)})$, which is the first nontrivial consistency rate for the heuristic tree learning algorithm. Our result sheds light on a theoretical understanding of the success of this type of algorithm.

References

Jean-Yves Audibert and Alexandre B. Tsybakov. Fast learning rates for plug-in classifiers. *The Annals of Statistics*, 35(2):608 – 633, 2007.

Gérard Biau, Luc Devroye, and Gábor Lugosi. Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, 9(66): 2015–2033, 2008.

Guy Blanc, Jane Lange, and Li-Yang Tan. Provable guarantees for decision tree induction: The agnostic setting. In *Proceedings of the 37th International Conference on Machine Learning*, pages 941–949, 2020a.

Guy Blanc, Jane Lange, and Li-Yang Tan. Top-down induction of decision trees: Rigorous guarantees and inherent limitations. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference*, pages 44:1–44:44, 2020b.

Leo Breiman. Random forests. *Machine Learning*, 45(1): 5–32, 2001.

Leo Breiman, Jerome Harold Friedman, Richard Allen Oshlen, and Charles Joel Stone. *Classification and Regression Trees (1st ed.).* Routledge, 1984.

Carla E. Brodley and Paul E. Utgoff. Multivariate decision trees. *Machine Learning*, 19(1):45–77, 1995.

Alon Brutzkus, Amit Daniely, and Eran Malach. ID3 learns juntas for smoothed product distributions. In *Proceedings of the 33rd Conference on Learning Theory*, pages 902–915, 2020.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

Luc Devroye, László Györfi, and Gábor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1997.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

Amos Fiat and Dmitry Pechyony. Decision trees: More theoretical justification for practical algorithms. In *Proceedings of the 15th International Conference on Algorithmic Learning Theory*, pages 156–170, 2004.

Wei Gao and Zhi-Hua Zhou. Towards convergence rate analysis of random forests for classification. In *Advances in Neural Information Processing Systems 33*, pages 9300–9311, 2020.

Wei Gao, Fan Xu, and Zhi-Hua Zhou. Towards convergence rate analysis of random forests for classification. *Artificial Intelligence*, 313:103788, 2022.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on boolean functions (extended abstract). In *Proceedings of the 29th Annual Symposium on Foundations of Computer Science*, pages 68–80, 1988.

Adam Tauman Kalai and Shang-Hua Teng. Decision trees are PAC-learnable from most product distributions: A smoothed analysis. *ArXiv preprint*, arXiv:0812.0933, 2008.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, 2017.

Nathan Keller. On the influences of variables on boolean functions in product spaces. *Combinatorics, Probability and Computing*, 20(1):83–102, 2011.

Jason Matthew Klusowski. Universal consistency of decision trees in high dimensions. *ArXiv preprint*, arXiv:2104.13881, 2021.

John Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3(4): 319–342, 1989.

Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

John Ross Quinlan. Discovering rules by induction from large collections of examples. *Expert Systems in the Micro Electronics Age*, 1979.

John Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

John Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

Erwan Scornet, Gérard Biau, and Jean-Philippe Vert. Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741, 2015.

Eiji Takimoto and Akira Maruoka. Top-down decision tree learning as information based boosting. *Theoretical Computer Science*, 292(2):447–464, 2003.

Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer, 2009.

Paul E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.

Zhi-Hua Zhou and Ji Feng. Deep forest. *National Science Review*, 6(1):74–86, 2019.