AVERAGE SENSITIVITY OF DECISION TREE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

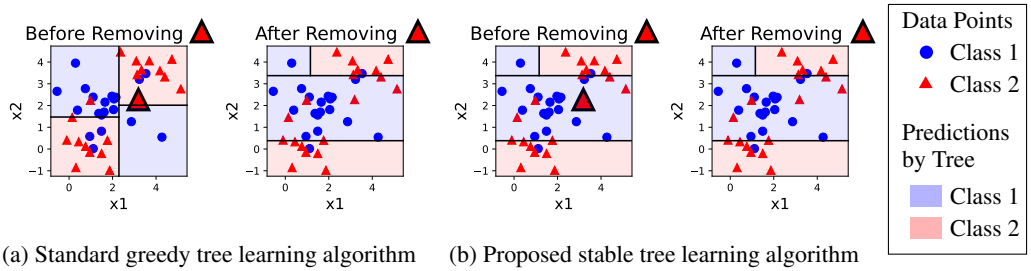
A decision tree is a fundamental model used in data mining and machine learning. In practice, the training data used to construct a decision tree may change over time or contain noise, and a drastic change in the learned tree structure owing to such data perturbation is unfavorable. For example, in data mining, a change in the tree implies a change in the extracted knowledge, which raises the question of whether the extracted knowledge is truly reliable or is only a noisy artifact. To alleviate this issue, we design decision tree learning algorithms that are stable against insignificant perturbations in the training data. Specifically, we adopt the notion of average sensitivity as a stability measure, and design an algorithm with low average sensitivity that outputs a decision tree whose accuracy is nearly equal to the optimal decision tree. The experimental results on real-world datasets demonstrate that the proposed algorithm achieves a low average sensitivity with an insignificant decrease in accuracy.

1 INTRODUCTION

A decision tree is a fundamental model in applications such as extracting knowledge in data mining and predicting outcomes in machine learning. Learned decision trees enable the extraction of hidden structures in the data in an interpretable manner using the if-then format. In data mining, the extracted structures are of fundamental interest (Rokach & Maimon, 2007; Gorunescu, 2011). Decision trees also play an essential role in decision making (Zeng et al., 2017; Rudin, 2019; Arrieta et al., 2020) because unlike complex models, such as deep neural networks, the decisions made by decision trees are explainable. With the increase of the utility of machine learning models in real-world problems, decision trees and their variants are widely used particularly for applications such as high-stake decision making, where explainability is crucial and transparency higher than post-hoc explanations (e.g., (Angelino et al., 2018; Rudin, 2019; Arrieta et al., 2020)) are required.

Current studies on decision trees and their families mainly focus on developing learning algorithms to improve two aspects of learned trees: accuracy and interpretability. Here, we demonstrate that there is a third essential aspect that is missing in current studies: the *stability* of the learning algorithm against insignificant perturbations on the training data. Decision trees are typically used to extract knowledge from data and help users make decisions that can be explained. If the learning algorithm is unstable, the structure of the learned trees can vary significantly even for insignificant changes in the training data. In data mining, this implies that the extracted knowledge can be unstable, which raises the question of whether the extracted knowledge is truly reliable or only a noisy artifact induced by the unstable learning algorithm. In model-based decision making, this implies that the decision process can change drastically whenever a few additional data are obtained and the tree is retrained on the new training data. Such noisy decision makers are unacceptable for several reasons. For example, stakeholders may lose their trust in such decision makers, or it may be extremely costly to frequently and drastically update the entire decision making system.

Figure 1 shows an illustrative example of sensitive/stable decision tree learning algorithms. In this example, the standard greedy tree learning algorithm induces different trees before and after one data point (large red triangle) is removed (Figure 1(a)). Thus, it can be observed that the greedy algorithm is sensitive to the removal of data points. The objective of this study is to design a tree learning algorithm that can induce (almost) same trees against the removal of a few data points (Figure 1(b)).



(a) Standard greedy tree learning algorithm (b) Proposed stable tree learning algorithm

Figure 1: Decision boundaries of the learned decision trees. (a) The standard greedy tree learning algorithm is sensitive to the removal of even a single data point (large red triangle) from the training data. (b) The proposed learning algorithm produces more stable trees.

In this study, we design a decision tree learning algorithm that is stable against insignificant perturbations in the training data. Specifically, we consider the change in (the distribution of) the learned tree upon deletion of a random data point from the training data, using the notion of *average sensitivity* (Varma & Yoshida, 2021). Subsequently, we design a (randomized) decision tree learning algorithm with low average sensitivity while preserving the accuracy of the learned decision tree up to a tolerance parameter.

A randomized algorithm may output completely different decision trees on the original training data and on the training data obtained by deleting a random data point even if the output distributions are close. To alleviate this issue, we design a (randomized) decision tree learning algorithm with low expected average sensitivity over random bits used in the algorithm, which implies that the output decision tree on the original training data and that on the training data obtained by deleting a random data point are close with a high probability over the choice of the random bits used.

Through real-world data experiments, we demonstrate that our learning algorithm exhibits a lower average sensitivity compared to the standard greedy decision tree learning algorithm, while maintaining the decrease in accuracy within the prescribed tolerance parameter.

2 Related Work

Decision Tree Learning Algorithms Generally, learning an optimal decision tree is NP-hard (Laurent & Rivest, 1976), and hence we can obtain optimal trees only for small problems (Bertsimas & Dunn, 2017; Angelino et al., 2018; Günlük et al., 2021). To avoid this issue, recursive greedy splitting is widely used for learning (non-optimal) decision trees (Rivest, 1987; Loh, 2011; Quinlan, 2014), and Bayesian approaches are used to learn a family of decision trees, such as rule lists and rule sets (Wang et al., 2017; Yang et al., 2017). These studies are concerned with learning trees with less computation or better interpretability. This study is orthogonal to them in that our interest is developing stable decision tree learning algorithms, which was not considered before. We stress that the focus of the current study is to learn a stand-alone decision tree and not to learn a collection of decision trees for ensemble models (e.g., (Ho, 1995; Breiman, 2001; Friedman, 2001; Chen & Guestrin, 2016)). For the latter, we want decision trees that make different predictions, and hence somewhat sensitive algorithms are favorable rather than stable ones.

Average Sensitivity Varma & Yoshida (2021) introduced the notion of average sensitivity and designed algorithms with low average sensitivity for various graph problems including the minimum spanning tree, minimum cut, and minimum vertex cover problems. Since then, average sensitivity of algorithms for various problems has been discussed, including the maximum matching problem (Yoshida & Zhou, 2021), problems that can be solved by dynamic programming (Kumabe & Yoshida, 2022a;b), spectral clustering (Peng & Yoshida, 2020), and Euclidean k-clustering (Yoshida & Ito, 2022).

Adversarial Robustness Insignificant human-imperceptible perturbations to the input can mislead trained models, and such perturbations are called *adversarial attacks*. It is known that adversarial attacks are harmful to decision trees (Chen et al., 2019a;b; Kantchelian et al., 2016). To alleviate this issue, several recent studies have considered the problems of robustness verification (Chen et al.,

2019b; Törnblom & Nadjm-Tehrani, 2019; Wang et al., 2020) and adversarial defense (Chen et al., 2019a; Andriushchenko & Hein, 2019; Calzavara et al., 2020; Chen et al., 2021). Adversarial attacks focus on the change in the predicted label against perturbing a single input data point at the inference time, whereas average sensitivity focuses on the change in the learned decision tree against perturbing the entire training data at the learning time.

Stability Bousquet and Elisseeff introduced the notion of the stability of a learning algorithm and discussed its relation to generalization ability (Bousquet & Elisseeff, 2002). Unlike average sensitivity, this notion only concerns the stability of the loss value against data perturbation. Hence, it cannot be used to stabilize the structure of learned decision tree.

Differential Privacy Differential privacy (Dwork, 2006) measures the stability of the output against perturbation to the input. Decision trees have been intensively studied from the perspective of differential privacy (see (Fletcher & Islam, 2019) and references therein). The average sensitivity of an algorithm can be bounded by the differential privacy parameter ϵ times the maximum size of the output (Varma & Yoshida, 2021). In the decision tree learning setting, the bound we obtain in this manner is roughly $O(\epsilon \cdot 2^B)$, where B specifies the depth of the output decision tree. By contrast, our bound (Theorem 4.1) is $O(B2^B/n)$, where n is the number of points in the training data. Our bound is much smaller because we usually set $2^B \ll n$ to avoid overfitting.

3 Preliminaries

3.1 Decision Tree

Let $\mathcal{X}$ and $\mathcal{Y}$ be input and output spaces, respectively. We call a Boolean function $\omega : \mathcal{X} \to \{0, 1\}$ a *decision rule*. Then, a *decision tree* is a function $\phi : \mathcal{X} \to \mathcal{Y}$ represented by a rooted proper binary tree, that is, it has a special node called the *root* and each node has either zero or two children, such that each internal node t is associated with a decision rule $\omega_t : \mathcal{X} \to \{0, 1\}$ and each leaf is associated with a label $y_t \in \mathcal{Y}$. Then given an input $x \in \mathcal{X}$, the decision tree ϕ predicts a label y according to PREDICT(ϕ, x) shown in Algorithm 1.

Let $\mathcal{L} = ((x_1, y_1), \dots, (x_n, y_n))$ be a training data. The *total score* of a decision tree $\phi : \mathcal{X} \to \mathcal{Y}$ with respect to $\mathcal{L}$ is defined to be $s(\phi, \mathcal{L}) := \sum_{i=1}^{n} 1[\phi(x_i) = y_i]$, where $1[X]$ is the indicator of the event X. Note that $s(\phi, \mathcal{L})/n$ is the accuracy of ϕ on $\mathcal{L}$.

For a nonnegative integer B, let $\mathcal{T}_B$ be the set of decision trees of depth B. Then, let $\mathrm{opt}_B(\mathcal{L}) := \max_{\phi \in \mathcal{T}_B} s(\phi, \mathcal{L})$ be the maximum total score of a decision tree of depth B. Also for a decision rule $\omega : \mathcal{X} \to \{0, 1\}$ and a nonnegative integer B, let $\mathcal{T}_{\omega,B}$ be the set of decision trees of depth B with the root node having the decision rule ω. Then we define $\mathrm{opt}_{\omega,B}(\mathcal{L}) := \max_{\phi \in \mathcal{T}_{\omega,B}} s(\phi, \mathcal{L})$. We clearly have $\mathrm{opt}_B(\mathcal{L}) = \max_\omega \mathrm{opt}_{\omega,B}(\mathcal{L})$.

For a decision tree ϕ, we denote by $|\phi|$ the number of nodes (including the leaves) in ϕ. For two decision trees ϕ and ϕ', we define the distance $d_{\mathrm{DT}}(\phi, \phi')$ between them as the output of DISTANCE(ϕ, ϕ') shown in Algorithm 2. Intuitively, the procedure DISTANCE(ϕ, ϕ') computes the maximal subtree common to ϕ and ϕ', and then outputs the total number of remaining nodes after subtracting the common subtree from ϕ and ϕ'. It is clear that $d_{\mathrm{DT}}(\cdot, \cdot)$ satisfies triangle inequality.

We note that, even if two decision trees $\phi, \phi' : \mathcal{X} \to \mathcal{Y}$ are equal as a function, they may have a large distance in d_{DT} if they have different tree structures. For interpretability, however, we believe this is the advantage of using d_{DT} because it is not easy to verify the equivalence of ϕ and ϕ' when they have different tree structures and d_{DT} correctly reflect this situation.

3.2 Average Sensitivity

For a training data $\mathcal{L} = ((x_1, y_1), \dots, (x_n, y_n))$ and $1 \leq i \leq n$, let $\mathcal{L}^{(i)} = ((x_1, y_1), \dots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), \dots, (x_n, y_n))$ be the training data obtained from $\mathcal{L}$ by dropping the ith data point.

Algorithm 1:

Procedure PREDICT(ϕ, x)
  Let t be the root node of ϕ;
  if *t is a leaf* **then return** y_t;
  else
    Let ϕ_L, ϕ_R be the decision trees rooted at the left and right children of t, respectively;
    if $\omega_t(x) = 0$ **then return** PREDICT(ϕ_L, x);
    else return PREDICT(ϕ_R, x);

Algorithm 2:

Procedure DISTANCE(ϕ, ϕ')
  Let t and t' be the root nodes of ϕ and ϕ', respectively;
  if *both t and t' are leaves* **then**
    return 0 if $y_t = y_{t'}$ and 2 otherwise.
  else if *either t or t' is a leaf* **then return** $|\phi| + |\phi'|$;
  else if $\omega_t \neq \omega_{t'}$ **then return** $|\phi| + |\phi'|$;
  else
    Let ϕ_L, ϕ_R be the decision trees rooted at the left and right children of t, respectively;
    Let ϕ'_L, ϕ'_R be the decision trees rooted at the left and right children of t', respectively;
    return DISTANCE(ϕ_L, ϕ'_L) + DISTANCE(ϕ_R, ϕ'_R).

Let A be a deterministic algorithm that, given a training data, outputs a decision tree. Then, the *average sensitivity* of A on $\mathcal{L}$ is

$$\frac{1}{n}\sum_{i=1}^{n} d_{\mathrm{DT}}(A(\mathcal{L}), A(\mathcal{L}^{(i)})).$$

Now, we generalize this definition for randomized algorithms. First for two distributions $\mathcal{D}$ and $\mathcal{D}'$ over decision trees, we define their *earth mover's distance* as $d_{\mathrm{EM}}(\mathcal{D}, \mathcal{D}') = \min_{\mathcal{P}} \mathbf{E}_{(\phi,\phi')\sim\mathcal{P}}\, d_{\mathrm{DT}}(\phi, \phi')$, where $\mathcal{P}$ is over distributions of pairs of decision trees such that the marginal distributions on the first and second coordinates are equal to $\mathcal{D}$ and $\mathcal{D}'$, respectively. Let A be a randomized algorithm that, given a training data, outputs a decision tree. Then, we define the *average sensitivity* of A on $\mathcal{L}$ as

$$\frac{1}{n}\sum_{i=1}^{n} d_{\mathrm{EM}}(A(\mathcal{L}), A(\mathcal{L}^{(i)})), \tag{1}$$

where we regard $A(\cdot)$ as a distribution over decision trees.

It is natural to consider a variant of average sensitivity such that we delete k data points instead of a single data point because in practice many data points can be dropped. As discussed in Varma & Yoshida (2021), this variant can be bounded by k times the average sensitivity above, and hence we focus on bounding the latter.

4 DECISION TREE CONSTRUCTION

In this section, we provide a decision tree learning algorithm with a high total score and a low average sensitivity. Specifically, we show the following:

Theorem 4.1. *There exists an (possibly inefficient) randomized algorithm that, given a training data $\mathcal{L}$ of size n, a depth bound B, and a parameter $\epsilon > 0$, returns a decision tree ϕ of depth at most B such that $\mathbf{E}_{\phi}[s(\phi, \mathcal{L})] \geq (1-\epsilon)^{B}\mathrm{opt}_B(\mathcal{L})$, and for the set of decision rules Ω, its average sensitivity is $O\left(\frac{B2^B \log|\Omega|}{\epsilon n}\right)$.*

Algorithm 3:

Procedure STABLEDR($\mathcal{L}, B, \epsilon$)
  Select $\omega \in \Omega$ with probability $\propto \exp(\lambda \cdot \mathrm{opt}_{\omega,B}(\mathcal{L}))$, where $\lambda = \frac{2 \log |\Omega|}{\epsilon \cdot \mathrm{opt}_B(\mathcal{L})}$;
  return ω.

Procedure STABLEDT'($\mathcal{L}, B, \epsilon, d$)
  if $|\mathcal{L}| \leq 1$ *or* $d = B$ **then return** an optimal label for $\mathcal{L}$;
  $\omega \leftarrow$ STABLEDR($\mathcal{L}, B, \epsilon$);
  Partition $\mathcal{L}$ into $\mathcal{L}_L \cup \mathcal{L}_R$ according to ω;
  $\phi_L \leftarrow$ STABLEDT'($\mathcal{L}_L, B, \epsilon, d+1$) and $\phi_R \leftarrow$ STABLEDT'($\mathcal{L}_R, B, \epsilon, d+1$);
  Let ϕ_ω be the decision tree such that the root node t has rule ω and the left and right children of t are ϕ_L and ϕ_R, respectively;
  return ϕ_ω.

Procedure STABLEDT($\mathcal{L}, B, \epsilon$)
  return STABLEDT'($\mathcal{L}, B, \epsilon, 0$).

We note that we usually choose $2^B \ll n$ to avoid overfitting and $\epsilon = \Theta(1/B)$ to achieve constant approximation, and hence the average sensitivity bound is eseentially $O(1/n)$. We note that the algorithm of Theorem 4.1 makes use of the optimal total score, which is not efficiently computable in general. We discuss practical implementations of the algorithm in Section 6.

The proposed algorithm is based on a procedure called STABLEDR, which selects a decision rule in a stable way (Algorithm 3). This is a simple application of the exponential mechanism (McSherry & Talwar, 2007) to $\mathrm{opt}_{\omega,B}(\mathcal{L})$ ($\omega \in \Omega$). Then, the proposed algorithm, STABLEDT (Algorithm 3), works as follows. At each node, we compute a decision rule ω using STABLEDR, split the training data according to ω, and recursively construct decision trees for each of the split data.

5 EXPECTED DETERMINISTIC AVERAGE SENSITIVITY

Recall that the average sensitivity (1) for randomized algorithms bounds the earth mover's distance between the distributions of $A(\mathcal{L})$ and $A(\mathcal{L}^{(i)})$. This does not immediately imply that, given a decision tree for $\mathcal{L}$, we can compute a similar decision tree for $\mathcal{L}^{(i)}$ with a similar total score because the mapping from decision trees for $\mathcal{L}$ to those for $\mathcal{L}^{(i)}$ that achieves the earth mover's distance is not always available. To address this issue, we consider bounding the expectation of deterministic average sensitivity between output decision trees over random bits:

$$\mathop{\mathbf{E}}_{\pi}\left[\frac{1}{n}\sum_{i=1}^{n} d_{\mathrm{DT}}(A_\pi(\mathcal{L}), A_\pi(\mathcal{L}^{(i)}))\right], \tag{2}$$

where A_π is the *deterministic* algorithm obtained from a randomized algorithm A by fixing the random bits used in A to $\pi \in \{0,1\}^*$. Then, using the same random bits π, we can obtain similar decision trees for $\mathcal{L}$ and $\mathcal{L}^{(i)}$. Note that the average sensitivity (1) is bounded from above by the expected deterministic average sensitivity (2) because the pair of random variables $(A_\pi(\mathcal{L}), A_\pi(\mathcal{L}^{(i)}))$ induces a joint distribution over pairs of decision trees.

By slightly modifying STABLEDT and the proof of Theorem 4.1, we obtain the following:

Theorem 5.1. *There exists a (possibly inefficient) randomized algorithm that, given a training data $\mathcal{L}$ of size n, a depth bound B, and a parameter $\epsilon > 0$, returns a decision tree ϕ of depth at most B such that $\mathbf{E}_\phi[s(\phi, \mathcal{L})] \geq (1-\epsilon)^B \mathrm{opt}_B(\mathcal{L})$, and for the set of decision rules Ω, its expected average sensitivity over random bits is $O\left(\frac{B 2^B \log |\Omega|}{\epsilon n}\right)$.*

6 PRACTICAL IMPLEMENTATIONS

The algorithm STABLEDR and the algorithm of Theorem 5.1 require the value of $\mathrm{opt}_{\omega,B}(\mathcal{L})$, which is NP-hard to compute (Laurent & Rivest, 1976). Although several algorithms based on integer programming were proposed (Bertsimas & Dunn, 2017; Günlük et al., 2021), they are not fast enough

	Dataset	training data size	test data size	# of features	# of classes	tree depth
small	breast cancer	546	137	10	2	5
	diabetes	614	154	8	2	1
large	cod-rna	59535	271617	8	2	10
	covtype	400000	181000	54	7	7
	higgs	10500000	500000	28	2	3
	ijcnn	49900	91701	22	2	1
	sensorless	48509	10000	48	11	9
	webspam	300000	50000	254	2	7

Table 1: Datasets

to handle large data. To address this issue, we suggest replacing $\mathrm{opt}_{\omega,B}(\mathcal{L})$ with $\mathrm{opt}_{\omega,1}(\mathcal{L})$ because the latter can be computed quite efficiently. Note that this strategy is the same as the standard greedy recursive splitting used for training (non-optimal) decision trees (Rivest, 1987; Loh, 2011; Quinlan, 2014). We use this version in our experiments in Section 7. The same analysis goes through and we obtain the following.

Theorem 6.1. *There exists a randomized algorithm that, given a training data $\mathcal{L}$ of size n, a depth bound B, and a parameter $\epsilon > 0$, returns a decision tree ϕ of depth at most B such that $\mathbf{E}[s(\phi, \mathcal{L})] \geq (1-\epsilon)\mathrm{opt}_1(\mathcal{L})$, and for the set of decision rules Ω, its expected average sensitivity over random bits is $O\left(\frac{B2^B \log|\Omega|}{\epsilon n}\right)$. The time complexity is $\sum_{d=0}^{\lceil \log n \rceil - 1} 2^B \cdot T(n/2^B) \cdot |\Omega|$, where $T(m)$ is the running time required to compute $\mathrm{opt}_{\omega,1}(\mathcal{L}')$ for $\omega \in \Omega$ with $|\mathcal{L}'| = m$.*

We note that, although the approximation guarantee of Theorem 6.1 is weaker than those of Theorems 4.1 and 5.1, practically used decision tree learning algorithms also lack approximation guarantees. In Section 7, we empirically confirm that the accuracy of the algorithm of Theorem 6.1 is not much worse than those of baseline algorithms. We also note that, in many cases, computing $\mathrm{opt}_{\omega,1}(\mathcal{L}')$ takes linear time, and sometimes we can compute $\mathrm{opt}_{\omega,1}(\mathcal{L}')$ for many ω's at once.

7 Experiments

In this section, we demonstrate that the proposed algorithm can output stable decision trees using real-world dataset experiments.

7.1 Setups

Datasets We used datasets shown in Table 1.[1] We split the datasets into small ones (breast cancer and diabetes) and large ones (cod-rna, covtype, higgs, ijcnn, sensorless, and webspam). We use small datasets to demonstrate the stability of the proposed algorithm, and large datasets to study trade-offs between average sensitivity and accuracy. In the experiments, we used subsamples of these datasets. For training, we randomly sampled 80% of the data points and 1000 data points for small and large datasets, respectively. In the experiments, we evaluated the test accuracy of the learned decision trees using the entire test data.

Tree Learning Algorithms In the experiment, we used the practical implementation described in Section 6 for training the decision trees. As we have shown in Theorem 6.1, this practical implementation can output stable decision trees without solving NP-hard problems. We also adopted the following greedy tree learning algorithm as the baseline for comparison. At each node, the greedy algorithm selects the decision rule ω with the highest score, i.e., $\omega \in \arg\max_{\omega\in\Omega} \mathrm{opt}_{\omega,1}(\mathcal{L})$. When multiple decision rules exist in $\arg\max_{\omega\in\Omega} \mathrm{opt}_{\omega,1}(\mathcal{L})$, we select one of them uniformly at random. We set the tree depth shown in Table 1 so that the greedy algorithm exhibits the highest accuracy in cross-validation.

[1]These datasets are obtained from `https://github.com/chenhongge/RobustTrees` We omitted MNIST-based datasets because decision trees are typically not used for images.

We implemented both the greedy and proposed algorithms in Python 3 using the JIT compiler of Numba. We used the equidistant points $\{x_{j,1}, \ldots, x_{j,Q}\}$ within the interval $[x_{j,\min}, x_{j,\max}]$ for each feature x_j as the decision rules, where $x_{j,\min} = x_{j,1}$ and $x_{j,\max} = x_{j,Q}$ are the minimum and the maximum of x_j in the dataset and we set $Q = 500$. That is, we set $\Omega = \{\omega(u) = 1[u_j \leq x_{j,q}]\}_{j,q}$,

Procedure We generated 10 sampled training data from the original training data. For each of the sampled training data, we trained decision trees using the greedy algorithm and proposed algorithms over different values of ϵ. Because the tree learning algorithms are randomized, we trained trees for ten different random bits $\pi_1, \pi_2, \ldots, \pi_{10}$. Through this procedure, we obtained 100 trees for each tree learning algorithm (over ten sampled training data and ten different random bits). We report the average of the average sensitivity and the training and test accuracy over these 100 trees. To estimate the average sensitivity, we trained additional trees using the sampled training data, with m data points removed at random. Formally, let $A(\mathcal{L})$ be the tree trained using the algorithm A on $\mathcal{L}$. Then, we consider another tree $A(\mathcal{L} \setminus \mathcal{S})$ trained on the set $\mathcal{L} \setminus \mathcal{S}$ with $|\mathcal{S}| = m$. We selected the sets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_R$ at random, and estimated the (normalized) expected deterministic average sensitivity over random bits as follows.

$$\frac{1}{R} \sum_{r=1}^{R} \frac{d_{\mathrm{DT}}(A_{\pi_t}(\mathcal{L}), A_{\pi_t}(\mathcal{L} \setminus \mathcal{S}_r))}{d_B}, \tag{3}$$

where π_t is the t-th random bits, and $d_B = 2^{B+2} - 2$ is the maximum distance between the trees with depth B, which normalizes the sensitivity within $[0, 1]$. We interpret this normalized sensitivity as the fraction of different nodes between the two trees. We set $R = 100$, and varied the number m of removed data points from 1 to 30% of the sampled training data.

7.2 Result 1: Dependency on ϵ

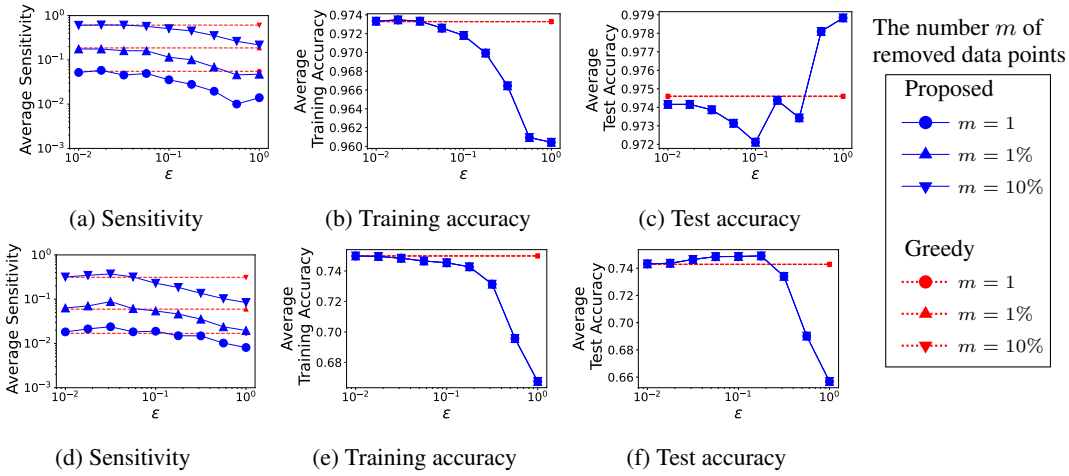
First, we evaluated the dependency of the proposed algorithm on the choice of ϵ for small datasets. In the experiment, we evaluated the algorithm over nine different ϵ values from 10^{-2} to 10^{0} on a logarithmic scale and over the number of removed data points $m = 1$, 1%, and 10% of the sampled training data. Figure 2 shows the average sensitivity, training accuracy, and test accuracy. It is evident from Figures 2 (a) and (d) that the average sensitivity decreases as ϵ increases as suggested by Theorem 6.1. Empirically, we observed that the average sensitivity is kept almost constant until $\epsilon = 0.1$ and it began to decrease for larger ϵ values. Figures 2 (b) and (e) show that the training accuracy decreases for ϵ larger than 0.1. These results suggest that the stability of the trees is obtained at the cost of the decrease in training accuracy. This observation is consistent with Theorem 6.1 as well. Based on these results, we can empirically confirm the correctness of Theorem 6.1: the trained decision trees become stable, particularly for larger ϵ, whereas the accuracy decrease. We also note that Figures 2(c) and (f) suggest that this may not always be the case for test accuracy, which is outside the scope of our theorem.

7.3 Result 2: Examples of Trees

Figure 3 shows examples of the trees trained on the breast cancer dataset using the greedy and the proposed algorithms with $\epsilon = 0.3$ and $m = 10\%$ of the sampled training data. In the experiment, we trained 100 trees on the set $\mathcal{L} \setminus \mathcal{S}_r$ for $r = 1, \ldots, 100$ to estimate the expected average sensitivity over random bits. Within these 100 trees, some had identical structures, while others did not. In the figure, we show the original tree trained using all the sampled training data, and the three most frequently identical structures within the 100 trees. In this example, the original tree and the most frequent trees were identical.

There are two important implications in Figure 3. Firstly, the second frequent trees of the greedy algorithm changed drastically from the original tree, whereas in the proposed algorithm, the changes appeared only in small substructures. These smaller changes in the proposed algorithm induced a smaller distance $d_{\mathrm{DT}}(A(\mathcal{L}), A(\mathcal{L} \setminus \mathcal{S}_r))$, resulting in smaller expected deterministic average sensitivity for $\epsilon > 0.1$, as shown in Figure 2.

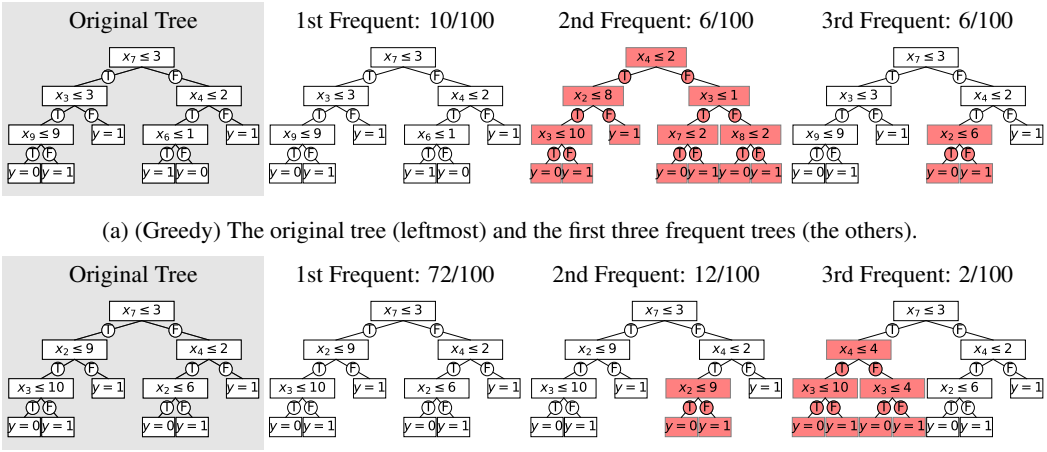
Secondly, the most frequent trees dominated 10/100 and 72/100 of the cases for the greedy and proposed algorithms, respectively. Therefore, the tree can change frequently upon data point removal



(a) Sensitivity (b) Training accuracy (c) Test accuracy

(d) Sensitivity (e) Training accuracy (f) Test accuracy

Figure 2: Average sensitivity and accuracy of the trained trees over different ϵ on small datasets, breast cancer (a)–(c) and diabetes (d)–(e). The blue and red lines denote the results for the proposed and greedy tree learning algorithms, respectively.



(a) (Greedy) The original tree (leftmost) and the first three frequent trees (the others).

(b) (Proposed) The original tree (leftmost) and the first three frequent trees (the others).

Figure 3: The three most frequent trees on the breast cancer dataset for (a) the greedy and (b) proposed tree learning algorithms over 100 random data point removals with $\epsilon = 0.3$ and $m = 10\%$. "T" and "F" on the edges denote the splitting whether the rule in the parent node is True and False, respectively. In both methods, the most frequent trees are identical to the original tree trained using all the sampled training data. The red nodes denote differences from the original tree.

in the greedy algorithm, whereas the change is less frequent in the proposed algorithm, where more than 70% of the trees are identical. These results indicate that the proposed algorithm can induce trees with small distances and less frequent structural changes. This property is favorable in practical situations because in data mining applications, the extracted knowledge is guaranteed to not change drastically, and in machine learning applications, we can continue using almost the same decision-making process.

7.4 Result 3: Sensitivity-Accuracy Trade-Off

The results in Figure 2 suggest that the average sensitivity and accuracy are in a trade-off relation through ϵ. Using a large ϵ, we obtain stable trees with a slight decrease in accuracy, where using a small ϵ, we obtain accurate trees; however they tend to have high sensitivities.

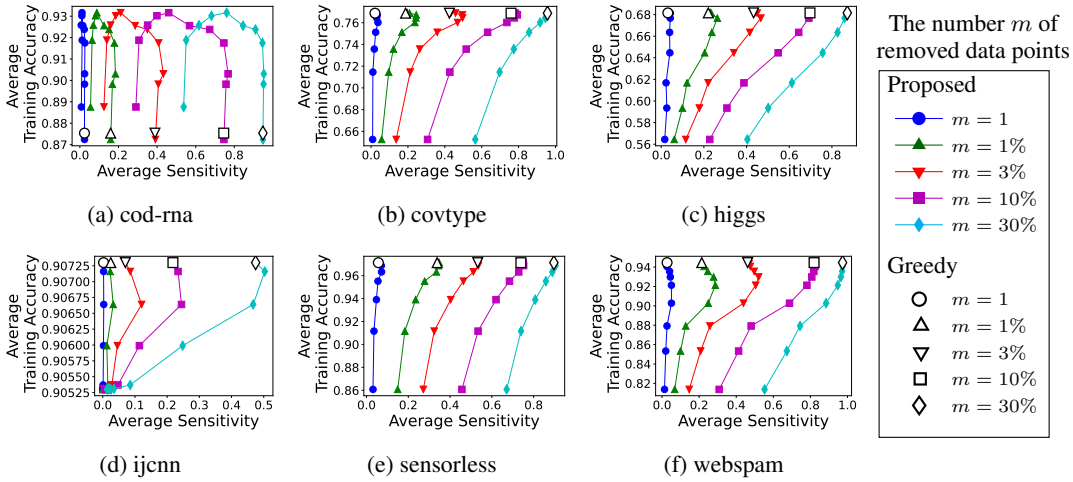


Figure 4: Trade-off curves between average sensitivity and training accuracy when ϵ is changed. We varied the number of training data points to be removed from one to 30% of the sampled training data. White markers denote the results of the greedy tree learning.

We computed this trade-off for large datasets by varying ϵ from 10^{-5} to 10^{0} on a logarithmic scale. Figure 4 shows the trade-off curves between the average sensitivity and training accuracy. In the figures, we observe that the average sensitivity and accuracy are in a trade-off relation for most of the datasets. The trees become stable as ϵ increases, while incurring an accuracy decrease in the bottom left of the figures. We note that our theory only guarantees that the decrease in training accuracy is limited. However, the test accuracy results in Figure 5 in Appendix D also confirm that the drop in test accuracy is limited. Thus, it is not necessary to sacrifice a considerable amount of the test accuracy to obtain stable decision trees in practice.

The result on cod-rna is an exception that did not exhibit clear trade-off curves. In this dataset, the trees become less stable and less accurate simultaneously for small ϵ, as shown in the bottom right of the figures. In particular, the greedy trees tend to be less accurate than stable ones. One possible reason of this phenomenon is overfitting. In Table 1, the tree depth is set to ten for this dataset. In such deep trees, the number of training data points that reach a deep node tends to be small. Then, the greedy selection of the best decision rule based on a small number of data points can be unstable, some arbitrary rules may be selected by chance, and overfitting occurs. Therefore, the greedily selected rule is not optimal if we consider deeper trees, i.e., $\mathrm{opt}_{\omega,B}(\mathcal{L})$, can be smaller than some $\mathrm{opt}_{\omega',B}(\mathcal{L})$ even if $\mathrm{opt}_{\omega,1}(\mathcal{L}) > \mathrm{opt}_{\omega',1}(\mathcal{L})$. On the contrary, the proposed algorithm randomizes the choice of the decision rule using an exponential mechanism. We conjecture that this randomization can help prevent suboptimal rules and overfitting, and increase the chance of gaining more accurate trees.

Finally, we emphasize that the results confirm that we gained the freedom to choose decision trees on these trade-off curves using the proposed algorithm. This will open up a new paradigm for practitioners who have suffered from the high sensitivity of existing learning algorithms. Practitioners can now tune ϵ and obtain stable trees upon their demand.

8 Conclusion

We proposed decision tree learning algorithms with theoretical guarantees on its average sensitivity and accuracy. We experimentally confirmed that the proposed algorithms performed well on real-world datasets. An obvious open problem is to show some hardness on the trade-off between average sensitivity and accuracy. In practice, it is important that the predictions made by the learned model do not change significantly with slight changes in the training data. Hence, it is also natural to measure the distance between models by the number of mismatches of their predictions and study average sensitivity in terms of this distance even for "non-explainable" learning models such as random decision forests Ho (1995); Breiman (2001) and deep neural networks.

REPRODUCIBILITY STATEMENT

In the experiments, we only used publicly available data so that all the results to be reproducible. Supplementary material contains both the proof details and the link to anatomized code repository.

REFERENCES

Maksym Andriushchenko and Matthias Hein. Provably robust boosted decision stumps and trees against adversarial attacks. *NeurIPS*, 32:13017–13028, 2019.

Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists for categorical data. *Journal of Machine Learning Research*, 18:1–78, 2018.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

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

Stefano Calzavara, Claudio Lucchese, Gabriele Tolomei, Seyum Assefa Abebe, and Salvatore Orlando. Treant: training evasion-aware decision trees. *Data Mining and Knowledge Discovery*, 34 (5):1390–1420, 2020.

Hongge Chen, Huan Zhang, Duane Boning, and Cho-Jui Hsieh. Robust decision trees against adversarial examples. In *ICML*, pp. 1122–1131, 2019a.

Hongge Chen, Huan Zhang, Si Si, Yang Li, Duane Boning, and Cho-Jui Hsieh. Robustness verification of tree-based models. *NeurIPS*, 32:12317–12328, 2019b.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pp. 785–794, 2016.

Yizheng Chen, Shiqi Wang, Weifan Jiang, Asaf Cidon, and Suman Jana. Cost-aware robust tree ensembles for security applications. In *USENIX Security*, pp. 2291–2308, 2021.

Cynthia Dwork. Differential privacy. In *ICALP*, pp. 1–12, 2006.

Sam Fletcher and Md Zahidul Islam. Decision tree classification with differential privacy: A survey. *ACM Computing Surveys (CSUR)*, 52(4):1–33, 2019.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, 29(5):1189–1232, 2001.

Florin Gorunescu. *Data mining: concepts, models and techniques*, volume 12. Springer Science & Business Media, 2011.

Oktay Günlük, Jayant Kalagnanam, Minhan Li, Matt Menickelly, and Katya Scheinberg. Optimal decision trees for categorical data via integer programming. *Journal of Global Optimization*, pp. 1–28, 2021.

Tin Kam Ho. Random decision forests. In *ICDAR*, pp. 278–282, 1995.

Alex Kantchelian, J Doug Tygar, and Anthony Joseph. Evasion and hardening of tree ensemble classifiers. In *ICML*, pp. 2387–2396, 2016.

Soh Kumabe and Yuichi Yoshida. Average sensitivity of dynamic programming. In *SODA*, pp. 1925–1961, 2022a.

Soh Kumabe and Yuichi Yoshida. Average sensitivity of the knapsack problem. In *ESA*, volume 244, pp. 75:1–75:14, 2022b.

Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17, 1976.

Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pp. 94–103, 2007.

Pan Peng and Yuichi Yoshida. Average sensitivity of spectral clustering. In *KDD*, pp. 1132–1140, 2020.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

Ronald L Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

Lior Rokach and Oded Z Maimon. *Data mining with decision trees: theory and applications*, volume 69. World scientific, 2007.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

John Törnblom and Simin Nadjm-Tehrani. An abstraction-refinement approach to formal verification of tree ensembles. In *SAFECOMP*, pp. 301–313, 2019.

Nithin Varma and Yuichi Yoshida. Average sensitivity of graph algorithms. In *SODA*, pp. 684–703, 2021.

Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. A bayesian framework for learning rule sets for interpretable classification. *The Journal of Machine Learning Research*, 18(1):2357–2393, 2017.

Yihan Wang, Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. On lp-norm robustness of ensemble decision stumps and trees. In *ICML*, pp. 10104–10114, 2020.

Hongyu Yang, Cynthia Rudin, and Margo Seltzer. Scalable bayesian rule lists. In *ICML*, pp. 3921–3930, 2017.

Yuichi Yoshida and Shinji Ito. Average sensitivity of euclidean k-clustering. In *NeurIPS*, 2022. to appear.

Yuichi Yoshida and Samson Zhou. Sensitivity analysis of the maximum matching problem. In *ITCS*, pp. 58:1–58:20, 2021.

Jiaming Zeng, Berk Ustun, and Cynthia Rudin. Interpretable classification models for recidivism prediction. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 180(3):689–722, 2017.