Fast Optimization of Weighted Sparse Decision Trees for use in Optimal Treatment Regimes and Optimal Policy Design

Ali Behrouz
University of British Columbia
Vancouver, British Columbia, Canada
alibez@cs.ubc.ca

Mathias Lécuyer
University of British Columbia
Vancouver, British Columbia, Canada
mathias.lecuyer@ubc.ca

Cynthia Rudin
Duke University
Durham, North Carolina, USA
cynthia@cs.duke.edu

Margo Seltzer
University of British Columbia
Vancouver, British Columbia, Canada
mseltzer@cs.ubc.ca

ABSTRACT

Sparse decision trees are one of the most common forms of interpretable models. While recent advances have produced algorithms that fully optimize sparse decision trees for *prediction*, that work does not address *policy design*, because the algorithms cannot handle weighted data samples. Specifically, they rely on the discreteness of the loss function, which means that real-valued weights cannot be directly used. For example, none of the existing techniques produce policies that incorporate inverse propensity weighting on individual data points. We present three algorithms for efficient sparse weighted decision tree optimization. The first approach directly optimizes the weighted loss function; however, it tends to be computationally inefficient for large datasets. Our second approach, which scales more efficiently, transforms weights to integer values and uses data duplication to transform the weighted decision tree optimization problem into an unweighted (but larger) counterpart. Our third algorithm, which scales to much larger datasets, uses a randomized procedure that samples each data point with a probability proportional to its weight. We present theoretical bounds on the error of the two fast methods and show experimentally that these methods can be two orders of magnitude faster than the direct optimization of the weighted loss, without losing significant accuracy.

CCS CONCEPTS

• **Computing methodologies** → *Classification and regression trees.*

KEYWORDS

Optimal Sparse Decision Trees; Interpretable Machine Learning; Explainability; Optimal Treatment Regimes.

1 INTRODUCTION

Sparse decision trees are a leading class of interpretable machine learning models that are commonly used for policy decisions [e.g., 12, 15, 38]. Historically, decision tree optimization has involved greedy tree induction, where trees are built from the top down [6, 13, 35], but more recently there have been several approaches that fully optimize sparse trees to yield the best combination of performance and interpretability [5, 16, 20, 33]. Optimization of sparse optimal trees is NP-hard, and many previous works have essentially leveraged the fact that the loss takes on a discrete number of values to provide a computational advantage [1, 2, 29, 31]. However, if one were to try to create a *policy tree* or estimate causal effects using one of these algorithms, it would become immediately apparent that such algorithms are not able to handle weighted data, because the weights do not come in a small number of discrete values. This means that common weighting schemes, such as inverse propensity weighting or simply weighting some samples more than others [8, 30], are not directly possible with these algorithms.

For example, let us consider developing a decision tree for describing medical treatment regimes. Here, the cost for misclassification of patients in different stages of the disease could be different. To create an optimal policy, we would weight the loss from each patient and minimize the sum of the weighted losses. While it is possible to approximately optimize this sum using CART's suboptimal greedy splitting procedures [6], there is not a way to do it with the current fastest optimal decision tree method, GOSDT [29].

We extend the framework of GOSDT-with-Guesses [31] to support weighted samples. GOSDT-with-Guesses produces sparse decision trees with closeness-to-optimality guarantees in seconds or minutes for most datasets; we refer to this algorithm as *GOSDTwG*. Our work introduces three approaches to allow weighted samples, where the first one is slow, the second one is fast, and the third one is fast and scales to large dataset sizes through the use of sampling.

In more detail, a key contributor to GOSDTwG's performance is its use of bitvectors to speed up the computation of the loss function. However, the introduction of weights requires a vector multiplication between the weights and this bitvector representation, which introduces a runtime penalty of one to two orders of magnitude. We demonstrate this effect in our first (direct) approach. Our second approach introduces a normalization and data duplication technique to mitigate the slowdown due to having real-valued weights. Here, we transform the weights to small integer values and then duplicate each sample by its transformed weights. In a third approach to this problem, which scales to much larger sample sizes, we propose a stochastic procedure, where we sample each data point with a probability proportional to its weight. Our experimental results show that: (1) the second and third techniques decrease run time by up to two orders of magnitude relative to that achieved by the slower direct weighted computation (the direct approach), (2) we can bound our accuracy loss from using the second approach rather than the first one; and (3) the proposed weighted optimal decision tree technique can outperform natural baselines in terms of running time, sparsity, and accuracy.

2 RELATED WORK

Decision trees are one of the most popular forms of interpretable models [36]. While full decision tree optimization is NP-hard [28], it is possible to make assumptions, e.g., feature independence, that simplify the hard optimization to cases where greedy methods suffice [23]. However, these assumptions are unrealistic in practice. Some other approaches [21, 32] assume that the data can be perfectly separated with zero error and use SAT solvers to find optimal decision trees; however, real data are generally not separable.

Recent work has addressed optimizing accuracy with soft or hard sparsity constraints on the tree size. Such decision tree optimization problems can be formulated using mixed integer programming (MIP) [2, 5, 20, 37, 42, 43], but MIP solvers tend to be slow. To improve the scalability of decision tree optimization, several studies have produced customized dynamic programming algorithms that incorporate branch-and-bound techniques. In particular, analytical bounds combined with bit-vector-based computation have been used to efficiently reduce the search space and improve run time [4, 7, 22]. Lin et al. [29] extend this approach to use dynamic programming, which leads to even more improved scalability. Demirović et al. [11] introduce constraints on both depth and the number of nodes to improve scalability. Recently, McTavish et al. [31] proposed smart guessing strategies, based on knowledge gleaned from black-box models, that can be applied to any optimal branch-and-bound-based decision tree algorithm to reduce the run time by multiple orders of magnitude. While these studies focus on improving running time and accuracy, they handle only uniform sample importance and do not consider weighted data points. Our work neatly fills this gap; our weighted objective function, data duplication method, and sampling approach enable us to find optimal decision trees for these problems quickly.

Several studies focus on learning tree- and list-based treatment regimes from data [9, 14, 24, 25, 40, 45, 47]. However, none of these methods fully optimize the policy because it was not known at the time how to perform optimization of the type we use in this work.

3 METHODOLOGY

Let $\{(\mathbf{x}_i, y_i, w_i)\}_{i=1}^N$ represent our training dataset, where $\mathbf{x}_i$ are M-vectors of features, $y_i \in \{0, 1, \dots, K\}$ are labels, $w_i \in \mathbb{R}^{\geq 0}$ is the weight associated with data $\mathbf{x}_i$, and N is the size of the dataset. Also, let $\mathbf{x}$ be the $N \times M$ covariate matrix, $\mathbf{w}$ be the N-vector of weights, and $\mathbf{y}$ be the N-vector of labels, and let x_{ij} denote the j-th feature of $\mathbf{x}_i$. To handle continuous features, we binarize them either by using all possible split points [22] to create dummy variables or by using a subset of these splits as done by McTavish et al. [31]. We let $\tilde{\mathbf{x}}$, the binarized covariate matrix, be notated as $\tilde{\mathbf{x}}_{ij} \in \{0, 1\}$.

3.1 Objective

Let $\mathcal{T}$ be a decision tree that gives predictions $\{\hat{y}_i^{\mathcal{T}}\}_{i=1}^N$. The weighted loss of the tree $\mathcal{T}$ on the training dataset is:

$$\mathcal{L}_{\mathbf{w}}(\mathcal{T}, \tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{\sum_{i=1}^N w_i} \sum_{i=1}^N \mathbb{1}[y_i \neq \hat{y}_i^{\mathcal{T}}] \times w_i \,. \tag{1}$$

To achieve interpretability and prevent overfitting, we provide the options to use either soft sparsity regularization on the number of leaves, hard regularization on the tree depth, or both [see 31]:

$$\underset{\mathcal{T}}{\text{minimize}} \;\; \mathcal{L}_{\mathbf{w}}(\mathcal{T}, \tilde{\mathbf{x}}, \mathbf{y}) + \lambda H_{\mathcal{T}} \quad s.t. \;\; \text{depth}(\mathcal{T}) \leq d \,, \tag{2}$$

where $H_{\mathcal{T}}$ is the number of leaves in the tree $\mathcal{T}$ and λ is a regularization parameter. We define $R_{\mathbf{w}}(\mathcal{T}, \tilde{\mathbf{x}}, \mathbf{y}) = \mathcal{L}_{\mathbf{w}}(\mathcal{T}, \tilde{\mathbf{x}}, \mathbf{y}) + \lambda H_{\mathcal{T}}$. We might refer to $\mathbb{1}[y_i \neq \hat{y}_i^{\mathcal{T}}]$ as $I_i(\mathcal{T})$, for simplicity. While in practice, depth constraints between 2 and 5 are usually sufficient, McTavish et al. [31] provide theoretically-proven guidance to select a depth constraint so that a single tree has the same expressive power (VC dimension) as an ensemble of smaller trees (e.g., a random forest or a boosted decision tree model). The parameter λ trades off between the weighted training loss and the number of leaves in the tree.

3.2 Learning Weighted Trees

We present three approaches for handling sample weights. The first is the *direct approach*, where we calculate the weighted loss directly. Implementing this approach requires multiplying each misclassification by its corresponding weight, which is computationally expensive in any algorithm that uses bitvectors to optimize loss computation. This overhead is due to replacing fast bitvector operations with slower vector multiplications. The direct approach slows GOSDTwG down by two orders of magnitude. To avoid this computational penalty, our second approach, *data-duplication*, involves a transformation of the weights; specifically, we normalize, scale, and round the weights to be small integer values. We then duplicate samples, where the number of duplicates is the value of the rounded weights, and use this larger unweighted dataset to learn the tree. This method avoids costly vector multiplications and does not substantially increase run time compared to the unweighted GOSDTwG; note that GOSDTwG scales extremely well with the sample size due to the bit-vector computations, so duplication does not add much to the computational cost. Finally, to scale to larger datasets, we present a randomized procedure, called *weighted sampling*, where we sample each data point with a probability proportional to its weight. This process introduces variance (not bias) and scales to large numbers of samples.

Direct Approach. We begin with the branch-and-bound algorithm of McTavish et al. [31] and adapt it to support weighted samples. Given a reference model T, they prune the search space using three "guessing" techniques: **(1)** guess how to transform continuous features into binary features, **(2)** guess tree depth for sparsity-regularized models, and **(3)** guess tight lower bounds on the objective for subsets of points to allow faster time-to-completion. It is straightforward to see that the first two techniques apply directly to our weighted loss function. However, we need to adapt the third guessing technique to have an effective and tight lower bound for the weighted loss function. Let $\hat{y}_i^T$ be the predictions of the reference model (perhaps a boosted decision tree model) on training observation i. Let s_a be the subset of training observations that satisfy a boolean assertion a:

$$\begin{aligned} s_a &:= \{i : a(\tilde{\mathbf{x}}_i) = \text{True}, i \in \{1, ..., N\}\} \\ \tilde{\mathbf{x}}(s_a) &:= \{\tilde{\mathbf{x}}_i : i \in s_a\} \\ \mathbf{y}(s_a) &:= \{y_i : i \in s_a\} \\ \mathbf{w}(s_a) &:= \{w_i : i \in s_a\} \,. \end{aligned}$$

Motivated by McTavish et al. [31], we define our guessed lower bound as follows:

$$lb_{\text{guess}}(s_a) := \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i \in s_a} \mathbb{1}[y_i \neq \hat{y}_i^T] \times w_i + \lambda \,. \tag{3}$$

Equation 3 is a lower bound guess for $R_{\mathbf{w}}(t, \tilde{\mathbf{x}}(s_a), \mathbf{y}(s_a))$ because we assume that the (possibly black box) reference model T has loss at most that of tree t on data s_a and we know that any tree has at least one node (hence the regularization term's lower bound of $\lambda \times 1$).

Accordingly, in the branch-and-bound algorithm, to optimize the weighted loss function introduced in Equation 2, we consider a subproblem to be solved if we find a subtree that achieves an objective less than or equal to its lb_{guess}. If we find such a subtree, our training performance will be at least as good as that of the reference model. For a subset of observations s_a, we let t_a be the subtree used to classify points in s_a, and H_{t_a} be the number of leaves in that subtree. We can then define the subset's contribution to the objective:

$$\begin{aligned} &R_{\mathbf{w}(s_a)}(t_a, \tilde{\mathbf{x}}(s_a), \mathbf{y}(s_a)) \\ &= \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i \in s_a} \mathbb{1}[y_i \neq \hat{y}_i^{t_a}] \times w_i + \lambda H_{t_a}. \end{aligned}$$

For any dataset partition A, where $a \in A$ corresponds to the data handled by a given subtree of t:

$$R_{\mathbf{w}}(t, \tilde{\mathbf{x}}, \mathbf{y}) = \sum_{a \in A} R_{\mathbf{w}(s_a)}(t_a, \tilde{\mathbf{x}}(s_a), \mathbf{y}(s_a)) \,.$$

By introducing the abovementioned lower bound guess, we can now replace the lower bound of McTavish et al. [31] with our lower bound and proceed with branch-and-bound. Their approach is provably close to optimal when the reference model makes errors similar to those made in an optimal tree. We now show that our approach using the weighted lower bound is also close to optimal. Let $s_{T,\text{incorrect}}$ be the set of observations incorrectly classified by the reference model T, i.e., $s_{T,\text{incorrect}} = \{i | y_i \neq \hat{y}_i^T\}$, and t_{g} be a tree returned from our lower-bound guessing algorithm. We have:

Theorem 1. *(Performance Guarantee). Let $R(t_g, \tilde{\mathbf{x}}, \mathbf{y})$ denote the objective of t_g on the full binarized dataset $(\tilde{\mathbf{x}}, \mathbf{y})$ for some per-leaf penalty λ. Then for any decision tree t that satisfies the same depth constraint d, we have:*

$$\begin{aligned} R(t_g, \tilde{\mathbf{x}}, \mathbf{y}) \leq &\frac{1}{\sum_{i=1}^{N} w_i} \Bigg(\sum_{i \in s_{T,\text{incorrect}}} w_i \\ &+ \sum_{i \in s_{T,\text{correct}}} \mathbb{1}[y_i \neq \hat{y}_i^t] \times w_i \Bigg) + \lambda H_t \,. \end{aligned}$$

That is, the objective of the guessing model t_g is no worse than the union of errors made by the reference model and tree t.

Hence, the model t_{g} from our lower bound guessing algorithm achieves a weighted objective that is as good as the error of the reference model (which should be small) plus (something smaller than) the error of the best possible tree of the same depth. The proof can be found in Appendix A.1.

Motivation for Data Duplication. While it might seem counter-intuitive that increasing the size of the dataset by replicating data would be substantially faster than the direct approach, it actually is. Let us consider the computation involved in these two options. In decision tree optimization, evaluation of the objective is performed repeatedly. Any small improvement in that evaluation step leads to a large improvement (possibly orders of magnitude) in the overall computational speed of the algorithm. In the direct approach, computing the objective (2) requires computing the inner product $\mathbf{w} \cdot \mathcal{I}$, where $\mathcal{I}_i = \mathbb{1}[y_i \neq \hat{y}_i^{\mathcal{T}}]$. In the unweighted case, as all weights are 1, this computation can be performed using bit operations, which are extremely fast. In the weighted case, we resort to standard inner products, which are two orders of magnitude slower (see Section 4). The data-duplication approach allows us to use bit-vectors as in the unweighted case, preserving fast computation.

Algorithm 1: Data Duplication

Input : Dataset $\mathbf{x}, \mathbf{y}$ and weights $\mathbf{w}$, duplication factor $p < 100$
Output: Duplicated dataset $\tilde{X}$, $\tilde{y}$

```
X̃ ← ∅; ỹ ← ∅;
Define w̃_i = round( p · ( w_i / Σ_{i=1}^N w_i ) );
for x_i ∈ x do
    for i = 1, 2, ..., w̃_i do
        X̃ ← X̃ ∪ {x_i};
        ỹ ← ỹ ∪ {y_i};
return X̃, ỹ
```

Data-duplication Algorithm. The data-duplication algorithm is shown in Algorithm 1. Given an integer $p > 0$, we first normalize all weights and scale them to $(0, 1]$. Then we multiply each normalized weight by p and round them to integers. Given the scaled integer weights, we then duplicate each sample, $\mathbf{x}_i$, by its corresponding integer weight, $\hat{w}_i$. Once the data are duplicated, *we can use any optimal decision tree technique.* In our experiments, we show that if we choose the value of p appropriately, this method can speed up the training process significantly without losing too much accuracy. After data-duplication, there are no longer weights associated with the samples, and we can use the fast bit-vector computations from the unweighted case. Even in the presence of substantially more data, since the bit-vector computation scales sublinearly with the number of samples, it is much faster than the direct approach.

Correctness of Data Duplication. One might ask whether the data duplication approach leads to suboptimal solutions because its loss function is an approximation to the weighted loss. As it turns out, as long as the weights do not change very much when rounding to integers, the minimum of the data duplication algorithm's objective is very close to the minimum of the original weighted objective.

Recall the objective

$$R(t) := \frac{1}{\sum_{i=1}^{N} w_i} \sum_i w_i I_i(t) + \lambda \#\text{leaves}.$$

Define the objective with the approximate weights as

$$\tilde{R}(t) := \frac{1}{\sum_{i=1}^{N} \tilde{w}_i} \sum_i \tilde{w}_i I_i(t) + \lambda \#\text{leaves}.$$

When we rounded, we ensured that the weights did not change much. That is, we know by design that $\|\mathbf{w} - \tilde{\mathbf{w}}\|_\infty \leq \epsilon$. Note that

multiplying w_is by a scalar cannot change the value of the objective function. Accordingly, normalizing and also scaling weights by p do not change the value of $R(t)$. Therefore, without loss of generality, we can assume that w_is are weights right before rounding.

Theorem 2. *Let t^* be a minimizer of the objective as $t^* \in \arg\min_t R(t)$, and $\tilde{\mathcal{T}}$ be a minimizer of the approximate loss function as $\tilde{\mathcal{T}} \in \arg\min_t \tilde{R}(t)$. If $\|\mathbf{w} - \tilde{\mathbf{w}}\|_\infty \leq \epsilon$, we have:*

$$|R(t^*) - \tilde{R}(\tilde{\mathcal{T}})| \leq max\left\{\frac{(\zeta-1)\psi+\epsilon}{\zeta}, \frac{(\eta-1)\psi+\epsilon}{\eta}\right\},$$

where $\eta = \max_{1\leq i\leq N}\left\{\frac{w_i}{\tilde{w}_i}\right\}$, $\zeta = \max_{1\leq i\leq N}\left\{\frac{\tilde{w}_i}{w_i}\right\}$, and $\psi = \frac{\max_i\{w_i,\tilde{w}_i\}}{\min_i\{w_i,\tilde{w}_i\}}$.

Note that, in other words, we provably will not lose substantial performance when using the rounded solution, as long as we did not change the weights very much when rounding. The value of η and ζ are usually small and near 1. If the value of ψ is very large, then the direct approach is more efficient, and we do not need to use data duplication. Accordingly, when we use data duplication, the value of ψ should also be small. The proof is in Appendix A.2.

Weighted Sampling. When the ratio of the biggest weight over the smallest weight is large, the data duplication approach might be inefficient, in that it could require us to create a huge number of samples. To address this issue, we present a stochastic sampling process as a pre-processing step. We sample each data point based on its weight. Given an arbitrary number r, we sample $S = r \times N$ data points such that the probability of choosing $\mathbf{x}_i$ is $\frac{\mathbf{w}_i}{\sum_{i=1}^N \mathbf{w}_i}$. After this step, we can use any unweighted optimal decision tree algorithm on the sampled dataset.

Quality Guarantee of Weighted Sampling. Let $\tilde{\mathcal{L}}(.)$ be the loss function on the sampled dataset, it is not hard to see that $\mathbb{E}[\tilde{\mathcal{L}}] = \mathcal{L}_{\mathbf{w}}$, where $\mathcal{L}_{\mathbf{w}}$ is the value of the misclassification (Eq. 1) on the weighted dataset. Based on this fact, we have the following theorem:

Theorem 3. *Given a weighted dataset $D = \{(\mathbf{x}_i, y_i, w_i)\}_{i=1}^N$, an arbitrary positive real number $r > 0$, an arbitrary positive real number $\varepsilon > 0$, and a tree $\mathcal{T}$, if we sample $S = r \times N$ data points from D, $\tilde{D} = \{(\tilde{\mathbf{x}}_i, \tilde{y}_i)\}_{i=1}^S$, we have:*

$$\mathbb{P}\left(|\tilde{\mathcal{L}}(\mathcal{T}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}}) - \mathcal{L}_{\mathbf{w}}(\mathcal{T}, \mathbf{x}, \mathbf{y})| \geq \varepsilon\right) \leq 2\exp\left(-\frac{2\varepsilon^2}{S}\right)$$

4 EXPERIMENTS

Our evaluation addresses the following questions:

(1) When is the direct approach more efficient than data-duplication and weighted sampling?
(2) In practice, how well do the second and third proposed methods perform relative to the direct approach?
(3) How sparse and fast are our weighted models relative to state-of-the-art optimal decision trees?
(4) How can our approach be used for policy making?

We use sparsity as a proxy for interpretability, because it can be quantified thus providing an objective means of comparision [36].

4.1 Datasets

In our experiments, we use seven publicly available real-world datasets; Table 1 shows sizes of these datasets: The Lalonde dataset [10, 26], Broward [44], the coupon dataset, which was collected on Amazon Mechanical Turk via a survey [46], Diabetes [39], which is a health care related dataset, the COMPAS [27], the Fair Isaac (FICO) credit risk dataset [17] from the Explainable ML Challenge, and Netherlands [41] datasets, which are recidivism datasets. Unless stated otherwise, we use inverse propensity score with respect to one of the features as our weights. For more details about datasets and weights see Appendix B.1.

Table 1: Datasets

Dataset	samples	features	binary features
Lalonde	723	7	447
Broward	1954	38	588
Coupon	2653	21	87
Diabetes	5000	34	532
COMPAS	6907	7	134
FICO	10459	23	1917
Netherlands	20000	9	53890

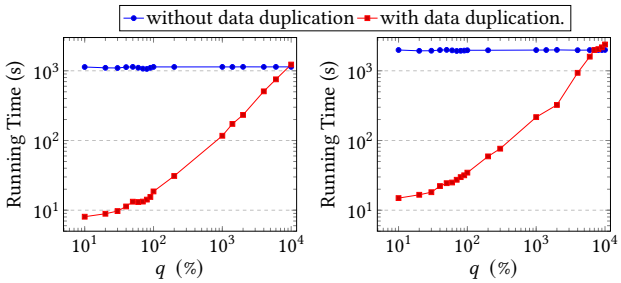

Figure 1: Training time of the model with and without data duplication on different machines.

For each dataset, we ran the experiments with different depth bounds and regularization, and each point in each plot shows the results for one setting. Table 2 (Appendix B.2) lists the configurations used for each dataset when training decision trees.

4.2 Baselines

We compared our proposed methods with the following baseline models: (1) CART [6], (2) DL8.5 [3], and (3) Gradient Boosted Decision Trees (GBDT) [18, 19]. CART and GBDT can both handle weighted datasets, so we use their default weighted implementation as the baselines. As DL8.5 does not supported weighted datasets, we use the data-duplication approach with it.

4.3 Results

Data duplication. We begin by demonstrating how much the direct approach penalizes runtime relative to the data-duplication approach. We use the unweighted FICO dataset and randomly pick q% of the data points, $\mathcal{S}$, to double weight, duplicate them and add them to the data set, producing a dataset of size $(1 + \frac{q}{100}) \times N$, where N is the size of the original data set, $|S|$. We then compare this to the running time of the direct approach in which we assign weight of 2 to all double-weighted samples and weight of 1 to all remaining samples. We run this experiment on two different machines, with different processors and RAMs, in order to show

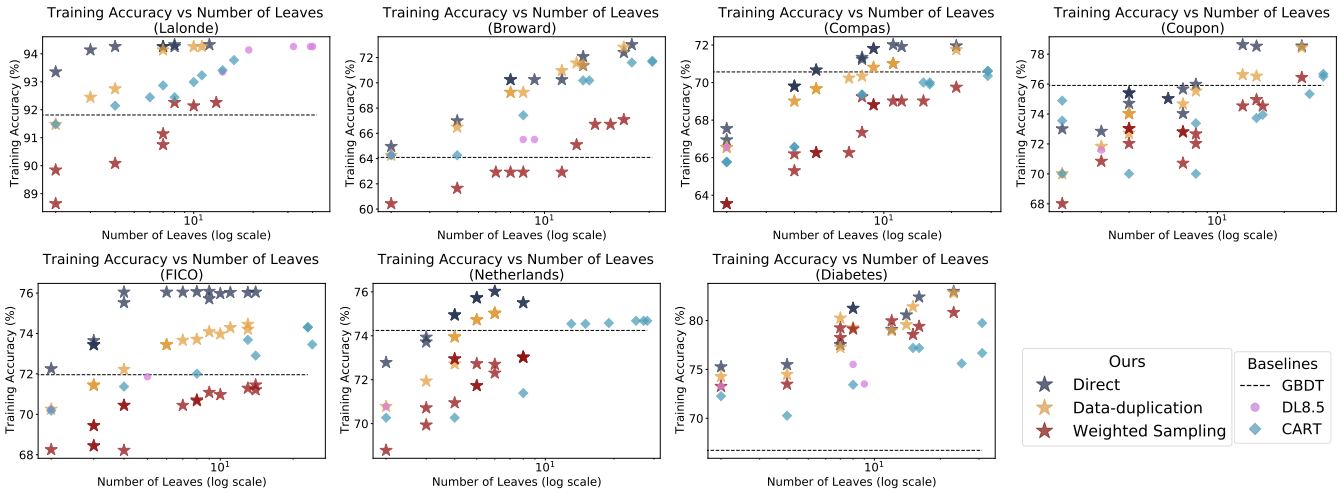

Figure 2: Sparsity vs. training accuracy: All methods but CART and GBDT use guessed thresholds. GBDT and DL8.5 use data duplication. DL8.5 frequently times out, so there are fewer markers for it. GOSDTwG achieves the highest accuracy for every level of sparsity.

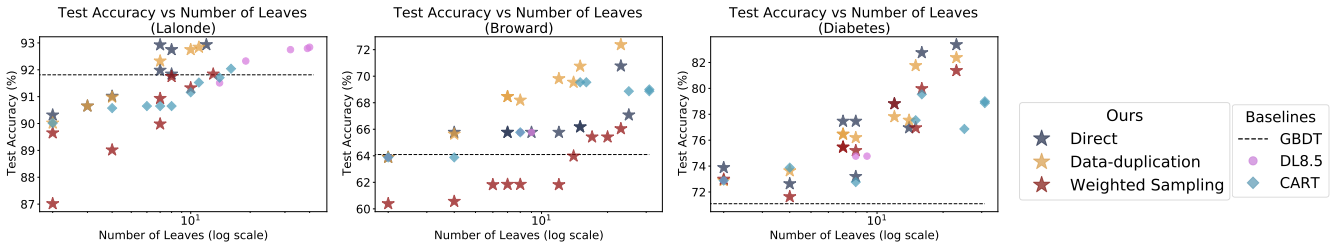

Figure 3: Sparsity vs. test accuracy: All methods but CART and GBDT use guessed thresholds. GBDT and DL8.5 use data duplication. DL8.5 frequently times out, so there are fewer markers for it. GOSDTwG achieves the highest accuracy for every level of sparsity.GOSDTwG also achieves the highest test accuracy for *almost* every level of sparsity.

the reliability of the results on different machines. The detailed property of machines can be found in Appendix B.3. Figure 1 shows the result of this experiment. We find that when the size of the duplicated dataset is less than 100 times the original dataset, the data-duplication approach is always faster.

Comparison of our approaches. We next compare the relative accuracy achieved using the direct, data-duplication, and weighted sampling approaches on all our evaluation datasets. The star-shaped points in Figures 2 and 3 show the result of this comparison. These results suggest a trade-off between accuracy and running time. Weighted sampling is the fastest approach, but it has the worst accuracy among our approaches because it uses only subsets of the data. Data duplication, while slower than weighted sampling, is faster than the direct method, without losing much accuracy.

Sparsity vs. accuracy. If we now consider the dotted line and round and diamond shapes in Figures 2 and 3, we can see the accuracy-sparsity tradeoff for different decision tree models (the black line represents accuracy for GBDT). GOSDTwG produces excellent training and test accuracy with a small number of leaves, and, compared to other decision tree models, achieves higher accuracy for every level of sparsity. Results of other datasets are in Appendix C.1.

Training time vs. accuracy. Figures 4 and 5 show the training time and accuracy for different decision tree models. While the training times of GOSDTwG and CART are almost the same, GOSDTwG achieves the highest training and test accuracy in almost all cases. DL8.5 struggled with the 1 hour termination condition for all datasets except Lalonde; because DL8.5 did not solve to optimality, it was outperformed by both CART and GOSDTwG. Results of other datasets are in Appendix C.2.

Lalonde Case Study. The Lalonde dataset is from the National Supported Work Demonstration [10, 26], a labour market experiment in which participants were randomized between treatment (on-the-job training lasting between nine months and a year) and control groups. Accordingly, for each unit U_i, we have a pre-treatment covariate vector X_i and observed assigned treatment Z_i. Let Y_i^1 be the potential outcome if unit U_i received the treatment, and Y_i^2 be the potential outcome if it was not treated. When a unit is treated, we do not observe the potential outcome if it was not treated and vice versa. To address this issue, we use the MALTS model [34] that estimates these missing values by matching. MALTS gives us an estimate of the conditional average treatment effect, which we call TE. Since our weighted decision tree is designed for classification, we classify participants into three groups—"should be treated," "should be treated if budget allows," and "should not be treated" -– based

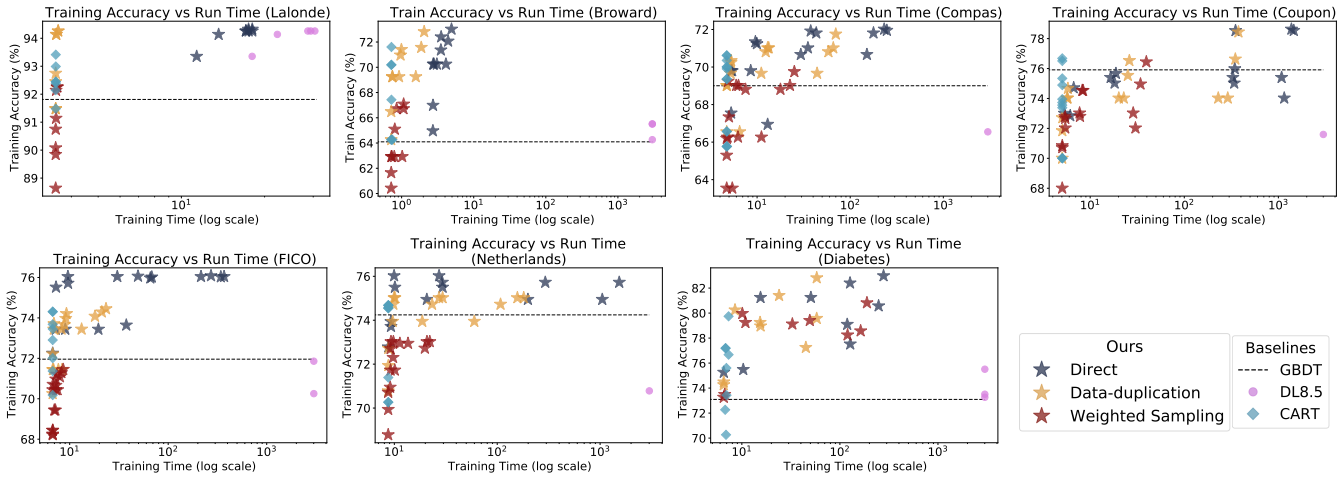

Figure 4: Training time vs. training accuracy: All methods but CART and GBDT use guessed thresholds. GBDT and DL8.5 use data duplication. DL8.5 frequently times out, so there are fewer markers for it. While CART is the fastest algorithm, GOSDTwGuses its additional runtime to produce models that produce higher accuracy.

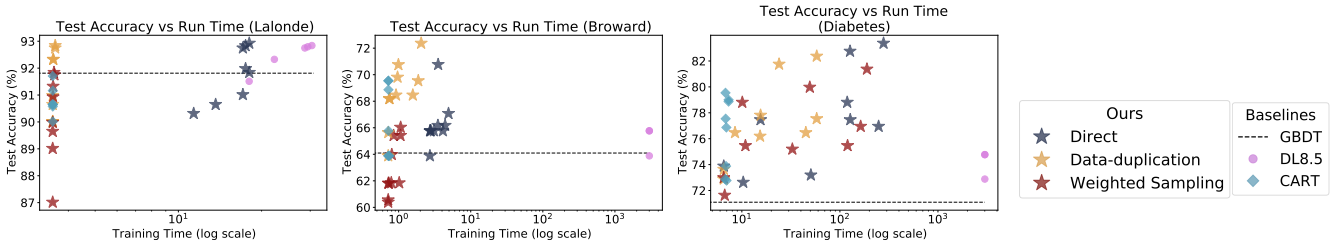

Figure 5: Training time vs. test accuracy: All methods but CART and GBDT use guessed thresholds. GBDT and DL8.5 use data duplication. DL8.5 frequently times out, so there are fewer markers for it. While CART is the fastest algorithm, GOSDTwGuses its additional runtime to produce models with higher accuracy and generalize better.

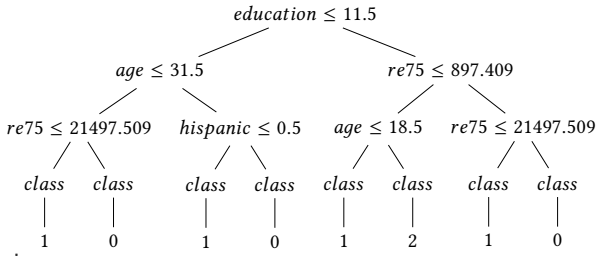

Figure 6: The tree generated by GOSDTwG (depth limit 3) on the Lalonde dataset.

on their conditional average treatment effect estimate. We selected features *age, education, black, hispanic, married, nodegree, re75* for the pre-treatment covariate vector. Then we labelled the data points as 2, 1, and 0 if the estimated treatment effect is larger than 2000, between −5000 and 2000, and less than −5000, respectively. Here, the penalty for each misclassification is defined as follows:

$$\text{cost} = \begin{cases} 0, & \text{correctly classified,} \\ 200 + 3 \times age & \text{label = 0 and misclassified,} \\ 100 + 3 \times age & \text{label = 1 and misclassified,} \\ 300 & \text{label = 2 and misclassified.} \end{cases}$$

We linearly scale the above costs to the range from 1 to 100, and in the case of data-duplication, we round them to integers and treat them as weights of the dataset. Figure 6 shows the generated tree by GOSDTwG with a depth limit of 3. Generated trees with other depth limits can be found in Appendix C.4.

5 CONCLUSIONS

To find the optimal weighted decision tree, we first suggest directly optimizing a weighted loss function. However, this approach might be time-consuming for large weighted datasets. To improve efficiency, we present the data-duplication approach, which rounds all weights to integers and then duplicates each sample by its weights. Finally, to further improve efficiency, we present a stochastic process, where we sample an unweighted dataset from our weighted dataset. Our results suggest a trade-off of accuracy and running time among these approaches.

ACKNOWLEDGMENTS

We acknowledge the following grant support: NIH/NIDA under grant number DA054994 and NSF under grant number IIS-2147061. This research was enabled in part by support provided by WestGrid (https://www.westgrid.ca) and The Digital Research Alliance (https://alliancecan.ca/en).

REFERENCES

[1] Sina Aghaei, Mohammad Javad Azizi, and Phebe Vayanos. 2019. Learning Optimal and Fair Decision Trees for Non-Discriminative Decision-Making. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (Jul. 2019), 1418–1426. https://doi.org/10.1609/aaai.v33i01.33011418

[2] Sina Aghaei, Andrés Gómez, and Phebe Vayanos. 2021. Strong Optimal Classification Trees. *arXiv preprint arXiv:2103.15965* (2021).

[3] Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. 2020. Learning optimal decision trees using caching branch-and-bound search. In *AAAI Conference on Artificial Intelligence*, Vol. 34. 3146–3153.

[4] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. 2018. Learning Certifiably Optimal Rule Lists for Categorical Data. *Journal of Machine Learning Research* 18, 234 (2018), 1–78. http://jmlr.org/papers/v18/17-716.html

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

[6] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and Regression Trees*. CRC press.

[7] Chaofan Chen and Cynthia Rudin. 2018. An optimization approach to learning falling rule lists. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[8] David A Cieslak and Nitesh V Chawla. 2008. Learning decision trees for unbalanced data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 241–256.

[9] Yifan Cui, Ruoqing Zhu, and Michael Kosorok. 2017. Tree based weighted learning for estimating individualized treatment rules with censored data. *Electronic Journal of Statistics* 11, 2 (2017), 3927–3953.

[10] Rajeev H Dehejia and Sadek Wahba. 1999. Causal effects in nonexperimental studies: Reevaluating the evaluation of training programs. *Journal of the American statistical Association* 94, 448 (1999), 1053–1062.

[11] Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J Stuckey. 2020. MurTree: Optimal Classification Trees via Dynamic Programming and Search. *arXiv preprint arXiv:2007.12652* (2020).

[12] Yashesh Dhebar, Kalyanmoy Deb, Subramanya Nageshrao, Ling Zhu, and Dimitar Filev. 2020. Interpretable-AI policies using evolutionary nonlinear decision trees for discrete action systems. *arXiv e-print arXiv:2009.09521* (2020).

[13] David Dobkin, Truxton Fulton, Dimitrios Gunopulos, Simon Kasif, and Steven Salzberg. 1997. Induction of shallow decision trees. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (1997).

[14] Kevin Doubleday, Hua Zhou, Haoda Fu, and Jin Zhou. 2018. An Algorithm for Generating Individualized Treatment Decision Trees and Random Forests. *Journal of Computational and Graphical Statistics* 27, 4 (2018), 849–860.

[15] Damien Ernst, Pierre Geurts, and Louis Wehenkel. 2005. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research* 6 (2005), 503–556.

[16] Alireza Farhangfar, Russell Greiner, and Martin Zinkevich. 2008. A fast way to produce near-optimal fixed-depth decision trees. In *Proceedings of the 10th International Symposium on Artificial Intelligence and Mathematics (ISAIM-2008)*.

[17] FICO, Google, Imperial College London, MIT, University of Oxford, UC Irvine, and UC Berkeley. 2018. Explainable Machine Learning Challenge. https://community.fico.com/s/explainable-machine-learning-challenge.

[18] Yoav Freund and Robert E. Schapire. 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Conference on Computational Learning Theory*. Springer, 23–37.

[19] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of Statistics* (2001), 1189–1232.

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

[21] Hao Hu, Mohamed Siala, Emmanuel Hébrard, and Marie-José Huguet. 2020. Learning optimal decision trees with MaxSAT and its integration in AdaBoost. In *29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI)*.

[22] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. 2019. Optimal sparse decision trees. In *Advances in Neural Information Processing Systems*. 7267–7275.

[23] Adam R Klivans, Rocco A Servedio, and Dana Ron. 2006. Toward Attribute Efficient Learning of Decision Lists and Parities. *Journal of Machine Learning Research* 7, 4 (2006).

[24] Eric B Laber and Ying-Qi Zhao. 2015. Tree-based methods for individualized treatment regimes. *Biometrika* 102, 3 (2015), 501–514.

[25] Himabindu Lakkaraju and Cynthia Rudin. 2017. Learning cost-effective and interpretable treatment regimes. In *Artificial intelligence and statistics*. PMLR, 166–175.

[26] Robert Lalonde. 1986. Evaluating the Econometric Evaluations of Training Programs with Experiment Data. *American Economic Review* 76 (02 1986), 604–20.

[27] J. Larson, S. Mattu, L. Kirchner, and J. Angwin. 2016. How We Analyzed the COMPAS Recidivism Algorithm. *ProPublica* (2016).

[28] Hyafil Laurent and Ronald L Rivest. 1976. Constructing optimal binary decision trees is NP-complete. *Inform. Process. Lett.* 5, 1 (1976), 15–17.

[29] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. 2020. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning (ICML)*. 6150–6160.

[30] Ariel Linden and Paul R Yarnold. 2018. Estimating causal effects for survival (time-to-event) outcomes by combining classification tree analysis and propensity score weighting. *Journal of Evaluation in Clinical Practice* 24, 2 (2018), 380–387.

[31] Hayden McTavish, Chudi Zhong, Reto Achermann, Ilias Karimalis, Jacques Chen, Cynthia Rudin, and Margo Seltzer. 2022. Fast Sparse Decision Tree Optimization via Reference Ensembles. In *AAAI Conference on Artificial Intelligence*, Vol. 36.

[32] Nina Narodytska, Alexey Ignatiev, Filipe Pereira, Joao Marques-Silva, and IS RAS. 2018. Learning Optimal Decision Trees with SAT. In *27th International Joint Conference on Artificial Intelligence (IJCAI)*. 1362–1368.

[33] Siegfried Nijssen and Elisa Fromont. 2007. Mining optimal decision trees from itemset lattices. In *13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 530–539.

[34] Harsh Parikh, Cynthia Rudin, and Alexander Volfovsky. 2018. MALTS: Matching after learning to stretch. *arXiv preprint arXiv:1811.07415* (2018).

[35] J. R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[36] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. 2022. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys* 16 (2022), 1–85.

[37] Cynthia Rudin and Seyda Ertekin. 2018. Learning Customized and Optimized Lists of Rules with Mathematical Programming. *Mathematical Programming C (Computation)* 10 (2018), 659–702.

[38] Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. 2020. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 1855–1865.

[39] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore. 2014. Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records. *BioMed Research International* 2014 (03 Apr 2014), 781670. https://doi.org/10.1155/2014/781670

[40] Yilun Sun and Lu Wang. 2021. Stochastic Tree Search for Estimating Optimal Dynamic Treatment Regimes. *J. Amer. Statist. Assoc.* 116, 533 (2021), 421–432.

[41] Nikolaj Tollenaar and PGM Van der Heijden. 2013. Which method predicts recidivism best?: a comparison of statistical, machine learning and data mining predictive models. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 176, 2 (2013), 565–584.

[42] Sicco Verwer and Yingqian Zhang. 2019. Learning optimal classification trees using a binary linear program formulation. In *AAAI Conference on Artificial Intelligence*, Vol. 33. 1625–1632.

[43] Matheus Guedes Vilas Boas, Haroldo Gambini Santos, Luiz Henrique de Campos Merschmann, and Greet Vanden Berghe. 2021. Optimal decision trees for the algorithm selection problem: integer programming based approaches. *International Transactions in Operational Research* 28, 5 (2021), 2759–2781.

[44] Caroline Wang, Bin Han, Bhrij Patel, Feroze Mohideen, and Cynthia Rudin. 2022. In Pursuit of Interpretable, Fair and Accurate Machine Learning for Criminal Recidivism Prediction. *Journal of Quantitative Criminology* (2022).

[45] Fulton Wang and Cynthia Rudin. 2015. Causal falling rule lists. *arXiv preprint arXiv:1510.05189* (2015).

[46] Tong Wang, Cynthia Rudin, Finale Doshi-Velez, Yimin Liu, Erica Klampfl, and Perry MacNeille. 2017. A Bayesian Framework for Learning Rule Sets for Interpretable Classification. *Journal of Machine Learning Research* 18, 70 (2017), 1–37. http://jmlr.org/papers/v18/16-003.html

[47] Yichi Zhang, Eric B Laber, Anastasios Tsiatis, and Marie Davidian. 2015. Using decision lists to construct interpretable and parsimonious treatment regimes. *Biometrics* 71, 4 (2015), 895–904.

A THEOREMS AND PROOFS

A.1 Proof of Theorem 1

Theorem 1. *(Performance Guarantee). Let $R(t_g, \tilde{\mathbf{x}}, \mathbf{y})$ denote the objective of t_g on the full binarized dataset $(\tilde{\mathbf{x}}, \mathbf{y})$ for some per-leaf penalty λ. Then for any decision tree t that satisfies the same depth constraint d, we have:*

$$R(t_g, \tilde{\mathbf{x}}, \mathbf{y}) \leq \frac{1}{\sum_{i=1}^{N} w_i} \left(\sum_{i \in s_{T,\text{incorrect}}} w_i + \sum_{i \in s_{T,\text{correct}}} \mathbb{1}[y_i \neq \hat{y}_i^t] \times w_i \right) + \lambda H_t.$$

That is, the objective of the guessing model is no worse than the union of errors made by the reference model and tree t.

Proof. We adapt the proof in the work of McTavish et al. [31] to the weighted setting. Similar to them, we use the following notation to discuss lower bounds:

- $\lambda \geq 0$ is a regularizing term
- For some decision tree $\mathcal{T}$, we calculate its risk as $R(\mathcal{T}, \tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^{\mathcal{T}}] + \lambda H_{\mathcal{T}}$, where $H_{\mathcal{T}}$ refers to the number of leaves in tree $\mathcal{T}$.
- T is a reference model we use to guess lower bounds.
- $s_{T,\text{correct}}$ and $s_{T,\text{incorrect}}$ are, respectively, the indices of the set of observations in the training set correctly classified by T and the set of observations incorrectly classified by T.
- s_a is the set of training set observations captured in our current subproblem.
- d represents the maximum allowed depth for solutions to our current subproblem. If $d = 0$, no further splits are allowed.
- $t_g(s_a, d, \lambda)$ is the lower-bound-guessing-algorithm's solution for subproblem s_a, depth limit d, and regularizer λ. When we just specify t_g without arguments, we are referring to the lower-bound-guessing-algorithm's solution for the root subproblem (the whole dataset), with the depth limit argument provided for the tree as a whole.
- Consider a subproblem s_a corresponding to the full set of points passing through a specific internal or leaf node of the optimal tree $\mathcal{T}^*$ (call it $\text{node}_{\mathcal{T}^*, s_a}$). Define $H_{\mathcal{T}_a^*}$ as the number of leaves below node_{s_a} (or 1 if node_{s_a} is a leaf). Note that this is also the number of leaves needed in an optimal solution for subproblem s_a. Similarly define $H_{t_{g,a}}$ as the number of leaves below node_{t_g, s_a} in t_g (when s_a corresponds to the full set of points passing through a node in t_g). Note that this does not necessarily correspond to the number of leaves needed in an optimal solution for subproblem s_a because t_g has not been fully optimized.
- $R_g(s_a, d, \lambda)$ is the objective of the solution found for subproblem s_a, depth limit d, and regularizer λ when we use lower bound guessing:

$$R_g(s_a, d, \lambda) := \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i \in s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^{t_g(s_a, d, \lambda)}] + \lambda H_{t_{g,a}}.$$

- $lb_g(s_a)$ was defined as

$$lb_g(s_a) := \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i \in s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^T] + \lambda,$$

  which could be obtained at equality if we achieve the accuracy of T in a single leaf. We add that this is equivalent (by definition) to

$$lb_g(s_a) = \frac{1}{\sum_{i=1}^{N} w_i} \left(\sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i \right) + \lambda.$$

- We additionally define $t^*(s_a, d, \lambda)$ as an optimal solution for subproblem s_a, depth limit d, and regularizer λ (that is, the solution found when we do not use lower bound guessing). When we just specify t^* without arguments, we are referring to a solution for the root subproblem (the whole dataset), with the depth limit argument provided for the tree as a whole.
- We also define $R(s_a, d, \lambda)$ as the objective of the optimal solution found for subproblem s_a, depth limit d, and regularizer λ (that is, the objective of the solution found when we do not use lower bound guessing):

$$R(s_a, d, \lambda) := \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i \in s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^{t^*(s_a, d, \lambda)}] + \lambda H_{t^* \cap s_a}.$$

  Here, by $t^* \cap s_a$ we refer to the part of the subproblem s_a captured by a part of tree t^*.
- Define $lb_{\max}(s_a, d, \lambda)$ as the highest lower bound estimate that occurs for a given subproblem s_a, depth budget d, and regularization λ, across the algorithm's whole execution when using lower bound guessing. Note that

$$R_g(s_a, d, \lambda) \leq lb_{\max}(s_a, d, \lambda)$$

  because when a subproblem is solved, the current lower bound is made to match the objective of the solution returned for that subproblem. As a reminder, R_g is computed after the subproblem is solved. When using lower bound guesses, it is possible for

intermediate lower bound estimates (and therefore $lb_{\max}$) to exceed $R_g(s_a, d, \lambda)$, and then the lower bound (but not $lb_{\max}$) is decreased to match the objective of the best solution found when the subproblem is solved.

Now, without loss of generality, select a tree t that is within some depth constraint d. We wish to prove that the risk on the full dataset (with some regularization λ) is bounded as:

$$R(t_g, \tilde{\mathbf{x}}, \mathbf{y}) \leq \frac{1}{\sum_{i=1}^{N} w_i} \left(\sum_{i \in s_{T,\text{incorrect}}} w_i + \sum_{i \in s_{T,\text{correct}}} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t] \right) + \lambda H_t. \tag{4}$$

or, equivalently, (defining s_{all} as the set of all points in the dataset):

$$R_g(s_{\text{all}}, d, \lambda) \leq \frac{1}{\sum_{i=1}^{N} w_i} \left(\sum_{i \in s_{T,\text{incorrect}}} w_i + \sum_{i \in s_{T,\text{correct}}} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t] \right) + \lambda H_t.$$

To show this, it is sufficient to show a result that is strictly more general. Specifically, we show that for any subproblem s_a (with $d, \lambda \geq 0$) that occurs as an internal or leaf node in t (including the root), we can bound $lb_{\max}(s_a, d, \lambda)$ as follows. Equation (4) is a direct consequence, using s_a as the full dataset s_{all}.

$$lb_{\max}(s_a, d, \lambda) \leq \frac{1}{\sum_{i=1}^{N} w_i} \left(\sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i + \sum_{i \in s_{T,\text{correct}} \cap s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t] \right) + \lambda H_{t_a}. \tag{5}$$

What we originally wished to prove then follows from noting that, by definition of $lb_{\max}$, we have $R_g(s_{\text{all}}, d, \lambda) \leq lb_{\max}(s_{\text{all}}, d, \lambda)$. Here, d, λ are the depth limit and regularization provided for t_g (where d matches or exceeds the depth of t).

We prove this sufficient claim (hereafter referred to as Equation 5) for all subproblems in t using induction.

Base Case: Let us take any subset of data s_a, depth constraint d, and regularization λ which corresponds to a subproblem in tree t whose solution in t is a leaf node.

Because the solution to s_a was a leaf in t, then its objective (without making further splits) is $ub = \frac{1}{\sum_{i=1}^{N} w_i} (\sum_{i \in s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t]) + \lambda \cdot 1$. We want to show, in this case, that (5) holds.

Our initial lower bound guess is lb_g. Either $lb_g > ub$, or $lb_g \leq ub$. If $lb_g > ub$, we are done with the subproblem as per the Branch-and-Bound algorithm in [31]. Otherwise, when $lb_g \leq ub$, we know from their algorithm that the lower bound can never increase above ub. Therefore, the highest value of the lower bound during execution, $lb_{\max}(s_a, d, \lambda)$, obeys $lb_{\max}(s_a, d, \lambda) \leq \max(ub, lb_g)$. Then,

$$\begin{aligned}
lb_{\max}(s_a, d, \lambda) &\leq \max(ub, lb_g) \quad \text{(by argument just above)} \\
&= \max\left(\frac{1}{\sum_{i=1}^{N} w_i} \sum_{i \in s_a} (w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t]) + \lambda, \frac{\sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i}{\sum_{i=1}^{N} w_i} + \lambda \right) \quad \text{(by definition of } ub \text{ and } lb_g) \\
&= \frac{1}{\sum_{i=1}^{N} w_i} \max\left(\sum_{i \in s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t], \sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i \right) + \lambda \\
&= \frac{1}{\sum_{i=1}^{N} w_i} \max\left(\sum_{i \in s_{T,\text{correct}} \cap s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t] + \sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t], \sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i \right) + \lambda.
\end{aligned}$$

Note that both terms inside the max are at most $\sum_{i \in s_{T,\text{correct}} \cap s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t] + \sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i$. Therefore,

$$lb_{\max}(s_a, d, \lambda) \leq \frac{1}{\sum_{i=1}^{N} w_i} \left(\sum_{i \in s_{T,\text{correct}} \cap s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t] + \sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i \right) + \lambda.$$

Moreover, the number of leaves in the optimal tree for subproblem s_a is 1, i.e., $H_{t_a} = 1$ (since s_a corresponds to a leaf in t), so

$$lb_{\max}(s_a, d, \lambda) \leq \frac{1}{\sum_{i=1}^{N} w_i} \left(\sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i + \sum_{i \in s_{T,\text{correct}} \cap s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t] \right) + \lambda H_{t_a}.$$

And this matches Equation (5), as required. Thus, we have shown that the base case obeys the statement of the theorem.

Inductive Step: Let the set of points s_a, depth constraint $d > 0$, and regularization $\lambda \geq 0$ be a subproblem that corresponds to an internal node in t. Let j indicate the feature that was split on in t for this node, and define s_j as the set of data points $\tilde{\mathbf{x}}_i$ such that $\tilde{x}_{ij} = 1$ and s_j^c as the set of data points $\tilde{\mathbf{x}}_i$ such that $\tilde{x}_{ij} = 0$. We assume (5) holds for both the left and right child subproblems and aim to show that it

holds for their parent subproblem. The left subproblem is $s_a \cap s_j$ with depth $d-1$ and the right subproblem is $s_a \cap s_j^c$ with depth $d-1$. Thus, assuming (as per the inductive hypothesis) that (5) holds, i.e.,

$$lb_{\max}(s_a \cap s_j, d-1, \lambda) \leq \frac{1}{\sum_{i=1}^N w_i}\left(\sum_{i \in s_{T,\text{incorrect}} \cap s_a \cap s_j} w_i + \sum_{i \in s_{T,\text{correct}} \cap s_a \cap s_j} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t]\right) + \lambda H_{t_a \cap s_j}$$

and

$$lb_{\max}(s_a \cap s_j^c, d-1, \lambda) \leq \frac{1}{\sum_{i=1}^N w_i}\left(\sum_{i \in s_{T,\text{incorrect}} \cap s_a \cap s_j^c} w_i + \sum_{i \in s_{T,\text{correct}} \cap s_a \cap s_j^c} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t]\right) + \lambda H_{t_{a \cap j^c}},$$

it remains to show that Equation (5) holds for s_a:

$$lb_{\max}(s_a, d, \lambda) \leq \frac{1}{\sum_{i=1}^N w_i}\left(\sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i + \sum_{i \in s_{T,\text{correct}} \cap s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t]\right) + \lambda H_{t_a}.$$

We prove the inductive step by cases:

(1) If $ub \leq lb_g + \lambda$, then as per the branch-and-bound algorithm in [31], s_a corresponds to a leaf in t_g, with a loss for this subproblem equal to ub. Since the algorithm returns immediately after changing the lower bound to ub, the maximum value the lower bound takes (that is, $lb_{\max}$) is whichever of lb_g or ub is higher. We have:

$$\begin{aligned}
lb_{\max}(s_a, d, \lambda) &\leq \max(lb_g, ub) \\
&\leq \max(lb_g, lb_g + \lambda) \quad \text{(Since we conditioned on } ub \leq lb_g + \lambda) \\
&\leq lb_g + \lambda \\
&= \frac{1}{\sum_{i=1}^N w_i} \sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i + 2\lambda \\
&\leq \frac{1}{\sum_{i=1}^N w_i}\left(\sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i + \sum_{i \in s_{T,\text{correct}} \cap s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t]\right) + 2\lambda.
\end{aligned}$$

Noting that because s_a corresponds to an internal node in t, there are at least two leaves below it, so $H_{t_a} \geq 2$. Thus,

$$lb_{\max}(s_a, d, \lambda) \leq \frac{1}{\sum_{i=1}^N w_i}\left(\sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i + \sum_{i \in s_{T,\text{correct}} \cap s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t]\right) + \lambda H_{t_a}.$$

This equation matches Equation 5, as required.

(2) Else, the lower bound (and therefore $lb_{\max}(s_a, d, \lambda)$) cannot exceed the combined lower bounds of the left and right subproblems from splitting on feature j. We know the split for feature j will lead to a lower bound estimate no more than $lb_{\max}(s_a \cap s_j, d-1, \lambda) + lb_{\max}(s_a \cap s_j^c, d-1, \lambda)$. Thus we have:

$$lb_{\max}(s_a, d, \lambda) \leq lb_{\max}(s_a \cap s_j, d-1, \lambda) + lb_{\max}(s_a \cap s_j^c, d-1, \lambda).$$

Using the inductive hypothesis, this reduces to

$$\begin{aligned}
lb_{\max}(s_a, d, \lambda) \leq\ & \frac{1}{\sum_{i=1}^N w_i}\left(\sum_{i \in s_{T,\text{incorrect}} \cap s_a \cap s_j} w_i + \sum_{i \in s_{T,\text{correct}} \cap s_a \cap s_j} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t]\right) + \lambda H_{t_{a \cap j}} \\
&+ \frac{1}{\sum_{i=1}^N w_i}\left(\sum_{i \in s_{T,\text{incorrect}} \cap s_a \cap s_j^c} w_i + \sum_{i \in s_{T,\text{correct}} \cap s_a \cap s_j^c} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t]\right) + \lambda H_{t_{a \cap j^c}}.
\end{aligned}$$

Noting that $s_a \cap s_j$ and $s_a \cap s_j^c$ partition s_a:

$$lb_{\max}(s_a, d, \lambda) \leq \frac{1}{\sum_{i=1}^N w_i}\left(\sum_{i \in s_{T,\text{incorrect}} \cap s_a} w_i + \sum_{i \in s_{T,\text{correct}} \cap s_a} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t]\right) + \lambda H_{t_a}.$$

This equation matches Equation (5), as required.

Thus we have proved the inductive step holds for all possible cases. By induction, then, we have proved Equation (5) holds for any internal or leaf node in t. Since the root node is an internal node of t, we have also proven Equation (5) holds for the root problem. As per the justification given when claiming Equation (5) was sufficient, that also means

$$R(t_g, \tilde{\mathbf{x}}, \mathbf{y}) \leq \frac{1}{\sum_{i=1}^{N} w_i} \left(\sum_{i \in s_{T,\text{incorrect}}} w_i + \sum_{i \in s_{T,\text{correct}}} w_i \times \mathbb{1}[y_i \neq \hat{y}_i^t] \right) + \lambda H_t$$

which is what we wished to show. □

A.2 Proof that Approximate Min Loss is not that far from True Min Loss

Recall the objective

$$R(\mathcal{T}) := \frac{1}{\sum_{i=1}^{N} w_i} \sum_i w_i I_i(\mathcal{T}) + \lambda \#\text{leaves}.$$

Define the objective with the approximate weights as

$$\tilde{R}(\mathcal{T}) := \frac{1}{\sum_{i=1}^{N} \tilde{w}_i} \sum_i \tilde{w}_i I_i(\mathcal{T}) + \lambda \#\text{leaves}.$$

When we rounded, we ensured that the weights did not change much. That is, we know by design that $\|\mathbf{w} - \tilde{\mathbf{w}}\|_\infty \leq \epsilon$.

Theorem 2. *Let t^* be a minimizer of the objective as $t^* \in \arg\min_t R(t)$, and $\tilde{\mathcal{T}}$ be a minimizer of the approximate loss function as $\tilde{\mathcal{T}} \in \arg\min_t \tilde{R}(t)$. If $\|\mathbf{w} - \tilde{\mathbf{w}}\|_\infty \leq \epsilon$, we have:*

$$|R(t^*) - \tilde{R}(\tilde{\mathcal{T}})| \leq max \left\{ \frac{(\zeta - 1)\psi + \epsilon}{\zeta}, \frac{(\eta - 1)\psi + \epsilon}{\eta} \right\},$$

where $\eta = \max_{1 \leq i \leq N} \left\{ \frac{w_i}{\tilde{w}_i} \right\}$, $\zeta = \max_{1 \leq i \leq N} \left\{ \frac{\tilde{w}_i}{w_i} \right\}$, and $\psi = \frac{\max_i \{w_i, \tilde{w}_i\}}{\min_i \{w_i, \tilde{w}_i\}}$.

Proof. Since we know that $\forall 1 \leq i \leq N : \frac{1}{\zeta} \leq \frac{w_i}{\tilde{w}_i} \leq \eta$, and also $\forall 1 \leq i \leq N : |w_i - \tilde{w}_i| \leq \epsilon$, then for any tree $\mathcal{T}$,

$$\begin{aligned}
|R(\mathcal{T}) - \tilde{R}(\mathcal{T})| &= \left| \frac{1}{\sum_{i=1}^{N} w_i} \sum_i w_i I_i(\mathcal{T}) + \lambda \#\text{leaves} - \frac{1}{\sum_{i=1}^{N} \tilde{w}_i} \sum_i \tilde{w}_i I_i(\mathcal{T}) - \lambda \#\text{leaves} \right| \\
&= \left| \sum_i \left(\frac{w_i}{\sum_{i=1}^{N} w_i} - \frac{\tilde{w}_i}{\sum_{i=1}^{N} \tilde{w}_i} \right) I_i(\mathcal{T}) \right| \\
&\leq \sum_i \left| \frac{w_i}{\sum_{i=1}^{N} w_i} - \frac{\tilde{w}_i}{\sum_{i=1}^{N} \tilde{w}_i} \right| I_i(\mathcal{T}) \leq \sum_i \max \left\{ \frac{w_i}{\sum_i w_i} - \frac{\tilde{w}_i}{\zeta \sum_i w_i}, \frac{\tilde{w}_i}{\sum_i \tilde{w}_i} - \frac{w_i}{\eta \sum_i \tilde{w}_i} \right\} I_i(\mathcal{T}) \\
&\leq \sum_i \max \left\{ \frac{(\zeta - 1)w_i + \epsilon}{\zeta \sum_{i=1}^{N} w_i}, \frac{(\eta - 1)\tilde{w}_i + \epsilon}{\eta \sum_{i=1}^{N} \tilde{w}_i} \right\} I_i(\mathcal{T}) \\
&\leq \sum_i \max \left\{ \frac{(\zeta - 1) \max_i\{w_i\} + \epsilon}{\zeta \times N \times \min_i\{w_i\}}, \frac{(\eta - 1) \max_i\{\tilde{w}_i\} + \epsilon}{\eta \times N \times \min_i\{\tilde{w}_i\}} \right\} I_i(\mathcal{T}) \quad (\text{since } N \times \max_i\{w_i\} \geq \sum_i^N w_i \geq N \times \min_i\{w_i\}) \\
&= \sum_i \frac{\max \left\{ \frac{(\zeta - 1)\psi + \epsilon}{\zeta}, \frac{(\eta - 1)\psi + \epsilon}{\eta} \right\}}{N} I_i(\mathcal{T}) \quad (\text{where we used that } \psi = \frac{\max_i \{w_i, \tilde{w}_i\}}{\min_i \{w_i, \tilde{w}_i\}}) \\
&= \max \left\{ \frac{(\zeta - 1)\psi + \epsilon}{\zeta}, \frac{(\eta - 1)\psi + \epsilon}{\eta} \right\} \mathcal{L}(\mathcal{T}). \qquad (6)
\end{aligned}$$

For simplicity, we let $K = \max \left\{ \frac{(\zeta - 1)\psi + \epsilon}{\zeta}, \frac{(\eta - 1)\psi + \epsilon}{\eta} \right\}$. We will use proof-by-contradiction. Let us assume that the statement of the theorem is false. This implies:

$$K < |R(t^*) - \tilde{R}(\tilde{\mathcal{T}})| = |\tilde{R}(\tilde{\mathcal{T}}) - R(t^*)|. \qquad (7)$$

Since the error rate $\mathcal{L}(\mathcal{T})$ is always less than 1 for any tree, we have that when $\tilde{R}(\tilde{\mathcal{T}}) \geq R(t^*)$,

$$\begin{aligned}
K \times \mathcal{L}(t^*) &< K < |\tilde{R}(\tilde{\mathcal{T}}) - R(t^*)| = \tilde{R}(\tilde{\mathcal{T}}) - R(t^*) \\
R(t^*) + K\mathcal{L}(t^*) &< \tilde{R}(\tilde{\mathcal{T}}) \qquad (8)
\end{aligned}$$

and conversely, when $R(t^*) \geq \tilde{R}(\tilde{\mathcal{T}})$,

$$\begin{aligned} K\mathcal{L}(\tilde{\mathcal{T}}) &< K < |R(t^*) - \tilde{R}(\tilde{\mathcal{T}})| = R(t^*) - \tilde{R}(\tilde{\mathcal{T}}) \\ \tilde{R}(\tilde{\mathcal{T}}) + K\mathcal{L}(\tilde{\mathcal{T}}) &< R(t^*). \end{aligned} \tag{9}$$

Using (6) for t^*, we have:

$$\begin{aligned} \tilde{R}(t^*) - R(t^*) &\leq |R(t^*) - \tilde{R}(t^*)| \leq K \times \mathcal{L}(t^*) \\ \tilde{R}(t^*) &\leq R(t^*) + K\mathcal{L}(t^*). \end{aligned}$$

Since $\tilde{\mathcal{T}}$ is the minimizer of $\tilde{R}$, we know $\tilde{R}(\tilde{\mathcal{T}}) \leq \tilde{R}(t^*)$. Combining that with the equation above, we have:

$$\tilde{R}(\tilde{\mathcal{T}}) \leq R(t^*) + K\mathcal{L}(t^*). \tag{10}$$

Analogously, using (6) for $\tilde{\mathcal{T}}$, we have:

$$\begin{aligned} R(\tilde{\mathcal{T}}) - \tilde{R}(\tilde{\mathcal{T}}) &\leq |R(\tilde{\mathcal{T}}) - \tilde{R}(\tilde{\mathcal{T}})| \leq K\mathcal{L}(\tilde{\mathcal{T}}) \\ R(\tilde{\mathcal{T}}) &\leq \tilde{R}(\tilde{\mathcal{T}}) + K\mathcal{L}(\tilde{\mathcal{T}}). \end{aligned}$$

Since t^* is the minimizer of R, we know $R(t^*) \leq R(\tilde{\mathcal{T}})$. Combining that with the equation above, we have:

$$R(t^*) \leq \tilde{R}(\tilde{\mathcal{T}}) + K\mathcal{L}(\tilde{\mathcal{T}}). \tag{11}$$

Combining (8) with (10), we have

$$R(t^*) + K\mathcal{L}(t^*) < \tilde{R}(\tilde{\mathcal{T}}) \leq R(t^*) + K\mathcal{L}(t^*), \tag{12}$$

which is a contradiction since one cannot have a number strictly less than itself. Similarly, combining (9) and (11), we have

$$\tilde{R}(\tilde{\mathcal{T}}) + K\mathcal{L}(\tilde{\mathcal{T}}) < R(t^*) \leq \tilde{R}(\tilde{\mathcal{T}}) + K\mathcal{L}(\tilde{\mathcal{T}}), \tag{13}$$

which again leads to a contradiction.

These contradictions, which occur whether $\tilde{R}(\tilde{\mathcal{T}}) \geq R(t^*)$ or $R(t^*) \geq \tilde{R}(\tilde{\mathcal{T}})$, imply that our assumption, namely (7), is incorrect, thus, we have its opposite, $|R(t^*) - \tilde{R}(\tilde{\mathcal{T}})| \leq K$. □

A.3 Proof of Theorem 3

Theorem 3. *Given a weighted dataset $D = \{(\mathbf{x}_i, y_i, w_i)\}_{i=1}^N$, an arbitrary positive real number $r > 0$, an arbitrary positive real number $\varepsilon > 0$, and a tree $\mathcal{T}$, if we sample $S = r \times N$ data points from D, $\tilde{D} = \{(\tilde{\mathbf{x}}_i, \tilde{y}_i)\}_{i=1}^S$, we have:*

$$\mathbb{P}\left(|\tilde{\mathcal{L}}(\mathcal{T}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}}) - \mathcal{L}_{\mathbf{w}}(\mathcal{T}, \mathbf{x}, \mathbf{y})| \geq \varepsilon\right) \leq 2\exp\left(-\frac{2\varepsilon^2}{S^2}\right).$$

Proof. For a given tree $\mathcal{T}$, let X_i be a random variable such that $X_i = \mathbb{1}[y_i \neq \hat{y}_i^T]$. Accordingly, for the sampled dataset, we have:

$$\tilde{\mathcal{L}} = \sum_{i=1}^{S} X_i.$$

Since for each $1 \leq i \leq S$ we have $0 \leq X_i \leq 1$, based on Hoeffding's inequality, we have:

$$\mathbb{P}\left(|\tilde{\mathcal{L}}(\mathcal{T}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}}) - \mathbb{E}\left[\tilde{\mathcal{L}}(\mathcal{T}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}})\right]| \geq \varepsilon\right) \leq 2\exp\left(-\frac{2\varepsilon^2}{S}\right).$$

As we discussed, $\mathbb{E}\left[\tilde{\mathcal{L}}(\mathcal{T}, \tilde{\mathbf{x}}, \tilde{\mathbf{y}})\right] = \mathcal{L}_{\mathbf{w}}(\mathcal{T}, \mathbf{x}, \mathbf{y})$. So we can easily conclude the theorem. □

B EXPERIMENTAL DETAILS

B.1 Datasets

Lalonde: The Lalonde dataset is from the National Supported Work Demonstration [10, 26], a labour market experiment in which participants were randomized between treatment (on-the-job training lasting between nine months and a year) and control groups. Accordingly, for each unit U_i, we have a pre-treatment covariate vector X_i and observed assigned treatment Z_i. Let Y_i^1 be the potential outcome if unit U_i received the treatment, and Y_i^2 be the potential outcome if it was not treated. When a unit is treated, we do not observe the potential outcome if it was not treated and vice versa. To address this issue, we use the MALTS model [34] that estimates these missing values by matching. MALTS gives us an estimation of the conditional average treatment effect, which we call TE. Since our weighted decision tree is designed for classification, we classify participants into three groups – "should be treated," "should be treated if budget allows," and "should not be treated" – based on their estimated conditional average treatment effect (TE). We selected features *age, education, black, hispanic, married, nodegree, re75* for the pre-treatment covariate vector. Here, re75 is earnings in 1975. Then we labelled the data points as 2, 1, and 0 if the estimated treatment effect is larger than 2000, between −5000 and 2000, and less than −5000, respectively. Here, the penalty for each misclassification is defined as follows:

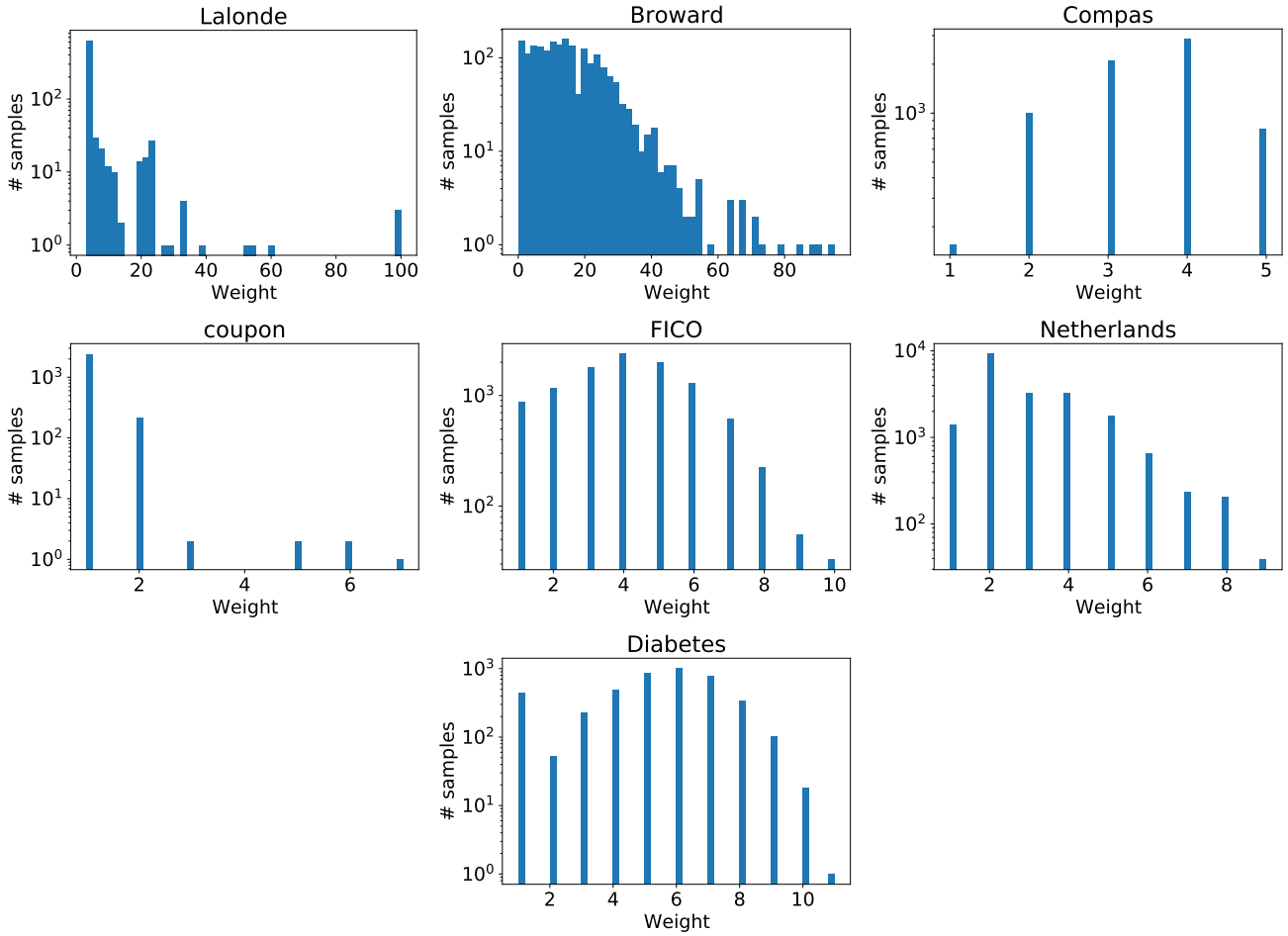

Figure 7: Distribution of weights.

$$\text{cost} = \begin{cases} 0, & \text{if we correctly classify them,} \\ 200 + 3 \times age & \text{if their label is 0 and we misclassify,} \\ 100 + 3 \times age & \text{if their label is 1 and we misclassify,} \\ 300 & \text{if their label is 2 and we misclassify.} \end{cases}$$

We linearly scale the above costs to the range from 1 to 100, and in the case of data-duplication, we round them to integers and treat them as weights of the dataset. Figure 7 shows the distribution of weights.

Broward: In this dataset [44], we predict whether defendants have any type of charge (for which they were eventually convicted) within two years from the current charge/release date. We selected features *sex, age_at_current_charge, age_at_first_charge, p_charges, p_incarceration, p_probation, p_juv_fel_count, p_felprop_viol, p_murder, p_felassault, p_misdeassault, p_famviol, p_sex_offense, p_weapon, p_fta_two_year, p_fta_two_year_plus, current_violence, current_violence20, p_pending_charge, p_felony, p_misdemeanor, p_violence, total_convictions, p_arrest, p_property, p_traffic, p_drug, p_dui, p_domestic, p_stalking, p_voyeurism, p_fraud, p_stealing, p_trespass, six_month, one_year, three_year, and five_year* and the label *general_two_year*. We use the inverse propensity scores as the weights to balance the number of samples for each value of feature "sex." Figure 7 shows the distribution of weights.

COMPAS: In this dataset [27], we predict whether individuals are arrested within two years of release. We selected features *sex, age, juv_fel_count, juv_misd_count, juv_other_count, priors_count, and c_charge_degree* and the label *two_year_recid*. We use inverse propensity scores as the weights to balance the number of samples for each value of feature "sex."

Coupon: In this dataset [46], we predict whether a customer will accept a coupon for takeaway food or a cheap restaurant depending on their coupon usage history, current conditions while driving, and coupon expiration time. We selected features *destination, passanger, weather, temperature, time, expiration, gender, age, maritalStatus, Childrennumber, education, occupation, income, Bar, CoffeeHouse, CarryAway,*

Table 2: Configurations used to train decision tree models

Dataset	Depth Limit	Regularizer (λ)
Lalonde	2, 3, 4, 5, 6	0.1, 0.05, 0.02, 0.01
Broward	2, 3, 4, 5, 6	0.005, 0.002, 0.001, 0.0005
Coupon	2, 3, 4, 5, 6	0.005, 0.002, 0.001, 0.0005
Diabetes	2, 3, 4, 5, 6	0.005, 0.002, 0.001, 0.0005
COMPAS	2, 3, 4, 5, 6	0.005, 0.002, 0.001, 0.0005, 0.0002
FICO	2, 3, 4, 5, 6	0.01, 0.005, 0.002, 0.001, 0.0005
Netherlands	2, 3, 4, 5, 6	0.01, 0.005, 0.002, 0.001

RestaurantLessThan20, Restaurant20To50, toCoupon_GEQ15min, toCoupon_GEQ25min, direction_same and the label *Y*, and removed observations with missing values. We then used one-hot encoding to transform these categorical features into binary features. We use inverse propensity scores as the weights to balance the number of samples for each value of feature "destination."

FICO: In the FICO dataset [17], we predict whether an individual will default on a loan. We use this dataset without preprocessing, and use propensity scores as the weights to balance the number of samples for each value of feature "ExternalRiskEstimate."

Netherlands: The prediction task in this dataset is to see whether defendants have any type of charge within four years [41]. We translated the feature names from Dutch to English and then selected features *sex, country of birth, log # of previous penal cases, age in years, age at first penal case, offence type, 11-20 previous case, >20 previous case, and age squared* and the label recidivism_in_4y. We use inverse propensity scores as the weights to balance the number of samples for each value of feature "sex".

Diabetes: The Diabetes dataset [39] includes the data of 10 years of clinical care at 130 US hospitals, that has over 50 features representing patient and hospital outcomes. In this dataset we predict whether an individual will readmit to the hospital or not. We selected features *race, gender, age, medical-specialty, diag1, diag2, diag3, max-glu-serum, A1Cresult, metformin, repaglinide, nateglinide, chlorpropamide, glimepiride, acetohexamide, glipizide, glyburide, tolbutamide, pioglitazone, rosiglitazone, acarbose, miglitol, troglitazone, tolazamide, examide, citoglipton, insulin, glyburide-metformin, glipizide-metformin, glimepiride-pioglitazone, metformin-rosiglitazone, metformin-pioglitazone, change, diabetesMed*. We use inverse propensity scores as the weights to balance the number of samples for each value of feature "gender".

B.2 Configurations

Table 2 lists the configurations used for each dataset when training decision trees. For GOSDTwG, DL8.5, and CART, we set the depth limit from 2 to 5. Also, the last column shows the different values of the regularizer, λ, for different datasets. Also, we have used different numbers of samples in our weighted samplinng approach. We vary the percentage of samplig from 100% to 600%.

B.3 Evaluation Platform

All reported times are from a 32-core dual Intel E5-2683 v4 Broadwell processor running at 2.1 Ghz, with approximately 125 GB of available memory. We ran all tests single-threaded (i.e., we used only one of the 32 cores) on the Cedar cluster of Compute Canada. For the experiment reported in Figure 1, we also ran on a 2.6GHz 6-core Intel Core i7 processor and 16GB of 2400MHz DDR4 onboard memory.

B.4 Software Packages Used

GOSDT and DL8.5 with Guessing: For guessing technique proposed by McTavish et al. [31], we use their publicly released code (https://github.com/ubc-systopia/gosdt-guesses) and (https://github.com/ubc-systopia/pydl8.5-lbguess).

DL8.5 For DL8.5 of Aglin et al. [3], we used the implementation in their main repository (https://github.com/aia-uclouvain/pydl8.5).

CART: We run CART using the Python implementation from Sci-Kit Learn.

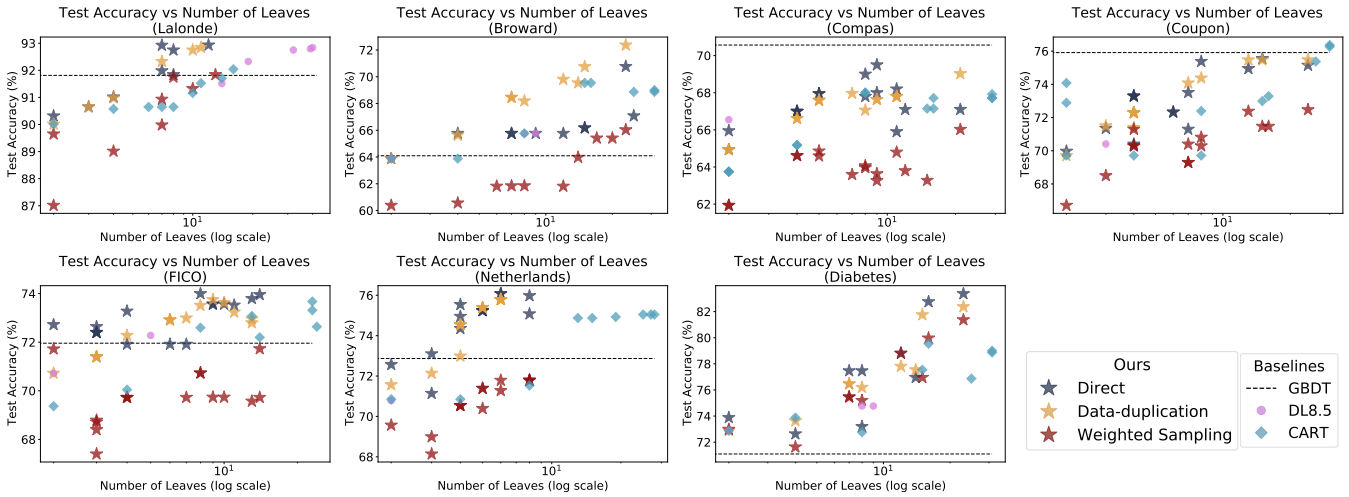

Figure 8: Sparsity vs. test accuracy: All methods but CART and GBDT use guessed thresholds. GBDT and DL8.5 use data duplication. DL8.5 frequently times out, so there are fewer markers for it. GOSDTwG achieves the highest training accuracy for every level of sparsity. GOSDTwG also achieves the highest test accuracy for *almost* every level of sparsity.

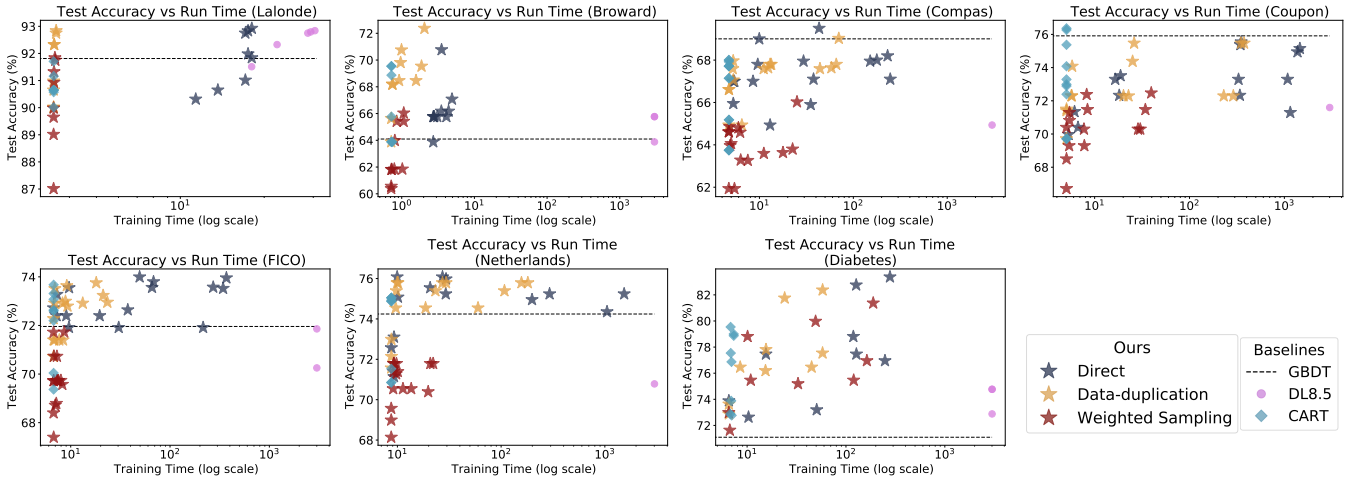

Figure 9: Training time vs. test accuracy: All methods but CART and GBDT use guessed thresholds. GBDT and DL8.5 use data duplication. DL8.5 frequently times out, so there are fewer markers for it. While CART is the fastest algorithm, GOSDTwGuses its additional runtime to produce models with higher accuracy and generalize better.

C MORE EXPERIMENTAL RESULTS

C.1 Sparsity vs. Test Accuracy

In Section 4, we discussed the results for three datasets. Figure 8 shows the results of sparsity vs. test accuracy for all datasets. GOSDTwG produced excellent test accuracy with a small number of leaves and compared to other decision tree models, achieves higher accuracy for every level of sparsity.

C.2 Training Time vs. Test Accuracy

In Section 4, we discussed the results of training time and test accuracy of our methods and baselines on three datasets. Figure 9 reports the results of training time vs. test accuracy on all datasets. While the training times of GOSDTwG and CART are almost the same, GOSDTwG achieves the highest test accuracy in almost all cases. DL8.5 struggled with the 1-hour termination condition for all datasets except Lalonde; because DL8.5 did not solve to optimality, it was outperformed by both CART and GOSDTwG.

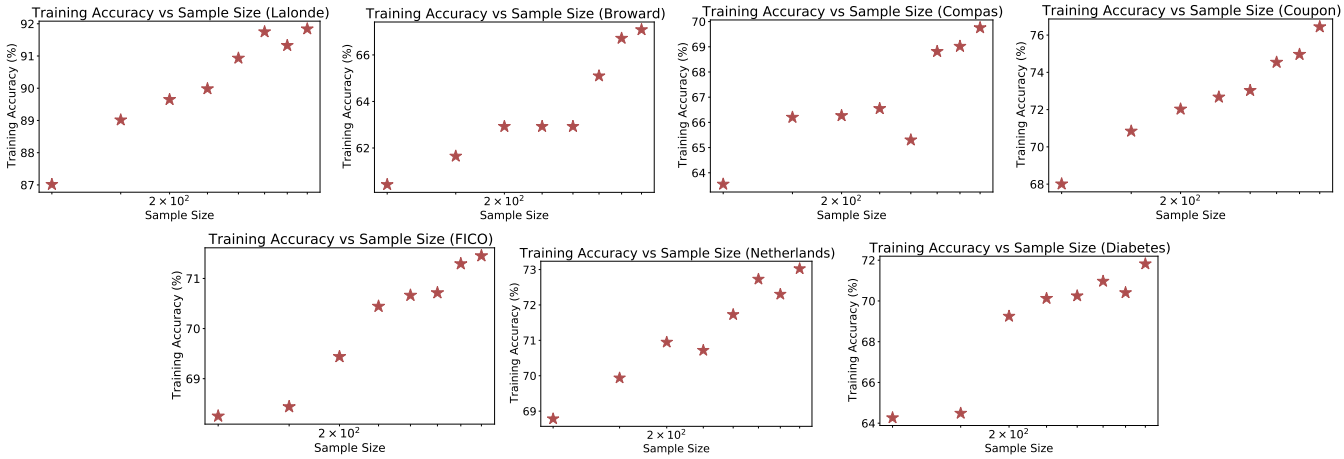

Figure 10: Training accuracy vs. sample size: the accuracy of weighted sampling approach increase with the sample size.

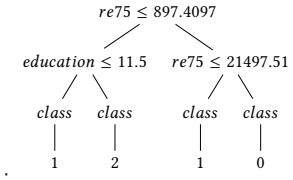

Figure 11: The tree generated by weighted GOSDTwG (depth limit 2) on the Lalonde dataset.

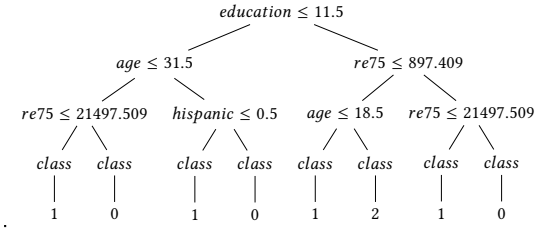

Figure 12: The tree generated by weighted GOSDTwG (depth limit 3) on the Lalonde dataset.

C.3 Effect of the Number Samples

Intuitively, the accuracy of weighted sampling approach increase with the sample size. To show this effect, Figure 10 shows the effect of sample size on accuracy. Results indicates that increasing the sample size lets us to have a more accurate model.

C.4 Lalonde Case Study

As discussed in Section 4 (Lalonde Case Study), to show the effectiveness of our approach for real-world problems, we conducted a case study on the Lalonde dataset. Trees produced by GOSDTwG with depth limits of 2, 3, 4, and 5 are found in Figures 11-14.

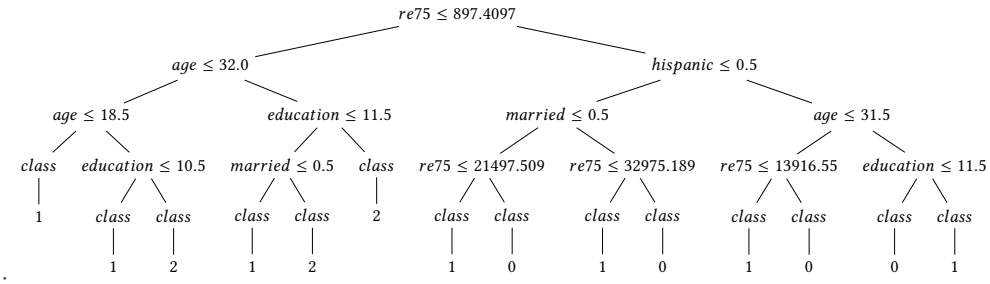

Figure 13: The tree generated by weighted GOSDTwG (depth limit 4) on the Lalonde dataset.

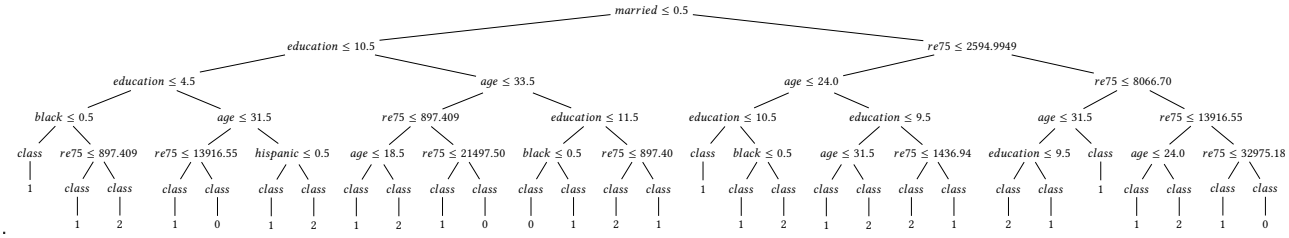

Figure 14: The tree generated by weighted GOSDTwG (depth limit 5) on the Lalonde dataset.