Fast optimization of weighted sparse decision trees for use in optimal treatment regimes and optimal policy design

Ali Behrouz[1], Mathias Lécuyer[1], Cynthia Rudin[2] and Margo Seltzer[1]

[1]University of British Columbia, Vancouver, British Columbia, Canada
[2]Duke University, Durham, North Carolina, USA

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 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 but is computationally inefficient. Our second approach scales better by transforming weights to integer values and using 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.

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., 1, 2, 3]. Historically, decision tree optimization has involved greedy tree induction, where trees are built from the top down [4, 5, 6], but more recently there have been several approaches that fully optimize sparse trees to yield the best combination of performance and interpretability [7, 8, 9, 10]. Optimization of sparse optimal trees is NP-hard, and recent work has leveraged the fact that the loss takes on a discrete number of values to provide a computational advantage [11, 12, 13, 14]. 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 [15, 16], are not directly possible with these algorithms.

For example, 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 weight the loss from each patient and minimize the sum of the weighted losses. While it is possible to construct a model using CART's suboptimal greedy splitting procedure [5], the current fastest optimal decision tree method, GOSDT [14], does not support this approach.

We extend the framework of GOSDT-with-Guesses [13] 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.

A key contributor to GOSDTwG's performance is its use of bitvectors to compute the loss function. However, introducing weights requires multiplying the weights by this bitvector representation, which introduces a runtime penalty of one to two orders of magnitude. We demonstrate this effect in our first approach. Our second approach introduces a normalization and data duplication technique to mitigate the slowdown due to real-valued weights. Here, we transform the weights to small integer values and then duplicate each sample by its transformed weight. Our third approach, which scales to much larger sample sizes, uses 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 the direct approach, (2) we can bound the accuracy loss that data duplication introduces; and (3) the weighted optimal

Advances in Interpretable Machine Learning and Artificial Intelligence 'AIMLAI, October 21, 2022, Atlanta, GA
alibez@cs.ubc.ca (A. Behrouz); mathias.lecuyer@ubc.ca (M. Lécuyer); cynthia@cs.duke.edu (C. Rudin); mseltzer@cs.ubc.ca (M. Seltzer)

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 [17]. While full decision tree optimization is NP-hard [18], it is possible to make assumptions, e.g., feature independence, that simplify the hard optimization to cases where greedy methods suffice [19]. However, these assumptions are unrealistic in practice. Other approaches [20, 21] 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) [9, 10, 12, 22, 23, 24], but MIP solvers tend to be slow. Several new algorithms use customized dynamic programming algorithms with branch-and-bound techniques to improve decision tree optimization scalability. In particular, analytical bounds combined with bitvector-based computation efficiently reduce the search space and improve runtime [25, 26, 27]. Lin et al. [14] extend this approach to use dynamic programming, which leads to even better scalability. Demirović et al. [28] introduce constraints on both depth and the number of nodes to improve scalability. Recently, McTavish et al. [13] 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 runtime 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 near-optimal decision trees quickly.

Several studies focus on learning tree- and list-based treatment regimes from data [29, 30, 31, 32, 33, 34, 35]. However, none of these methods fully optimize the policy, because the techniques used for optimization were not known when the work was done.

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, $y_i \in \{0, 1, \ldots, K\}$ 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 to create dummy variables [25] or by using a subset of these splits as done by McTavish et al. [13]. 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 $\mathcal{T}$ on the is:

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

To achieve interpretability and prevent overfitting, we provide the option to use either soft sparsity regularization on the number of leaves, hard regularization on the tree depth, or both [see 13]:

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

where $H_{\mathcal{T}}$ is the number of leaves in $\mathcal{T}$ and λ is a per-leaf 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 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. [13] 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). 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*, transforms the weights; specifically, we normalize, scale, and round the weights to 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. Finally, to scale to even 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. [13] 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 depth-constrained 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 a potentially complex reference model (e.g., a boosted decision tree model) on training observation i. The reference model is used as an upper bound on the performance of the sparse decision tree we are optimizing. 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. [13], we define our guessed lower bound on the achievable loss on subset s_a as:

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

Eq. 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 a loss less than or equal to 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 define the subset's contribution to the objective as:

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

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 above-mentioned lower bound guess, we can now replace the lower bound of McTavish et al. [13] 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 the optimal tree. 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.

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:*

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

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

Hence, the model t_{g} 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 appears in our supplementary material [36].

Motivation for Data Duplication. Surprisingly, increasing the dataset size by replicating data is substantially faster than using the direct approach. Decision tree optimization requires repeatedly evaluating the objective. Small improvements in that computation lead to a large improvement (possibly orders of magnitude) in execution time. In the direct approach, computing the objective (2) requires computing the inner product $\mathbf{w} \cdot \mathcal{I}$, where $\mathcal{I}_i = \mathbb{1}[y_i \neq \hat{y}_i^T]$. In the unweighted case, as all weights are 1, this computation can be performed using bitvectors, which is 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 bitvectors as in the unweighted case, preserving fast computation.

Data-duplication Algorithm. The data-duplication algorithm is shown in Algorithm 1. We first normalize all weights and scale them to $(0, 1]$. Given an integer, $p > 0$, we then multiply each normalized weight by p and round to integers. 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.* Our experimentsshow that if we choose the value of p appropriately, this method improves training runtime

Algorithm 1: Data Duplication

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

$\tilde{X} \leftarrow \emptyset$; $\tilde{y} \leftarrow \emptyset$;
Define $\tilde{w}_i = \text{round}(p \cdot (\frac{w_i}{\sum_{i=1}^{N} w_i}))$;
for $x_i \in \mathbf{x}$ **do**
  for $i = 1, 2, \ldots, \tilde{w}_i$ **do**
    $\tilde{X} \leftarrow \tilde{X} \cup \{x_i\}$; $\tilde{y} \leftarrow \tilde{y} \cup \{y_i\}$;
return $\tilde{X}, \tilde{y}$

significantly without losing too much accuracy. After data-duplication, there are no weights associated with samples, and we can use the fast bit-vector computations from the unweighted case.

Correctness of Data Duplication. One might ask if the data duplication approach produces suboptimal solutions, because its loss function is an approximation to the weighted loss. If 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

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

Define the objective with the approximate weights as

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

By design, the rounding phase rounds amplified weights, ensuring that the absolute change in weights remains small. That is, we know 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 or scaling weights by p does 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 $\|\mathbf{w} - \tilde{\mathbf{w}}\|_\infty \leq \epsilon$,*

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

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 $\psi = \frac{\max_i\{w_i, \tilde{w}_i\}}{\min_i\{w_i, \tilde{w}_i\}}$.

In other words, the rounded solution provably will not lose substantial performance, as long as both the additive and multiplicative changes in weights due to rounding are small. The value of η and ζ are usually small and near 1, if the original weights do not have extreme imbalances. If the value of ψ is large, then the direct approach is more efficient, so we should duplicate data. When we use data duplication, the value of ψ should also be small. The proof is in our supplementary material [36].

Table 1

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

Weighted Sampling. When the ratio between the biggest and smallest weights is large, data duplication might be inefficient if it requires creating many samples. To address this issue, we present a stochastic sampling process based on weights. Given an arbitrary amplification 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 [17].

4.1. Datasets

We use seven publicly available real-world datasets; Table 1 shows sizes of these datasets: The Lalonde dataset [37, 38], Broward [39], the coupon dataset, which was collected on Amazon Mechanical Turk via a survey [40], Diabetes [41], which is a health care related dataset, the

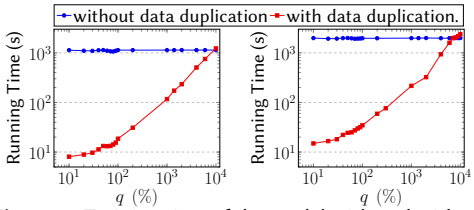


Figure 1: Training time of the model with and without data duplication on different machines.

Fair Isaac (FICO) credit risk dataset [42] from the Explainable ML Challenge, and the COMPAS [43] and Netherlands [44] datasets, which are recidivism datasets. Unless stated otherwise, we use inverse propensity score with respect to one of the features as our weights.

We ran the experiments with different depth bounds and regularization; each point in each plot shows the results for one setting. A full description of the data sets and configurations appear in our supplement [36].

4.2. Baselines

We compared our methods with the following baseline models: (1) CART [5], (2) DL8.5 [45], and (3) Gradient Boosted Decision Trees (GBDT) [46, 47]. 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 data-duplication. We use the unweighted FICO dataset and randomly pick $q\%$ of the original $\mathcal{S}$ data points. We assign each selected point $weight = 2$ by duplicating it, 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 runtimes for this data-duplicated data set and the original dataset in which we assign $weight = 2$ to the selected samples and $weight = 1$ to the remaining samples. We run this experiment on two machines, with different processors and amounts of memory, to show the consistency of the results on different machines. The full machine descriptions appear in our supplementary material [36]. Figure 1 shows 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 all three approaches. 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, 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. The dotted line and round and diamond shapes in Figures 2 and 3(a) illustrate 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 can be found in [36].

Training time vs. test accuracy. Figures 3(b) and 3(c) show the training time and accuracy for different methods. While the training times of GOSDTwG and CART are almost the same, GOSDTwG achieves the highest training and test accuracy in almost all cases. As DL8.5 timed out at one hour on all datasets except Lalonde, it did not reach optimality and was outperformed by both CART and GOSDTwG. Results of other datasets can be found in our supplement [36].

Lalonde Case Study. The Lalonde dataset is from the National Supported Work Demonstration [38, 37], a labour market experiment in which participants were randomized between treatment (9-12 month on-the-job training) and control groups. Each unit U_i has a pre-treatment covariate vector X_i and observed assigned treatment Z_i. Let Y_i^1 be the outcome if unit U_i received the treatment and Y_i^2 be the outcome if it was not treated. When a unit is treated, we do not observe the outcome had it not been treated and vice versa. We use the MALTS model [48] to estimate these missing values by matching, producing an estimate of the conditional average treatment effect. We classify participants into three groups—"should be treated," "should be treated if budget allows," and "should not be treated" -- based on their conditional average treatment effect estimate. 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. We define the penalty for each misclassification as **(i)** cost = 0 if correctly classified, **(ii)** cost = $200 + 3 \times age$ if label = 0 and misclassified, **(iii)** cost = $100 + 3 \times age$ if label = 1 and misclassified, and **(iv)** cost = 300 if label = 2 and misclassified. 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 4 shows the tree produced by GOSDTwG with a depth limit of 3; trees with other depth limits appear in [36].

5. Conclusions

To find the optimal weighted decision tree, we first suggest directly optimizing a weighted loss function. To

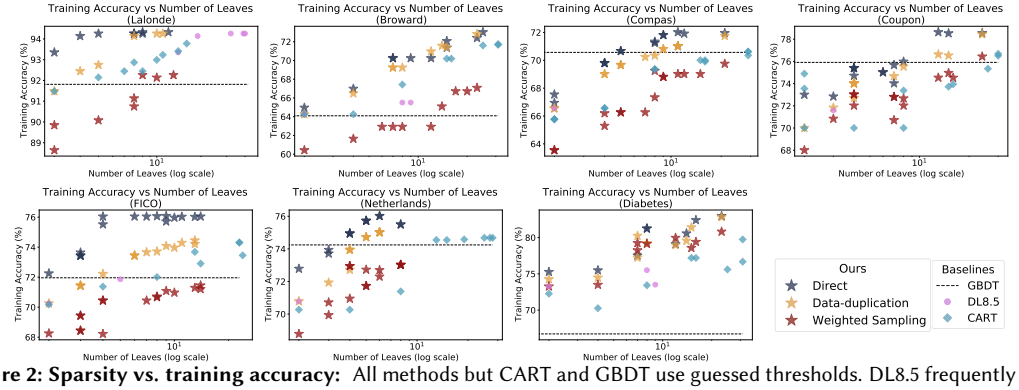
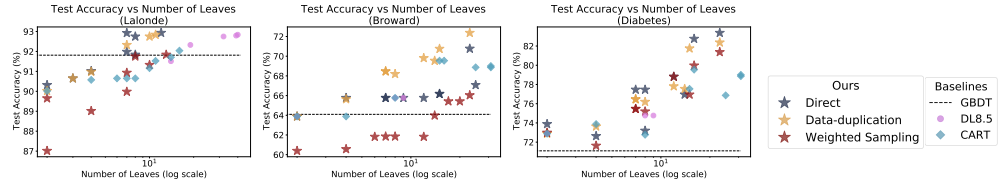
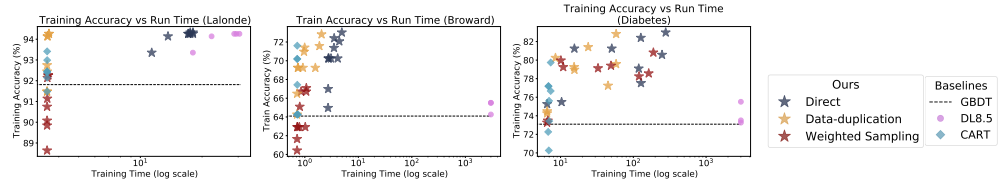


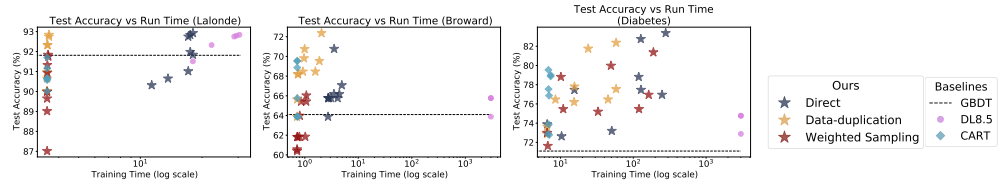
Figure 2: Sparsity vs. training accuracy: All methods but CART and GBDT use guessed thresholds. DL8.5 frequently times out, so there are fewer markers for it. GOSDTwG achieves the highest accuracy for every level of sparsity.



(a) Sparsity vs. test accuracy:



(b) Training time vs. training accuracy:



(c) Training time vs. test accuracy:

Figure 3: GOSDTwG achieves the highest test accuracy for ***almost*** every level of sparsity. While CART is the fastest algorithm, GOSDTwG uses its additional runtime to produce models with higher accuracy and that generalize better.

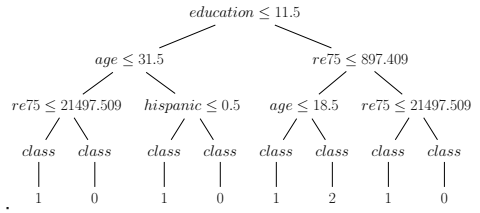


Figure 4: The tree generated by GOSDTwG (depth limit 3) on the Lalonde dataset.

improve efficiency, we present the data-duplication approach, which rounds all weights to integers and then duplicates each sample by its weight. To further improve efficiency, we present a stochastic process in which we sample an unweighted dataset from our weighted dataset. Our results suggest a trade-off of accuracy and runtime 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). We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] D. Ernst, P. Geurts, L. Wehenkel, Tree-based batch mode reinforcement learning, Journal of Machine Learning Research 6 (2005) 503–556.

[2] A. Silva, M. Gombolay, T. Killian, I. Jimenez, S.-H. Son, Optimization methods for interpretable differentiable decision trees applied to reinforcement learning, in: International Conference on Artificial Intelligence and Statistics (AISTATS), 2020, pp. 1855–1865.

[3] Y. Dhebar, K. Deb, S. Nageshrao, L. Zhu, D. Filev, Interpretable-AI policies using evolutionary nonlinear decision trees for discrete action systems, arXiv e-print arXiv:2009.09521 (2020).

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

[5] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen, Classification and Regression Trees, CRC press, 1984.

[6] D. Dobkin, T. Fulton, D. Gunopulos, S. Kasif, S. Salzberg, Induction of shallow decision trees, IEEE Trans. on Pattern Analysis and Machine Intelligence (1997).

[7] A. Farhangfar, R. Greiner, M. Zinkevich, 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), 2008.

[8] S. Nijssen, E. Fromont, Mining optimal decision trees from itemset lattices, in: 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007, pp. 530–539.

[9] D. Bertsimas, J. Dunn, Optimal classification trees, Machine Learning 106 (2017) 1039–1082.

[10] O. Günlük, J. Kalagnanam, M. Li, M. Menickelly, K. Scheinberg, Optimal decision trees for categorical data via integer programming, Journal of Global Optimization (2021) 1–28.

[11] S. Aghaei, M. J. Azizi, P. Vayanos, Learning optimal and fair decision trees for non-discriminative decision-making, Proceedings of the AAAI Conference on Artificial Intelligence 33 (2019) 1418–1426. URL: https://ojs.aaai.org/index.php/AAAI/article/view/3943. doi:10.1609/aaai.v33i01.33011418.

[12] S. Aghaei, A. Gómez, P. Vayanos, Strong optimal classification trees, arXiv preprint arXiv:2103.15965 (2021).

[13] H. McTavish, C. Zhong, R. Achermann, I. Karimalis, J. Chen, C. Rudin, M. Seltzer, Fast sparse decision tree optimization via reference ensembles, in: AAAI Conference on Artificial Intelligence, volume 36, 2022.

[14] J. Lin, C. Zhong, D. Hu, C. Rudin, M. Seltzer, Generalized and scalable optimal sparse decision trees, in: International Conference on Machine Learning (ICML), 2020, pp. 6150–6160.

[15] A. Linden, P. R. Yarnold, 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 (2018) 380–387.

[16] D. A. Cieslak, N. V. Chawla, Learning decision trees for unbalanced data, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2008, pp. 241–256.

[17] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, C. Zhong, Interpretable machine learning: Fundamental principles and 10 grand challenges, Statistics Surveys 16 (2022) 1–85.

[18] H. Laurent, R. L. Rivest, Constructing optimal binary decision trees is np-complete, Information Processing Letters 5 (1976) 15–17.

[19] A. R. Klivans, R. A. Servedio, D. Ron, Toward attribute efficient learning of decision lists and parities., Journal of Machine Learning Research 7 (2006).

[20] N. Narodytska, A. Ignatiev, F. Pereira, J. Marques-Silva, I. RAS, Learning optimal decision trees with sat, in: 27th International Joint Conference on Artificial Intelligence (IJCAI), 2018, pp. 1362–1368.

[21] H. Hu, M. Siala, E. Hébrard, M.-J. Huguet, 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), 2020.

[22] S. Verwer, Y. Zhang, Learning optimal classification trees using a binary linear program formulation, in: AAAI Conference on Artificial Intelligence, volume 33, 2019, pp. 1625–1632.

[23] C. Rudin, S. Ertekin, Learning customized and optimized lists of rules with mathematical programming, Mathematical Programming C (Computation) 10 (2018) 659–702.

[24] M. G. Vilas Boas, H. G. Santos, L. H. d. C. Merschmann, G. Vanden Berghe, Optimal decision trees for the algorithm selection problem: integer programming based approaches, International Transactions in Operational Research 28 (2021) 2759–2781.

[25] X. Hu, C. Rudin, M. Seltzer, Optimal sparse decision trees, in: Advances in Neural Information Processing Systems, 2019, pp. 7267–7275.

[26] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, C. Rudin, Learning certifiably optimal rule lists for categorical data, Journal of Machine Learning Research 18 (2018) 1–78. URL: http://jmlr.org/papers/v18/17-716.html.

[27] C. Chen, C. Rudin, An optimization approach to

learning falling rule lists, in: International Conference on Artificial Intelligence and Statistics (AISTATS), 2018.

[28] E. Demirović, A. Lukina, E. Hebrard, J. Chan, J. Bailey, C. Leckie, K. Ramamohanarao, P. J. Stuckey, Murtree: Optimal classification trees via dynamic programming and search, arXiv preprint arXiv:2007.12652 (2020).

[29] H. Lakkaraju, C. Rudin, Learning cost-effective and interpretable treatment regimes, in: Artificial intelligence and statistics, PMLR, 2017, pp. 166–175.

[30] Y. Zhang, E. B. Laber, A. Tsiatis, M. Davidian, Using decision lists to construct interpretable and parsimonious treatment regimes, Biometrics 71 (2015) 895–904.

[31] F. Wang, C. Rudin, Causal falling rule lists, arXiv preprint arXiv:1510.05189 (2015).

[32] E. B. Laber, Y.-Q. Zhao, Tree-based methods for individualized treatment regimes, Biometrika 102 (2015) 501–514.

[33] Y. Cui, R. Zhu, M. Kosorok, Tree based weighted learning for estimating individualized treatment rules with censored data, Electronic Journal of Statistics 11 (2017) 3927–3953.

[34] K. Doubleday, H. Zhou, H. Fu, J. Zhou, An algorithm for generating individualized treatment decision trees and random forests, Journal of Computational and Graphical Statistics 27 (2018) 849–860.

[35] Y. Sun, L. Wang, Stochastic tree search for estimating optimal dynamic treatment regimes, Journal of the American Statistical Association 116 (2021) 421–432.

[36] A. Behrouz, M. Lecuyer, C. Rudin, M. Seltzer, Fast optimization of weighted sparse decision trees for use in optimal treatment regimes and optimal policy design, 2022. URL: https://arxiv.org/abs/2210.06825. doi:10.48550/ARXIV.2210.06825.

[37] R. Lalonde, Evaluating the econometric evaluations of training programs with experiment data, American Economic Review 76 (1986) 604–20.

[38] R. H. Dehejia, S. Wahba, Causal effects in nonexperimental studies: Reevaluating the evaluation of training programs, Journal of the American statistical Association 94 (1999) 1053–1062.

[39] C. Wang, B. Han, B. Patel, F. Mohideen, C. Rudin, In pursuit of interpretable, fair and accurate machine learning for criminal recidivism prediction, Journal of Quantitative Criminology (2022).

[40] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, P. MacNeille, A bayesian framework for learning rule sets for interpretable classification, Journal of Machine Learning Research 18 (2017) 1–37. URL: http://jmlr.org/papers/v18/16-003.html.

[41] B. Strack, J. P. DeShazo, C. Gennings, J. L. Olmo, S. Ventura, K. J. Cios, J. N. Clore, Impact of hba1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records, BioMed Research International 2014 (2014) 781670. URL: https://doi.org/10.1155/2014/781670. doi:10.1155/2014/781670.

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

[43] J. Larson, S. Mattu, L. Kirchner, J. Angwin, How we analyzed the COMPAS recidivism algorithm, ProPublica (2016).

[44] N. Tollenaar, P. Van der Heijden, 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 (2013) 565–584.

[45] G. Aglin, S. Nijssen, P. Schaus, Learning optimal decision trees using caching branch-and-bound search, in: AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 3146–3153.

[46] Y. Freund, R. E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in: Conference on Computational Learning Theory, Springer, 1995, pp. 23–37.

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

[48] H. Parikh, C. Rudin, A. Volfovsky, MALTS: Matching after learning to stretch, arXiv preprint arXiv:1811.07415 (2018).