Weighted Optimal Decision Trees

Ryan Shar **Mathias Lecuyer**

Abstract

Due to their popularity, many algorithms exist for learning decision trees. Finding the optimal decision tree is an NP-hard problem, so heuristic search methods were designed for faster fitting. These methods can find near-optimal decision trees but often rely on discrete loss functions. The reliance on discrete loss makes these algorithms ill-suited for weighted datasets with real-valued weights having non-discrete loss. Preprocessing the weighted data is required to fit a tree with a heuristic search. One method of transforming a weighted dataset to an unweighted approximation involves duplication, sampling points uniformly with probability proportional to their weight. This form of fully stochastic duplication has high variance and is prone to excluding low-weight points. We propose a novel duplication method to remedy this: Deterministic and Bernoulli sampling. Our duplication method has lower variance and has better approximation for low-weight points. To experimentally compare sampling methods, we design three methods of synthetic data generation: Linearly Separable with Errors, Single d-Spherical, and Multi-Spherical. These datasets have controllable parameters that directly affect weighted loss. Our experiments demonstrate how the weighted error of each duplication method is affected by these factors and that Deterministic and Bernoulli sampling outperforms fully stochastic sampling.

1 Introduction

Decision trees are a popular class of machine learning model due to their interpretability and effectiveness. While the resulting model may be simple, learning an optimal decision tree is NP-hard in general [Hyafil and Rivest, 1976]. Rather than finding an optimal decision tree, efficient heuristic searches exist for learning sub-optimal trees that outperform greedy learning methods. These heuristics often rely on discrete loss in trees, making them unsuitable for real-valued loss [Behrouz et al., 2022]. Tasks that have real-valued weights in their datasets cannot directly apply these efficient search methods. Far from niche, many applications have a reliance on weighted datasets, from correcting class imbalance [Cieslak and Chawla, 2008] to malware detection [Firdausi et al., 2010], and especially in healthcare [Deepak and Ameer, 2023]. This problem exists for more than just heuristic decision tree algorithms. Support for weighted data is missing in many modern, popular machine learning methods, like the ‘SuperLearner’ package in R [MacNell et al., 2023].

Preprocessing the weighted data to an unweighted approximation is required to use these methods directly. The resulting unweighted dataset must maintain properties similar to the weighted data. A simple and effective method of transformation involves the duplication of data. Duplication aims to have points with more weight appear more frequently than points with smaller weights. We can achieve this by using the relative proportion of weights: if one point has twice the weight of another, it should appear twice as frequently. Previous work accomplishes this with a fully stochastic approach, uniformly sampling points according to their weight [Behrouz et al., 2022]. While this method works, it often requires extensive storage to achieve consistency. With a constrained amount of storage, this method suffers from high variance and is prone to under-representing points with low weight. We propose a novel duplication method to produce matching or better approximations of weighted data. Our duplication method involves Deterministic and Bernoulli (DaB) components to sample points.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

DaB duplication approximates weighted datasets with less variance and better represents low-weight points.

2 Duplication Under Constraint

Let $D = \{(x_i, y_i, w_i)\}_{i=1}^N$ denote the training dataset, where each x_i is a data vector of d features, $y_i \in \{1, ..., k\}$ is the classification label for the point i, and w_i is the weight given to the point i. Let $\ell : \mathbb{R}^d \to \{1, 0\}$ denote the classification loss of some decision tree on a data vector (1 for incorrect prediction and 0 otherwise). The exact weighted training loss is then defined as:

$$\mathcal{L}_w = \frac{1}{\sum_{i=1}^N w_i} \sum_{i=1}^N \ell(x_i) w_i. \tag{1}$$

Our goal is to use duplication in order to recreate the weights of each point by replicating each x_i some $m_i \propto w_i$ number of times. While it is possible to precisely recreate the weights of a dataset with an arbitrary amount of duplication, practical storage and time constraints prevent exact replication. In our setting, we are given a number $S := Np$ representing the total number of allowed points after duplication for some multiple $p \in \mathbb{R}$. We aim to find a strategy that approximates a dataset D with weights w constrained by S.

Existing work by Behrouz et al. [2022] duplicates by sampling S points uniformly, with each point having probability $w_i / \sum_{i=j}^N w_j$. This approximation is exact in the limit, but suffers from high variance. Let $\tilde{x}_i$ denote the randomly sampled points under this strategy. We use Hoeffding's inequality to bound the error:

$$\mathbb{P}\left(\left| \sum_{i=1}^{Np} \ell(\tilde{x_i}) - \mathbb{E}\left[\sum_{i=1}^{Np} \ell(x_i) \right] \right| \geq \epsilon \right) \leq 2\text{exp}\left(\frac{-2\epsilon^2}{Np} \right).$$

By re-parameterizing with $\delta := 2\text{exp}(\frac{-2\epsilon^2}{Np})$, we have

$$\mathbb{P}\left(\left| \sum_{i=i}^{Np} \ell(\tilde{x_i}) - \mathbb{E}\left[\sum_{i=1}^{Np} \ell(x) \right] \right| \leq \sqrt{\frac{Np}{2} \ln(\frac{2}{\delta})} \right) \geq 1 - \delta$$

$$\mathbb{P}\left(\left| \frac{1}{Np} \sum_{i=i}^{Np} \ell(\tilde{x_i}) - \frac{1}{Np} \mathbb{E}\left[\sum_{i=1}^{Np} \ell(x) \right] \right| \leq \frac{1}{Np} \sqrt{\frac{Np}{2} \ln(\frac{2}{\delta})} \right) \geq 1 - \delta$$

$$\mathbb{P}\left(\left| \frac{1}{Np} \sum_{i=i}^{Np} \ell(\tilde{x_i}) - \mathcal{L}_w \right| \leq \frac{1}{\sqrt{p}} \sqrt{\frac{1}{2N} \ln(\frac{2}{\delta})} \right) \geq 1 - \delta.$$

Since there are 2^N ways of classifying the points, we union bound over the possible classification trees to find that for any tree,

$$\mathbb{P}\left(\left| \frac{1}{Np} \sum_{i=i}^{Np} \ell(\tilde{x_i}) - \mathcal{L}_w \right| \leq \frac{1}{\sqrt{p}} \sqrt{\frac{1}{2N} \ln(\frac{2 \cdot 2^N}{\delta})} \right) \geq 1 - \delta.$$

$$\mathbb{P}\left(\left| \frac{1}{Np} \sum_{i=i}^{Np} \ell(\tilde{x_i}) - \mathcal{L}_w \right| \leq \frac{1}{\sqrt{p}} \sqrt{\frac{1}{2N} \left[N \ln(2) + \ln(\frac{2}{\delta}) \right]} \right) \geq 1 - \delta.$$

$$\mathbb{P}\left(\left| \frac{1}{Np} \sum_{i=i}^{Np} \ell(\tilde{x_i}) - \mathcal{L}_w \right| \leq \frac{1}{\sqrt{p}} \sqrt{\frac{\ln(2)}{2} + \frac{1}{2N} \ln(\frac{2}{\delta})} \right) \geq 1 - \delta.$$

Since $\sqrt{\frac{\ln(2)}{2}} \leq 0.6$, a reasonably large N and $p \approx 10^4$ gives us about 1% difference in error.

To improve on this bound, we propose a duplication method with both deterministic and stochastic components: Deterministic and Bernoulli (DaB) duplication. DaB duplication is able to achieve the same error with a smaller p by using a deterministic component t_i and a stochastic component sampled from a Bernoulli parameterized by θ_i. We define the quantities:

$$\hat{w}_i := \frac{w_i}{\sum_{k=1}^N w_k}, \qquad t_i := \lfloor \hat{w}_i N p \rfloor, \quad \text{and} \quad \theta_i := \hat{w}_i N p - t_i.$$

We copy each point $x_i \in D$ deterministically t_i times in our duplicated dataset $\tilde{D}$, then add one additional copy $\tilde{x_i}$ according to a Bernoulli random variable: $\tilde{x_i} \sim \text{Bern}(\theta_i)$ (Algorithm 1 line 9). This allows each point to appear in $\tilde{D}$ at approximately the same frequency as the weight w_i.

Algorithm 1: Determinisitc and Bernoulli Duplication

```
Input  : Dataset $D, y$, weights $w$, number of points in dataset $N$, max number of samples after
         duplication $S$
Output : Duplicated dataset $\tilde{D}, \tilde{y}$ which approximates $D, y$ having weights $w$
1  $\tilde{D} \leftarrow \emptyset; \tilde{y} \leftarrow \emptyset$;
2  $p \leftarrow \frac{S}{N}$;
3  Define $\hat{w}_i \leftarrow = \frac{w_i}{\sum_{i=1}^N w_i}, t_i \leftarrow \lfloor \hat{w}_i \cdot N \cdot p \rfloor$;
4  for $x_i, y_i \in D$ do
       // Deterministic duplication
5      for $1, 2, ..., t_i$ do
6          $\tilde{D} \leftarrow \tilde{D} \cup \{x_i\}$;
7          $\tilde{y} \leftarrow \tilde{y} \cup \{y_i\}$;
8      end
       // Sample $x_i$ following Bernoulli($\theta$)
9      $\theta \leftarrow \hat{w}_i N p - t_i$;
10     $a \leftarrow$ Unif[0, 1];
11     if $a < \theta$ then
12         $\tilde{D} \leftarrow \tilde{D} \cup \{x_i\}$;
13         $\tilde{y} \leftarrow \tilde{y} \cup \{y_i\}$;
14     end
15 end
16 return $\tilde{D}, \tilde{y}$
```

Our approximation to the weighted error, $\mathcal{L}_{\hat{w}}$, is defined by a deterministic component L and a stochastic component $\Delta\ell$ as:

$$L := \frac{1}{Np} \sum_{i=1}^N t_i \ell(x_i), \qquad \Delta\ell := \frac{1}{Np} \sum_{1=1}^N \mathbb{1}\{\tilde{x}_i = 1\} \ell(x_i), \qquad \mathcal{L}_{\hat{w}} = L + \Delta\ell.$$

The deterministic loss exactly accounts for the weighted loss for $S - N \cdot \lfloor p \rfloor$ points so we only need to analyze the loss of the stochastic N points. To bound the error of $\Delta\ell$, we use Hoeffding's inequality again. With probability at least $1 - \delta$,

$$\left| \sum_{i=1}^N \mathbb{1}\{\tilde{x}_i = 1\} \ell(x_i) - \mathbb{E}\left[\sum_{i=1}^N \ell(x) \right] \right| \leq \sqrt{-\frac{N}{2} \ln(\frac{2}{\delta})}$$

$$\left| \frac{1}{Np} \sum_{i=1}^N \mathbb{1}\{\tilde{x}_i = 1\} \ell(x_i) - \frac{1}{Np} \mathbb{E}\left[\sum_{i=1}^N \ell(x) \right] \right| \leq \frac{1}{Np} \sqrt{-\frac{N}{2} \ln(\frac{2}{\delta})}$$

$$|\Delta\ell - \mathbb{E}\left[\Delta\ell\right]| \leq \sqrt{\frac{1}{2Np^2} \ln(\frac{2}{\delta})}.$$

Since there are 2^N ways of classifying the points, we union bound over the possible classification trees to find that for any tree,

$$\begin{aligned}
|\Delta\ell - \mathbb{E}(\Delta\ell)| &\leq \sqrt{\frac{1}{2Np^2}\ln(\frac{2\cdot 2^N}{\delta})} \\
&= \sqrt{\frac{1}{2Np^2}[N\ln(2) + \ln(\frac{2}{\delta})]} \\
&= \frac{1}{p}\sqrt{\frac{\ln(2)}{2} + \frac{1}{2N}\ln(\frac{2}{\delta})},
\end{aligned}$$

with probability at least $1-\delta$. Since $\sqrt{\frac{\ln(2)}{2}} \leq 0.6$, a reasonably large N and $p \approx 100$ gives us about 1% difference, improving on the previous $p \approx 10^4$. With this bound, we see that DaB duplication has less variance than fully random sampling for the same S.

3 Experiments

We compare our duplication method with existing approaches by training decisions trees with each method and comparing their weighted loss (Equation 1). To isolate the effects of p on each duplication scheme, we developed synthetic datasets with controlled dimension and shape. This allows us to target specific properties of each duplication method and examine their shortcomings.

3.1 Synthetic Data

We design three synthetic datasets: Linearly Separable with Errors (LSE), Single d-Spherical, and Multi-Spherical (MS). The LSE and spherical methods are able to have arbitrary number of class labels k and arbitrary dimension d.

3.1.1 Linearly Separable with Errors Dataset

To generate points with the LSE method, we define p_{wrong} as the proportion of samples that are labelled incorrectly. This generation scheme allows us to control the number of classification errors by modifying p_{wrong}. A dataset with $p_{\text{wrong}} = 0$ is linearly separable and can be perfectly classified by a tree with $O(dk)$ rules. Setting $p_{\text{wrong}} = 1$ requires deep trees for classification as the points become completely mixed. Since errors increase as p_{wrong} increases, we could effectively control the difficulty of classification by modifying a single hyper-parameter.

To create a dataset with this method, we chose to sample using a uniform distribution. We generated k disjoint domains for each feature j by sampling a lower bound a_j ~Unif[-10,0] (Algorithm 2, line 2) and width u_j ~round(Unif[3,5]) (Algorithm 2, line 4). The disjoint domains are constructed based on the width and start value, with the kth domain as (a_j + $(k-1)u_j$, a_j + $k \cdot u_j$). The total domain of the feature is then $[a, a + u \cdot k]$. To generate some point, select a label c and uniformly sample each feature from the c th domain (Algorithm 2, line 7). If the point is designated as "incorrect," then randomly assign a label from $\{1, ..., k\}/\{c\}$, otherwise use c (Algorithm 2, line 13). We plot an instance of LSE data in the results section, Figure 1.

3.1.2 d-Spherical Dataset

Decision trees with linear partitions require multiple rules to approximate circular regions. Similar to p_{wrong} in LSE data, the weighted error of a tree will increase with the number of different circular regions. We take advantage of this fact when designing our spherical dataset. To still make classification feasible, we ensure that each circular region is disjoint. Our synthetic method creates these disjoint regions by generating points using a three sphere (3S). As defined in algorithm 3, 3S points are generated to have an equal number of spherical points around some centre with radius 1, 2, and 3. This creates three separate balls of points around the origin which can be assigned arbitrary labels.

The weighted error of 3S data can be controlled by label assignment to each region. For example, if the innermost region (radius 1) is assigned the same label as the middle region (radius 2), then our

Algorithm 2: Linearly Seperable with Errors

```
Input  : Dimension $d$, number of classes $k$, number of points $n$, and the proportion of errors
         $p_{\text{wrong}}$
Output : Dataset $D$ and labels $y$
1  $D \leftarrow \emptyset; y \leftarrow \emptyset$;
   // Sample starting value for each feature and width to get the range for
      each class
2  start[$i$] $\leftarrow$ Unif[-10,0];
3  width[$i$] $\leftarrow$ Unif[3,5];
4  range[$j, k$]$\leftarrow$ (start[$j$] + $(k-1)$width[$j$], start[$j$] + $k\cdot$ width[$j$]) ;
5  $n_{\text{wrong}} \leftarrow \lfloor p_{\text{wrong}} \cdot n \rfloor$;
6  $n_{\text{correct}} \leftarrow n - n_{\text{wrong}}$;
7  for $c \leftarrow 1$ to $k$ do
       // Generate points with correct label
8      for $i \leftarrow 1$ to $\lfloor \frac{n_{correct}}{k} \rfloor$ do
9          $x \leftarrow d$-dimensional vector where $x_j$ is uniformly sampled from range[$j, c$];
10         $y \leftarrow y \cup c$ ; // Add correct label
11         $D \leftarrow D \cup x$
12     end
       // Generate points with wrong label
13     for $i \leftarrow 1$ to $\lfloor \frac{n_{wrong}}{k} \rfloor$ do
14         $x \leftarrow d$-dimensional vector where $x_j$ is sampled from range[$j, c$];
15         $y \leftarrow y\cup$ Unif[$\{1, ..., k\}/\{c\}$];
16         $D \leftarrow D \cup x$
17     end
18 end
19 return $D, y$
```

algorithm only needs to define the space fore the outermost region (radius 3). When all regions are labelled differently, the decision tree must partition the space for all three spheres. As the number of unique labels increases, the weighted loss will increase (for the same depth).

Algorithm 3: 3-Spherical

```
Input  : Dimension $d$, a vector $k$ where $k[i]$ is the label for sphere $i$, number of points $n$, and the
         centre of the circle $\vec{c}$
Output : Dataset $D$ and labels $y$
1  $D \leftarrow \emptyset; y \leftarrow \emptyset$;
   // One third of the points per class
2  $n_{\text{circle}} \leftarrow \lfloor \frac{n}{3} \rfloor$
3  for $i \leftarrow 1$ to $n_{circle}$ do
4      $X_1 \leftarrow$ spherical point around the $\vec{c}$ with dimension $d$ and radius 1;
5      $X_2 \leftarrow$ spherical point around the $\vec{c}$ with dimension $d$ and radius 2;
6      $X_3 \leftarrow$ spherical point around the $\vec{c}$ with dimension $d$ and radius 3;
7      $D \leftarrow D \cup \{X_1, X_2, X_3\}$;
8      $y \leftarrow y \cup \{k[0], k[1], k[2]\}$
9  end
10 return $D, y$
```

In our tests, we generate two types of spherical data: the Single d-spherical dataset and the Multi-Spherical (MS) dataset. The Single d-spherical dataset is a special case of 3-spherical data around the origin, where the innermost and outermost (points with radius 1 and 3) circles have label 1 and the middle circle (points with radius 2) has label 0. MS data uses the 3S generation method with the same labelling scheme as Single d-spherical, but with varying centres. In general, MS data can have an arbitrary number of centres as long as they have ℓ_2 distance at least 6 to ensure disjoint regions. In our experiments, we fixed the number of spheres and their centres (Algorithm 4, line 3). Similarly to p_{wrong} in the LSE generation method, the classification error of a tree increases as the number of three spheres increases. Some instances of 3S and MS data are plotted in the results section, Figure 1 and Figure 2.

Algorithm 4: Four d-Spheres

```
Input  : Dimension $d$ and the number of points $n$
Output : Dataset $D$ and labels $y$
1  $D \leftarrow \emptyset; y \leftarrow \emptyset$;
   // One fourth of the points per class
2  $n_{\text{circle}} \leftarrow \lfloor \frac{n}{4} \rfloor$
3  $c_1 \leftarrow \langle -10, ..., -10 \rangle$;
4  $c_2 \leftarrow \langle 10, ..., 10 \rangle$;
5  $c_3 \leftarrow$ vector where first $\lfloor \frac{d}{2} \rfloor$ entries are 10 and the rest are $-10$ ;
6  $c_4 \leftarrow$ vector where first $\lfloor \frac{d}{2} \rfloor$ entries are -10 and the rest are $10$ ;
7  $k \leftarrow [1, 0, 1]$
8  $D_1, y_1 \leftarrow$ Generate $n_{\text{circle}}$ 3S points with center $c_1$ and labels $k$ ;
9  $D_2, y_2 \leftarrow$ Generate $n_{\text{circle}}$ 3S points with center $c_2$ and labels $k$;
10 $D_3, y_3 \leftarrow$ Generate $n_{\text{circle}}$ 3S points with center $c_3$ and labels $k$;
11 $D_4, y_4 \leftarrow$ Generate $n_{\text{circle}}$ 3S points with center $c_4$ and labels $k$;
12 return $\{D_1 \cup D_2 \cup D_3 \cup D_4\}, \{y_1 \cup y_2 \cup y_3 \cup y_4\}$
```

3.2 Weight Generation Methods

Along with point generation, we propose two schemes for generated weights: biasing towards a single label (bias-class) and biasing towards the mistakes of an unweighted tree (bias-error). These methods aim to show scenarios where fully stochastic sampling is likely to miss a large subset of points that our deterministic method is able to capture.

Bias-class weights were generated by choosing some class c and some value $v \in \mathbb{R} > 1$. We assign weight v for points where $y_i = c$ and 1 otherwise. By increasing the value of v, we are able to bias duplication to a single group of points. When a high probability is assigned to single label, fully stochastic sampling may completely miss points from the other classes, under-represent those regions. A decision tree fit with this data will miss entire classes of points, regardless of depth. In comparison, the stochastic portion of DaB duplication still samples each point of the low weight class with probability θ_i. Even if each individual θ_i has low probability, the chance of sampling no points for an entire class is low (with reasonable S). A decision tree trained on this data, while having holes in some regions, has a better chance of generalizing the data.

To generate weights for bias-error, we first had to fit a tree τ with un-duplicated dataset D. We set the weight of correctly classified points to 1 while setting the weights of incorrectly classified points to some value $v \in \mathbb{R} > 1$. Alternatively, the weights are defined by,

$$w_i = \mathbb{1}[\tau(x_i) = y_i] + v \cdot \mathbb{1}[\tau(x_i) \neq y_i].$$

Similar to bias-class, increasing the value of v increases the number of errors in the data, but this method aims to bias towards regions that are already difficult for classification. Like bias-class weights, we expect fully stochastic sampling to miss regions that were previously correct and DaB sampling to capture more of those regions. With sufficient representation of these low weight points, decision trees can represent the low weight regions while generating rules for the biased points.

3.3 Results

We found that our method produced consistently better weighted loss across a variety of experimental conditions. Our experiments used the GOSDT decision tree algorithm to fit sparse trees using a 2 class labels, dimension of 2, depth of 5, and regularization constant 0.001. We ran experiments for small ($N = 1500$), medium ($N = 6000$), and large ($N = 9000$) datasets to analyze the effect of p on the duplication algorithm in each case.

3.3.1 Weighted Errors

We first examine the loss of the bias-error weighing scheme (Table 1, Table 2, Table 3). We observed that our method performs worse on small N with Single Circular dataset for some p values. As noted in 3.1.2, decision trees have difficulty separating multiple circular regions in a dataset. Datasets with small N exacerbate this effect as fewer examples exist for separation. Still, as p increased this difference become negligible and our method produced the same results. For larger N, the difference in loss was better or roughly the same.

Additionally, our method performed about the same as random sampling on the MS dataset. This is likely due to the depth constraint imposed on our trees as multiple rules are required to adequately partition a single 3S region. Since the MS dataset contains many 3S regions, we would require deeper trees to accurately classify each 3S ball.

Finally, we examine the results of LSE with bias-error weighting. We find that the weighted loss does not change by much across the N and p values. This indicates that our trees were unable to find rules which isolated the "error" points within a class cluster. The tree is unable to capture these "error" points without potentially misclassifying a large portion of correct points. This effect is more apparent when examining the decision regions in Figure 3 and Figure 4.

	Bias Error Weighting on Small Dataset					
p	LSE ($p_{\text{wrong}} = 0.25$)		Single Circular		MS	
	DaB	Full Sampling	DaB	Full Sampling	DaB	Full Sampling
2.0	0.438 ± 0.000	0.457 ± 0.000	0.166 ± 0.013	0.173 ± 0.011	0.328 ± 0.011	0.342 ± 0.009
3.0	0.431 ± 0.000	0.448 ± 0.000	0.177 ± 0.012	0.203 ± 0.009	0.330 ± 0.011	0.343 ± 0.009
5.0	0.435 ± 0.000	0.441 ± 0.000	0.177 ± 0.012	0.164 ± 0.012	0.328 ± 0.011	0.338 ± 0.010
8.0	0.435 ± 0.000	0.440 ± 0.000	0.208 ± 0.013	0.183 ± 0.013	0.331 ± 0.011	0.345 ± 0.008
10.0	0.431 ± 0.000	0.436 ± 0.000	0.169 ± 0.013	0.184 ± 0.011	0.327 ± 0.011	0.345 ± 0.008
15.0	0.432 ± 0.000	0.436 ± 0.000	0.220 ± 0.010	0.220 ± 0.010	0.329 ± 0.011	0.335 ± 0.010
30.0	0.433 ± 0.000	0.436 ± 0.000	0.169 ± 0.013	0.191 ± 0.012	0.329 ± 0.011	0.333 ± 0.010
100.0	0.432 ± 0.000	0.435 ± 0.000	0.170 ± 0.012	0.165 ± 0.013	0.327 ± 0.011	0.333 ± 0.010

Table 1: Weighted loss across different p for small datasets (N=1500) when using a bias-error weighting scheme with $v = 3$

	Bias Error Weighting on Medium Dataset					
p	LSE($p_{\text{wrong}} = 0.25$)		Single Circular		MS	
	DaB	Full sampling	DaB	Full sampling	DaB	Full sampling
2.0	0.477 ± 0.000	0.476 ± 0.000	0.136 ± 0.004	0.186 ± 0.005	0.377 ± 0.014	0.376 ± 0.015
3.0	0.480 ± 0.000	0.483 ± 0.000	0.137 ± 0.004	0.153 ± 0.005	0.380 ± 0.014	0.381 ± 0.014
5.0	0.478 ± 0.000	0.479 ± 0.000	0.148 ± 0.004	0.153 ± 0.005	0.380 ± 0.014	0.383 ± 0.013
8.0	0.478 ± 0.000	0.482 ± 0.000	0.173 ± 0.002	0.189 ± 0.004	0.380 ± 0.014	0.381 ± 0.014
10.0	0.480 ± 0.000	0.482 ± 0.000	0.136 ± 0.004	0.163 ± 0.005	0.380 ± 0.014	0.381 ± 0.014
15.0	0.479 ± 0.000	0.481 ± 0.000	0.175 ± 0.002	0.194 ± 0.002	0.380 ± 0.014	0.381 ± 0.014
30.0	0.479 ± 0.000	0.481 ± 0.000	0.136 ± 0.004	0.137 ± 0.003	0.380 ± 0.014	0.381 ± 0.014
100.0	0.479 ± 0.000	0.479 ± 0.000	0.137 ± 0.004	0.139 ± 0.004	0.380 ± 0.014	0.383 ± 0.013

Table 2: Weighted loss across different p for medium datasets (N=6000) when using a bias-error weighting scheme with $v = 3$

We then examine the experiments using bias-class weights (Table 4, Table 5, Table 6). The LSE results for bias class follow a similar trend to bias-error weights as the weighted loss is relatively consistent for different N and p values. The tree is still unable to partition "error" points of LSE data without potentially misclassifying correct points in the region. This can be seen in the decision regions from Figure 3 and Figure 4.

	Bias Error Weighting on Large Dataset					
p	LSE($p_{\text{wrong}} = 0.25$)		Single Circular		MS	
	DaB	Full sampling	DaB	Full sampling	DaB	Full sampling
2.0	0.477 ± 0.000	0.478 ± 0.000	0.183 ± 0.011	0.207 ± 0.010	0.371 ± 0.015	0.370 ± 0.015
3.0	0.478 ± 0.000	0.476 ± 0.000	0.185 ± 0.011	0.197 ± 0.010	0.370 ± 0.015	0.383 ± 0.012
5.0	0.478 ± 0.000	0.477 ± 0.000	0.180 ± 0.011	0.174 ± 0.011	0.373 ± 0.014	0.376 ± 0.013
8.0	0.476 ± 0.000	0.478 ± 0.000	0.209 ± 0.009	0.208 ± 0.011	0.372 ± 0.014	0.372 ± 0.014
10.0	0.479 ± 0.000	0.475 ± 0.000	0.172 ± 0.011	0.169 ± 0.011	0.372 ± 0.014	0.363 ± 0.016
15.0	0.478 ± 0.000	0.475 ± 0.000	0.182 ± 0.009	0.181 ± 0.011	0.373 ± 0.014	0.368 ± 0.015
30.0	0.478 ± 0.000	0.476 ± 0.000	0.172 ± 0.011	0.172 ± 0.011	0.372 ± 0.014	0.373 ± 0.014
100.0	0.478 ± 0.000	0.478 ± 0.000	0.172 ± 0.011	0.163 ± 0.011	0.370 ± 0.015	0.372 ± 0.014

Table 3: Weighted loss across different p for large datasets (N=9000) when using a bias-error weighting scheme with $v = 3$

In general, our method tended to have similar results across p values for the spherical data. Since DaB duplication has less variance, the resulting unweighted datasets are more consistent across the p values. Similar to bias-error data, our DaB sampling struggles to generalize MS data with the limited depth budget but still outperforms fully random sampling. Finally, we observe that our duplication method outperforms fully stochastic sampling on the Single Circular data, achieving both lower loss and variance.

	Bias Class Weighting on Small Dataset					
p	LSE($p_{\text{wrong}} = 0.25$)		Single Circular		MS	
	DaB	Full sampling	DaB	Full sampling	DaB	Full sampling
2.0	0.247 ± 0.000	0.249 ± 0.000	0.091 ± 0.007	0.095 ± 0.006	0.219 ± 0.001	0.243 ± 0.003
3.0	0.247 ± 0.000	0.248 ± 0.000	0.091 ± 0.007	0.187 ± 0.011	0.219 ± 0.001	0.254 ± 0.003
5.0	0.246 ± 0.000	0.247 ± 0.000	0.091 ± 0.007	0.118 ± 0.011	0.219 ± 0.001	0.232 ± 0.002
8.0	0.247 ± 0.000	0.246 ± 0.000	0.141 ± 0.000	0.141 ± 0.000	0.219 ± 0.001	0.262 ± 0.002
10.0	0.247 ± 0.000	0.248 ± 0.000	0.091 ± 0.007	0.153 ± 0.012	0.219 ± 0.001	0.261 ± 0.002
15.0	0.247 ± 0.000	0.246 ± 0.000	0.141 ± 0.000	0.140 ± 0.000	0.219 ± 0.001	0.233 ± 0.002
30.0	0.247 ± 0.000	0.247 ± 0.000	0.091 ± 0.007	0.157 ± 0.011	0.219 ± 0.001	0.233 ± 0.002
100.0	0.247 ± 0.000	0.247 ± 0.000	0.091 ± 0.007	0.055 ± 0.000	0.219 ± 0.001	0.231 ± 0.002

Table 4: Weighted loss across different p for small datasets (N=1500) when using a bias-class weighting scheme with $v = 3$

	Bias Class Weighting on Medium Dataset					
p	LSE($p_{\text{wrong}} = 0.25$)		Single Circular		MS	
	DaB	Full sampling	DaB	Full sampling	DaB	Full sampling
2.0	0.248 ± 0.000	0.249 ± 0.000	0.063 ± 0.000	0.194 ± 0.010	0.280 ± 0.001	0.268 ± 0.002
3.0	0.248 ± 0.000	0.249 ± 0.000	0.063 ± 0.000	0.098 ± 0.006	0.280 ± 0.001	0.282 ± 0.001
5.0	0.248 ± 0.000	0.248 ± 0.000	0.063 ± 0.000	0.098 ± 0.006	0.280 ± 0.001	0.287 ± 0.002
8.0	0.248 ± 0.000	0.248 ± 0.000	0.145 ± 0.000	0.143 ± 0.000	0.280 ± 0.001	0.281 ± 0.001
10.0	0.248 ± 0.000	0.248 ± 0.000	0.063 ± 0.000	0.099 ± 0.006	0.280 ± 0.001	0.281 ± 0.001
15.0	0.248 ± 0.000	0.249 ± 0.000	0.143 ± 0.000	0.143 ± 0.000	0.280 ± 0.001	0.281 ± 0.001
30.0	0.248 ± 0.000	0.248 ± 0.000	0.063 ± 0.000	0.068 ± 0.000	0.280 ± 0.001	0.281 ± 0.001
100.0	0.248 ± 0.000	0.248 ± 0.000	0.063 ± 0.000	0.067 ± 0.000	0.280 ± 0.001	0.287 ± 0.002

Table 5: Weighted loss across different p for medium datasets (N=6000) when using a bias-class weighting scheme with $v = 3$

	Bias Class Weighting on Large Dataset					
p	LSE($p_{\text{wrong}} = 0.25$)		Single Circular		MS	
	DaB	Full sampling	DaB	Full sampling	DaB	Full sampling
2.0	0.248 ± 0.000	0.248 ± 0.000	0.101 ± 0.006	0.168 ± 0.010	0.274 ± 0.003	0.268 ± 0.002
3.0	0.248 ± 0.000	0.248 ± 0.000	0.101 ± 0.006	0.136 ± 0.009	0.274 ± 0.003	0.295 ± 0.000
5.0	0.248 ± 0.000	0.248 ± 0.000	0.101 ± 0.006	0.105 ± 0.006	0.274 ± 0.003	0.281 ± 0.001
8.0	0.248 ± 0.000	0.248 ± 0.000	0.144 ± 0.000	0.143 ± 0.000	0.274 ± 0.003	0.281 ± 0.001
10.0	0.248 ± 0.000	0.248 ± 0.000	0.101 ± 0.006	0.104 ± 0.006	0.274 ± 0.003	0.254 ± 0.002
15.0	0.248 ± 0.000	0.248 ± 0.000	0.143 ± 0.000	0.143 ± 0.000	0.274 ± 0.003	0.260 ± 0.002
30.0	0.248 ± 0.000	0.248 ± 0.000	0.101 ± 0.006	0.101 ± 0.006	0.274 ± 0.003	0.274 ± 0.003
100.0	0.248 ± 0.000	0.248 ± 0.000	0.101 ± 0.006	0.073 ± 0.000	0.274 ± 0.003	0.279 ± 0.002

Table 6: Weighted loss across different p for large datasets (N=9000) when using a bias-class weighting scheme with $v = 3$

3.3.2 Decision Boundaries

We plot the decision boundaries of GOSDT trees fit with each method to examine the general trends captured by duplication. We find that the decision regions on spherical data greatly varies depending on seed/randomization. We selected plots that displayed the overall trend: DaB sampling represents the points better than fully stochastic sampling.

As can be see in Figure 1, stochastic sampling is unable to capture a sufficient number of points for class 0 (the middle circle) and the tree trained on this data is unable to separate the middle region. DaB sampling duplicates has better replication of the middle points, allowing the tree to partition the space correctly. Although given a depth budget of 5, the fully stochastic tree uses only 2 rules as it is unable to find more splits. We see a similar effect in Figure 2 where fully stochastic sampling does not give sufficient weighting to the inner spheres, causing the tree to miss them entirely. Our duplication method incorporates sufficient points to define regions around the middle circles and inner circles.

In the case of LSE data in two dimensions, using only two rules would be sufficient for capturing the general trend of the data. Since we gave the trees a depth budget of five, we expect the learned regions to overfit on the data. However, as weighted loss shows, this did not happen most of the time. Often the trees could not find splits which isolated "error" points within a cluster without misclassifying many correct points. Figure 3 shows that these trees with the same loss can learn different regions. The tree fit with our method still captured the general trend better than the fully random sampling tree.

Still, some instances of LSE had "errors" which could be separated given sufficient depth. Figure 4 displays a case of LSE data which could be overfit by a GOSDT tree. We find that DaB sampling represents the numerous "errors" inside each region better, allowing the decision tree to overfit the data as expected. Fully stochastic sampling duplicates an insufficient number of "error" points, resulting in a tree that cannot use its depth budget to overfit.

4 Conclusion

Our method of duplication with stochastic and deterministic components approximates weighted datasets better than fully stochastic duplication. Experimental results show that DaB sampling has similar or better weighted loss compared to fully stochastic sampling. Additionally, our method had smaller variance in weighted loss and had more consistent results when increasing duplication. This allowed decision trees trained with our method to capture overall trends better and incorporate points with lower weight.

Future work can extend our duplication method to regression tasks and non-tree based models. Our experiments and bounds were only accounting for decision tree classifiers, but this same framework could be used for models that use a weighted loss function like weighted least squares.

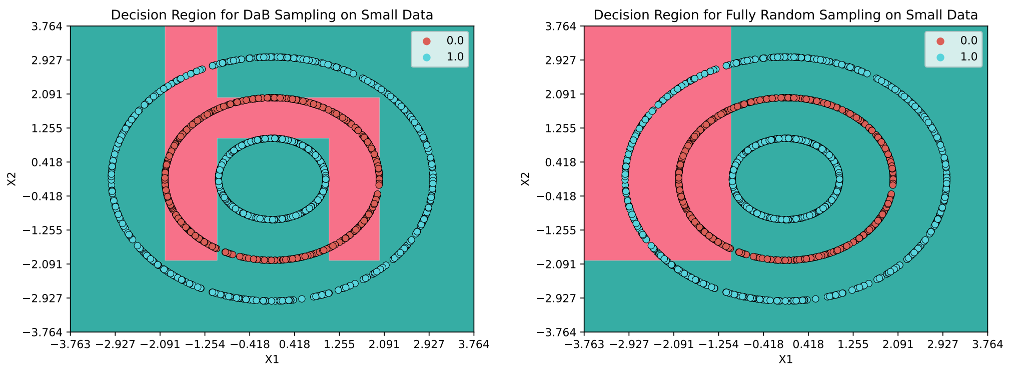

Figure 1: The decision regions classified by GOSDT for a single 3-sphere using bias-mistake weights after duplication. The tree trained with our duplication method (left) is able to better define the region around the middle circle. The points with fully stochastic duplication do not represent class 0 well, the tree fit with these points (right) fails at using the given depth to define regions in the middle circle.

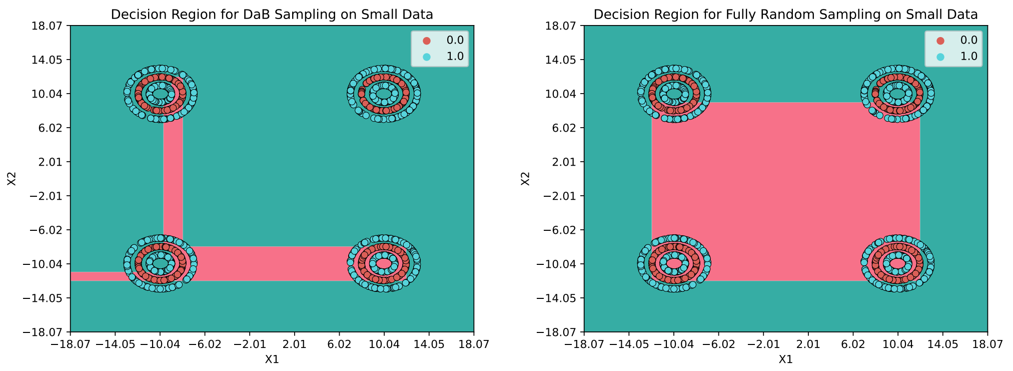

Figure 2: The decision regions classified by GOSDT for MS data using bias-mistake weights after duplication. Our duplication method (left) sufficiently represents points in the middle and inner circle, allowing the tree to define the middle regions. The tree with fully random sampling (right) has insufficient representation of the middle and inner regions, leading to a lack of separation in the region from the inner circle.

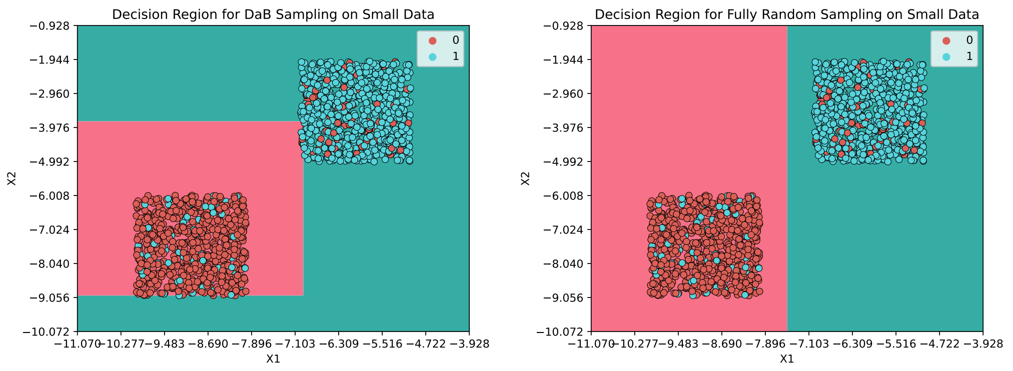

Figure 3: A typical decision regions defined by GOSDT on 25% error LSE data using bias-mistake weights after duplication. Although both trees have about the same weighted loss, our method (left) captures the general trend of the regions better and finds more rules than fully stochastic sampling (right)

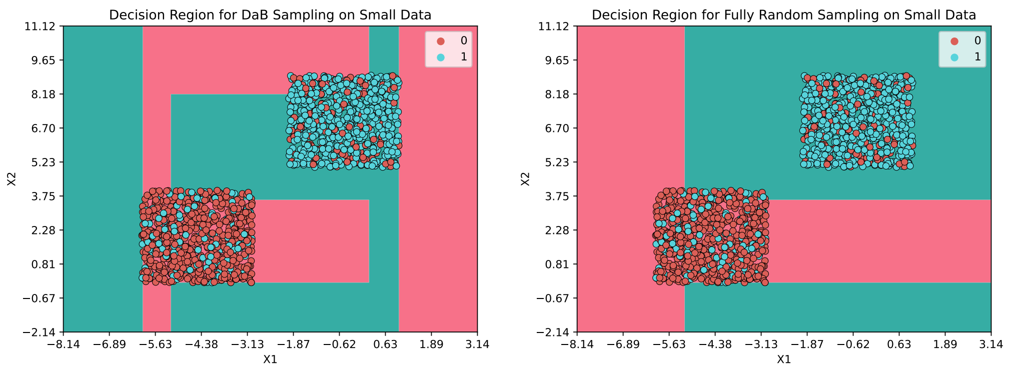

Figure 4: An infrequent case of overfitting in LSE data with 25% errors using bias-mistake weights. Our duplication method (left) captures the small errors within each cluster better than fully stochastic sampling (right). This allows the tree to overfit the data as expected on most linear, low dimensional data.

References

Ali Behrouz, Mathias Lecuyer, Cynthia Rudin, and Margo Seltzer. Fast optimization of weighted sparse decision trees for use in optimal treatment regimes and optimal policy design, 2022.

David A. Cieslak and Nitesh V. Chawla. Learning decision trees for unbalanced data. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 241–256, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-87479-9.

S. Deepak and P.M. Ameer. Brain tumor categorization from imbalanced mri dataset using weighted loss and deep feature fusion. *Neurocomputing*, 520:94–102, 2023. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2022.11.039. URL `https://www.sciencedirect.com/science/article/pii/S0925231222014266`.

Ivan Firdausi, Charles lim, Alva Erwin, and Anto Satriyo Nugroho. Analysis of machine learning techniques used in behavior-based malware detection. In *2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies*, pages 201–203, 2010. doi: 10.1109/ACT.2010.33.

Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17, 1976. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(76)90095-8. URL `https://www.sciencedirect.com/science/article/pii/0020019076900958`.

Nathaniel MacNell, Lydia Feinstein, Jesse Wilkerson, Pivi M Salo, Samantha A Molsberry, Michael B Fessler, Peter S Thorne, Alison A Motsinger-Reif, and Darryl C Zeldin. Implementing machine learning methods with complex survey data: Lessons learned on the impacts of accounting sampling weights in gradient boosting. *PLoS One*, 18(1):e0280387, January 2023.