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

Ali Behrouz, Mathias Lécuyer, Cynthia Rudin, Margo Seltzer

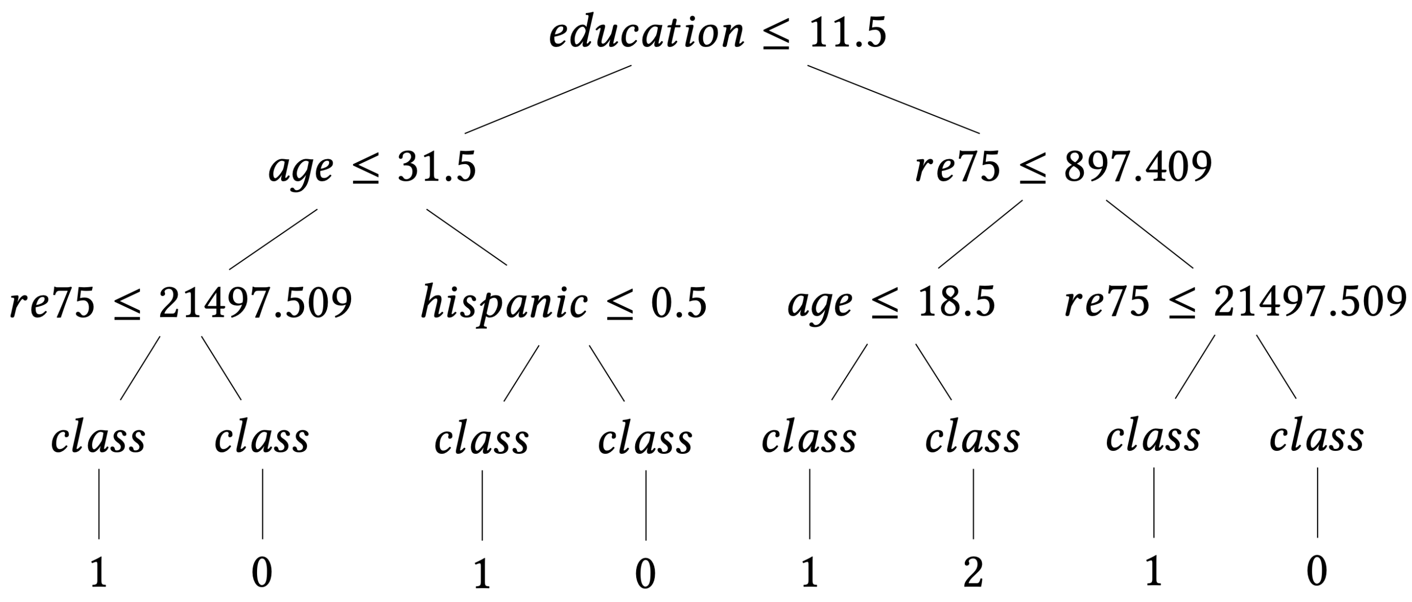
AIMLAI @ CIKM 2022

SYSTOPIA

DUKE UNIVERSITY 1838

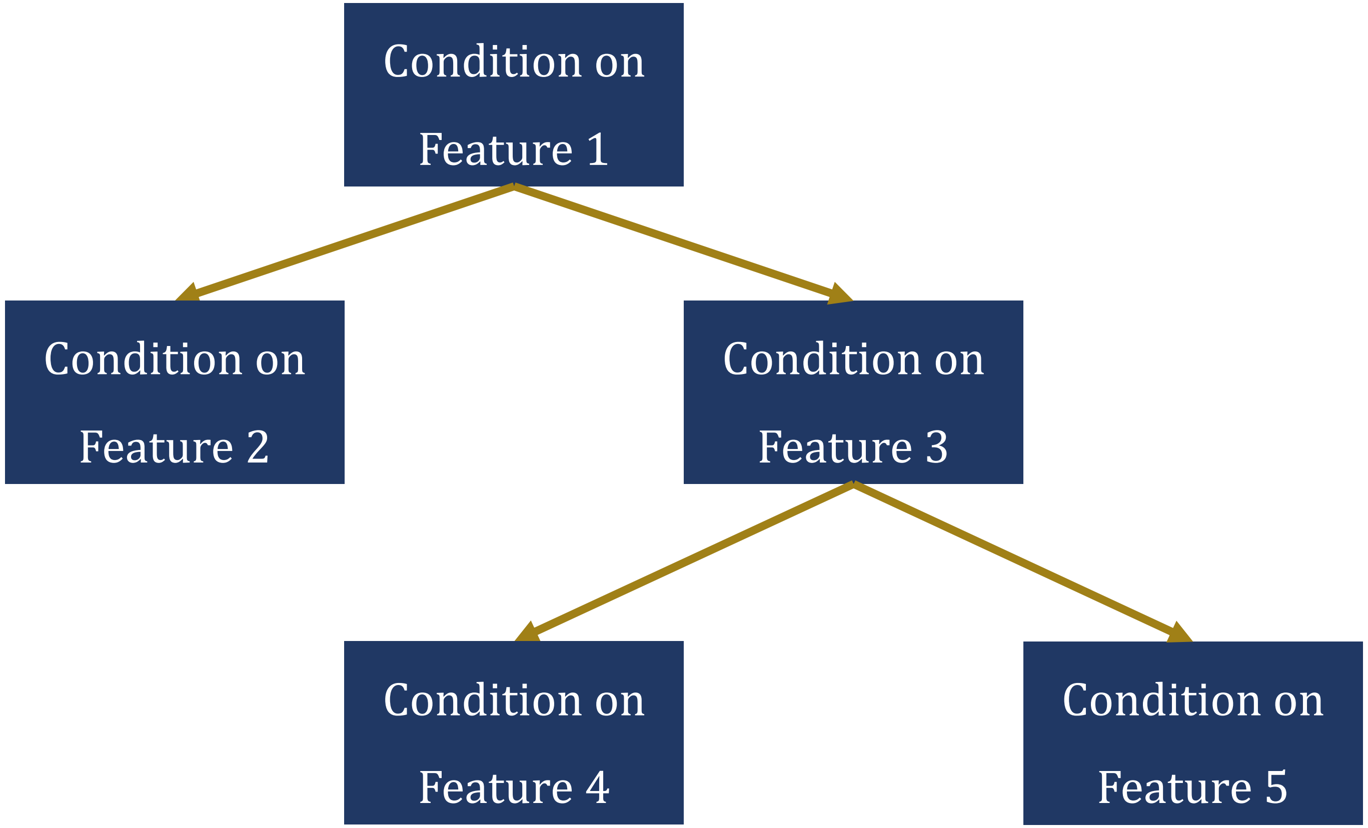
Decision Trees and Interpretability

- Decision trees are a family of interpretable models.
- What about accuracy?



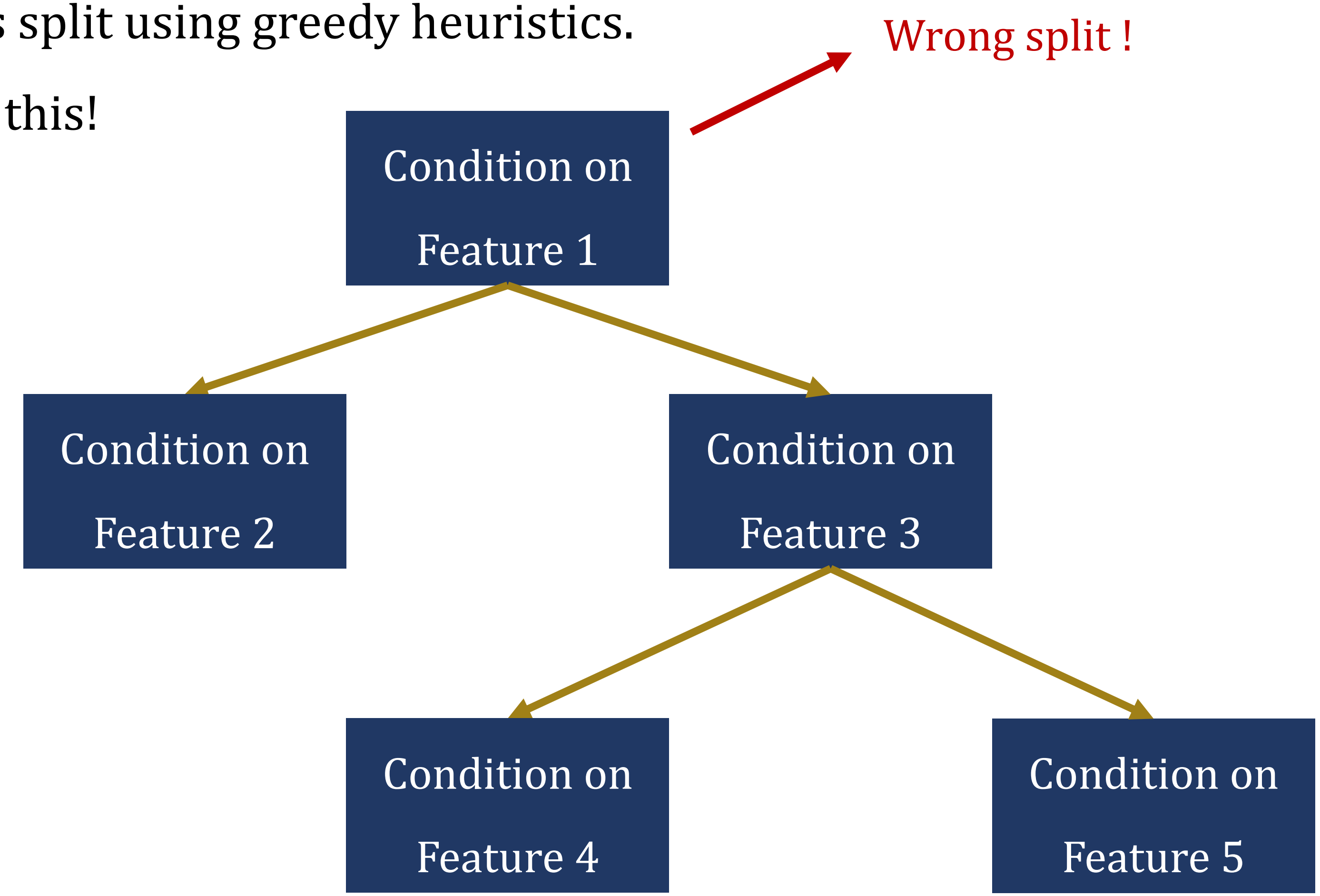
Greedy Algorithms for Decision Trees

- Old fashioned decision tree algorithms split using greedy heuristics.



Greedy Algorithms for Decision Trees

- Old fashioned decision tree algorithms split using greedy heuristics.
- Optimal decision tree models don't do this!



Optimal Sparse Decision Trees

- How to find the right decision tree?
- Optimize an objective over all possible decision trees!

Let $\mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[y_i \neq \hat{y}_i^t]$, find decision tree t such that:

$$\underset{t}{\text{minimize}} \quad \mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}), \ \text{s.t. depth}(t) \leq d$$

[1] Learning optimal decision trees using caching branch-and-bound search. Aglin et al. (AAAI 2020)

Optimal Sparse Decision Trees

- How to find the right decision tree?
- Optimize an objective over all possible decision trees!

We do not know the optimal depth!

Let $\mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[y_i \neq \hat{y}_i^t]$, find decision tree t such that:

$$\underset{t}{\text{minimize}} \quad \mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}), \ \text{s.t. depth}(t) \leq d$$

[1] Learning optimal decision trees using caching branch-and-bound search. Aglin et al. (AAAI 2020)

Optimal Sparse Decision Trees

- How to find the right decision tree?
- Optimize an objective over all possible decision trees!

Let $\mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[y_i \neq \hat{y}_i^t]$, find decision tree t such that:

$$\underset{t}{\text{minimize}} \quad \mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}), \ \text{s.t. depth}(t) \leq d$$

Let $\mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[y_i \neq \hat{y}_i^t]$, find decision tree t such that:

$$\underset{t}{\text{minimize}} \quad \mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}) + \underbrace{\lambda H_t}_{\text{Sparsity}}$$

[1] Learning optimal decision trees using caching branch-and-bound search. Aglin et al. (AAAI 2020)
[2] Optimal sparse decision trees. Hu et al. (Neurips 2019)
[3] Generalized and scalable optimal sparse decision trees. Lin et al. (ICML 2020)

Optimal Sparse Decision Trees

- How to find the right decision tree?
- Optimize an objective over all possible decision trees!

Let $\mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[y_i \neq \hat{y}_i^t]$, find decision tree t such that:

$$\underset{t}{\text{minimize}} \quad \mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}), \ \text{s.t. depth}(t) \leq d$$

Harder to solve!

Let $\mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[y_i \neq \hat{y}_i^t]$, find decision tree t such that:

$$\underset{t}{\text{minimize}} \quad \mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}) + \underbrace{\lambda H_t}_{\text{Sparsity}}$$

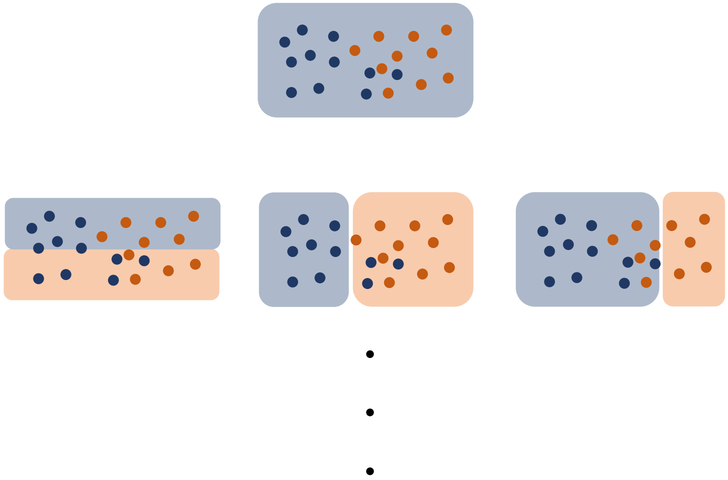
[1] Learning optimal decision trees using caching branch-and-bound search. Aglin et al. (AAAI 2020)
[2] Optimal sparse decision trees. Hu et al. (Neurips 2019)
[3] Generalized and scalable optimal sparse decision trees. Lin et al. (ICML 2020)

Branch & Bound Algorithm and its Limitations

- Start with the full dataset and a majority class label.
- Iteratively split it into subsets using each feature.
- Use some computational tricks to save time.

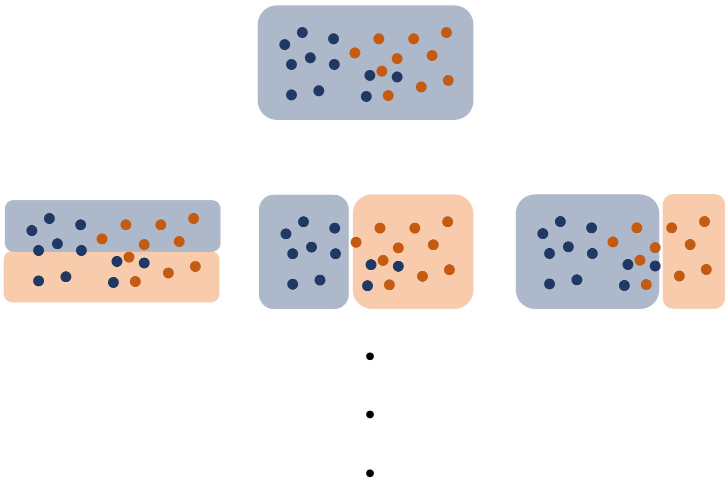
Branch & Bound Algorithm and its Limitations

- Start with the full dataset and a majority class label.
- Iteratively split it into subsets using each feature.
- Use some computational tricks to save time.



Branch & Bound Algorithm and its Limitations

- Start with the full dataset and a majority class label.
- Iteratively split it into subsets using each feature.
- Use some computational tricks to save time.



✶ These algorithms are slow due to continuous features and convergence of bounds.

Guessing Techniques

- McTavish et al. have addressed these limitations.

Let $\mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[y_i \neq \hat{y}_i^t]$, find decision tree t such that:

$$\underset{t}{\text{minimize}} \quad \mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}) + \lambda H_t \text{ s.t. depth}(t) \leq d$$

- Use a black box reference model to inform the search!
  1. Threshold guessing
  2. Lower bound guessing
  3. Depth bound guessing

[4] Fast Sparse Decision Tree Optimization via Reference Ensembles. McTavish et al. (AAAI 2022)

How to Incorporate Weights?

- These algorithms **cannot** handle weighted data samples!
- Existing techniques **cannot** produce policies that incorporate inverse propensity weighting on individual data points!

- Example: In policy design, Imbalanced datasets, fairness, treatment regimes, different cost of misclassification, etc.

How to Incorporate Weights? (Cont.)

- We present three algorithms for efficient sparse weighted decision tree optimization:
  - ✶ **Directly optimize the weighted loss function,**
  - ✶ **Data duplication,**
  - ✶ **Weighted sampling.**

Approach 1: Direct Approach

- Directly optimize the weighted loss function:

Let $\mathcal{L}_{\mathbf{w}}(\mathcal{T}, \tilde{\mathrm{x}}, \mathrm{y}) = \frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} \mathbb{1}[y_i \neq \hat{y}_i^{\mathcal{T}}] \times w_i$, find decision tree $\mathcal{T}$ such that:

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

- We adapt the branch-and-bound algorithm with guessing technique of McTavish et al. [4] to support weighted samples.

[4] Fast Sparse Decision Tree Optimization via Reference Ensembles. McTavish et al. (AAAI 2022)

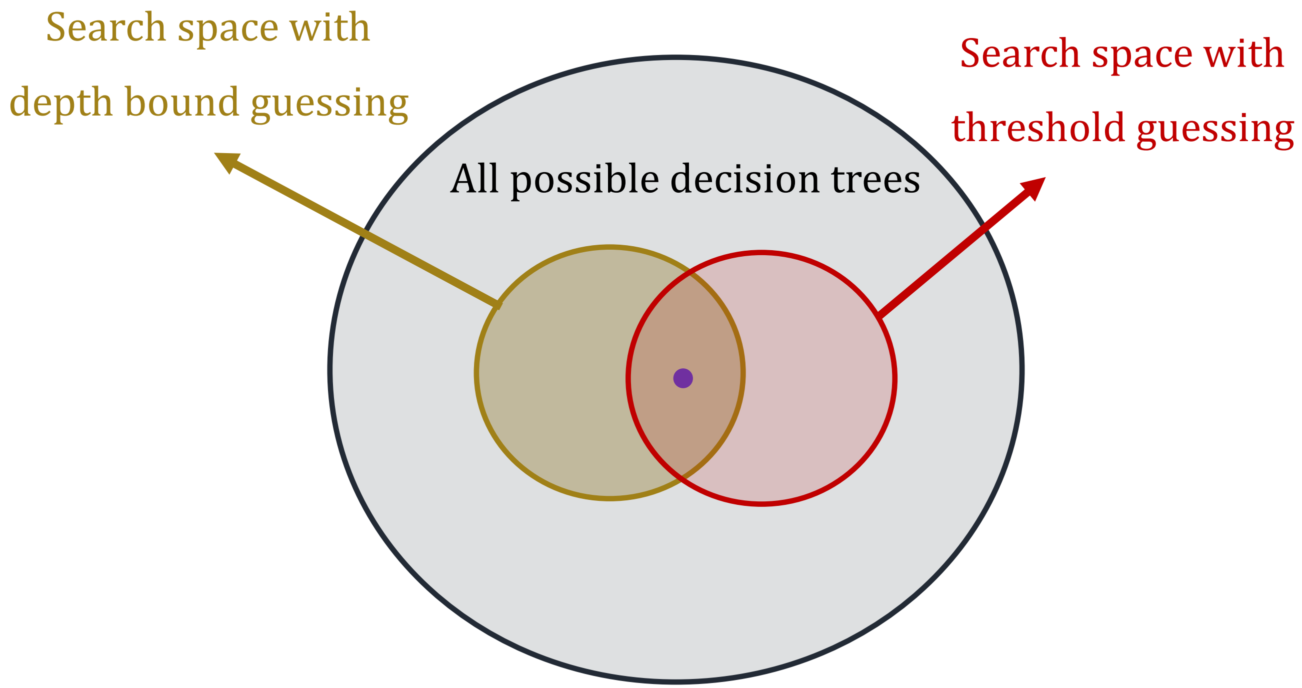
Theoretical Guarantees

- Guessing Techniques:
    1. Threshold guessing
    2. Lower bound guessing
    3. Depth bound guessing

[4] Fast Sparse Decision Tree Optimization via Reference Ensembles. McTavish et al. (AAAI 2022)

Theoretical Guarantees

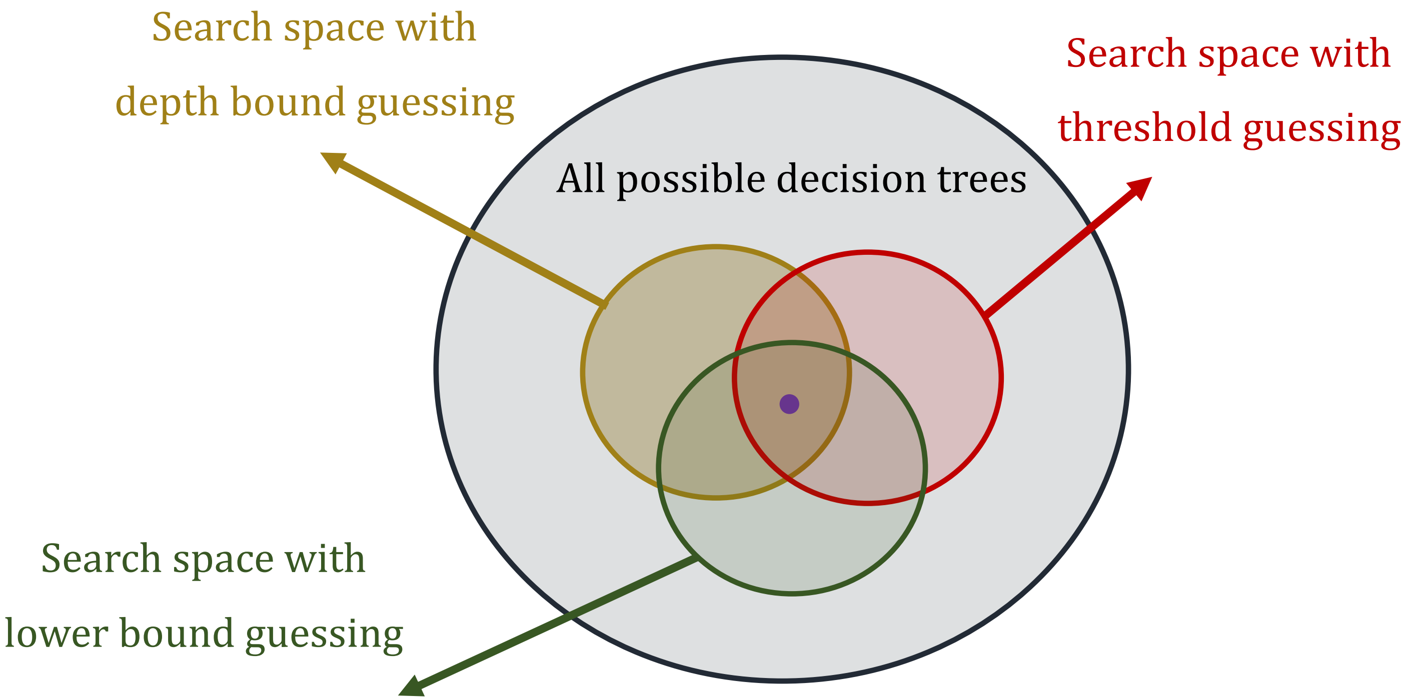
- Guessing Techniques:
    1. **Threshold guessing**
    2. Lower bound guessing
    3. **Depth bound guessing**



[4] Fast Sparse Decision Tree Optimization via Reference Ensembles. McTavish et al. (AAAI 2022)

Theoretical Guarantees

- Guessing Techniques:
  1. Threshold guessing
  2. **Lower bound guessing**
  3. Depth bound guessing



[4] Fast Sparse Decision Tree Optimization via Reference Ensembles. McTavish et al. (AAAI 2022)

Limitation of Direct Approach

- In decision tree optimization, evaluation of the objective is performed repeatedly.
- Recall the objective function:

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

- Computing the objective function requires computing the inner product of weights and indicator vector of misclassifications.

Limitation of Direct Approach (cont.)

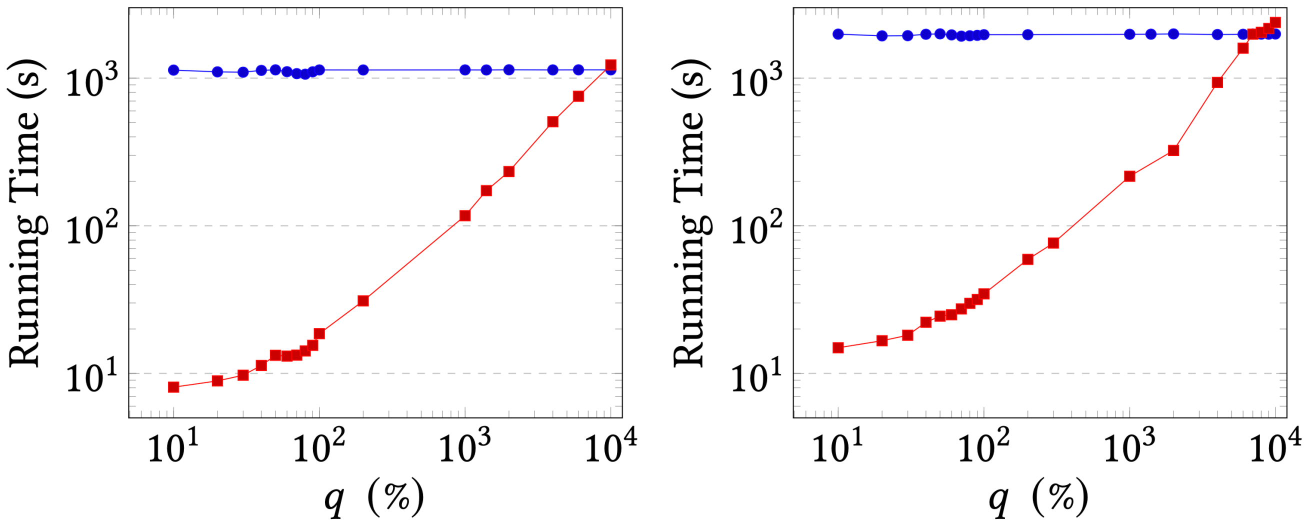
- Bit operations are two orders of magnitude faster than standard inner product.

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

(a) Weighted Loss

$$\mathcal{L}(t, \tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}[y_i \neq \hat{y}_i^t]$$

(b) Unweighted Loss



Approach 2: Data-duplication

- Data Duplication Algorithm:
    1. Normalize weights.
    2. Scale the normalized weights by a factor p.
    3. Round each to its nearest integer.
- Use any unweighted optimal decision tree algorithm on the duplicated dataset.

Algorithm 1: Data Duplication

Input : Dataset $\mathbf{x}, \mathbf{y}$ and weights $\mathbf{w}$, duplication factor $p < 100$

Output : Duplicated dataset $\tilde{X}$, $\tilde{y}$

$\tilde{X} \leftarrow \emptyset$; $\tilde{y} \leftarrow \emptyset$;

Define $\tilde{w}_i = \text{round}\left(p \cdot \left(\frac{w_i}{\sum_{i=1}^{N} w_i}\right)\right)$;

for $x_i \in \mathbf{x}$ **do**

  for $i = 1, 2, \ldots, \tilde{w}_i$ **do**

    $\tilde{X} \leftarrow \tilde{X} \cup \{x_i\}$;

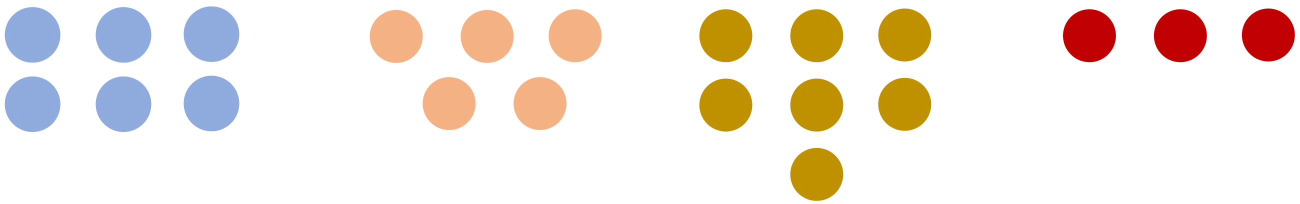
    $\tilde{y} \leftarrow \tilde{y} \cup \{y_i\}$;

return $\tilde{X}$, $\tilde{y}$

Approach 2: Data-duplication (Cont.)

Weighted Dataset:	3.4	2.7	4.1	1.6
Normalized Weights:	0.2881	0.2288	0.3475	0.1356
Scaled Weights ($p = 20$):	5.7627	4.5763	6.9492	2.7119
Rounded Weights:	6	5	7	3

Duplicated Dataset:



Correctness of Data Duplication

- Normalizing and scaling weights do not change the optimal solution.
- Rounding to integers can affects the solution.
- We will not lose substantial performance when using the rounded solution, as long as we did not change the weights very much when rounding.

Correctness of Data Duplication

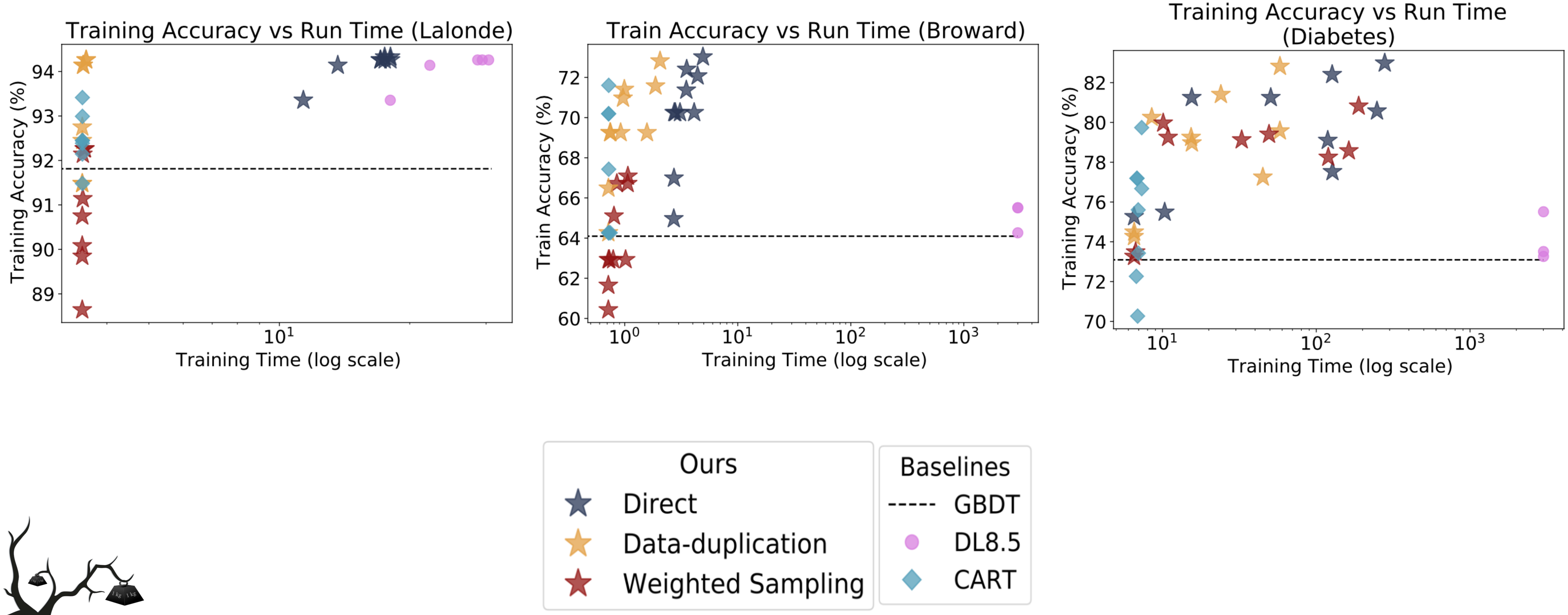
- Normalizing and scaling weights do not change the optimal solution.
- Rounding to integers can affects the solution.
- We will not lose substantial performance when using the rounded solution, as long as we did not change the weights very much when rounding.
- When the ratio of the biggest weight over the smallest weight is large, the data duplication approach might be inefficient.

Approach 3: Weighted Sampling

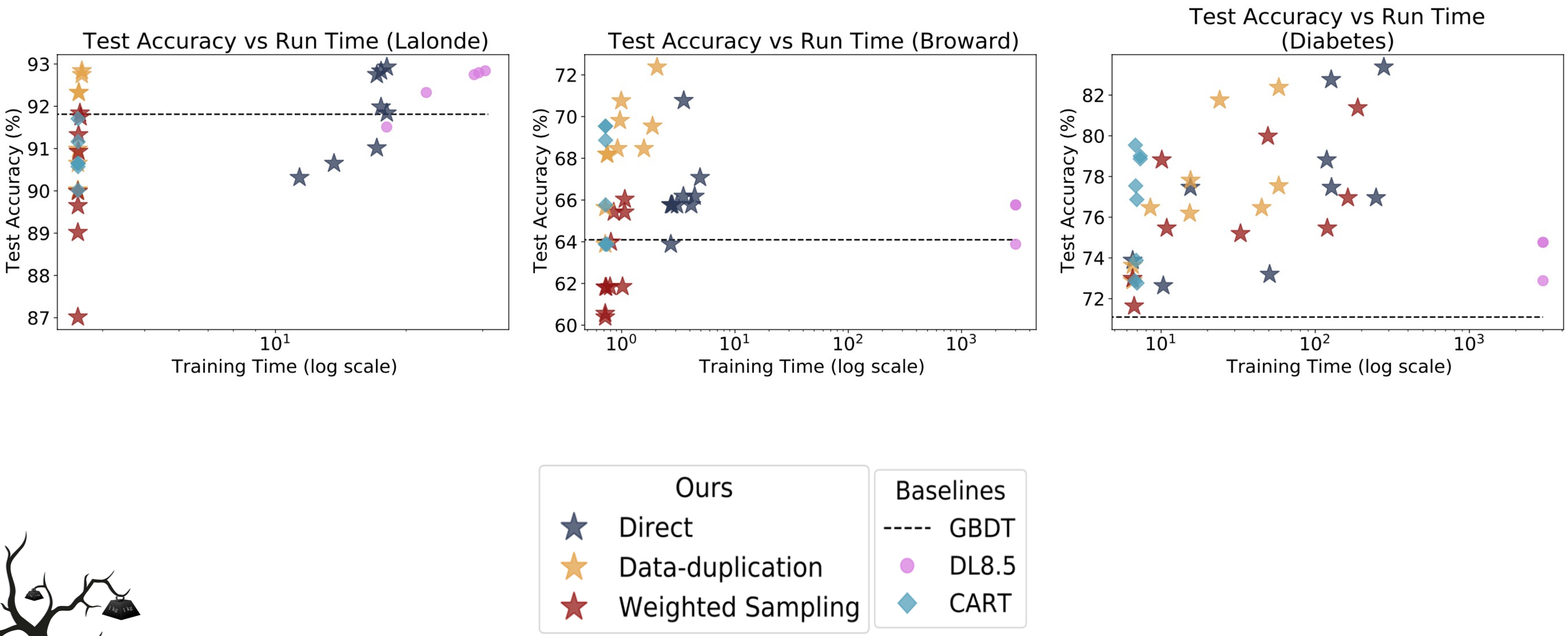
- Weighted Sampling:
  - Given an arbitrary number r, we sample $S = r \times N$ data points.
  - The probability of choosing x_i is $\frac{w_i}{\sum_{k=1}^{N} w_k}$.
- Use any unweighted optimal decision tree algorithm on the sampled dataset.
- **With high probability, we will not lose substantial performance when using the sampled dataset!**

Experiments

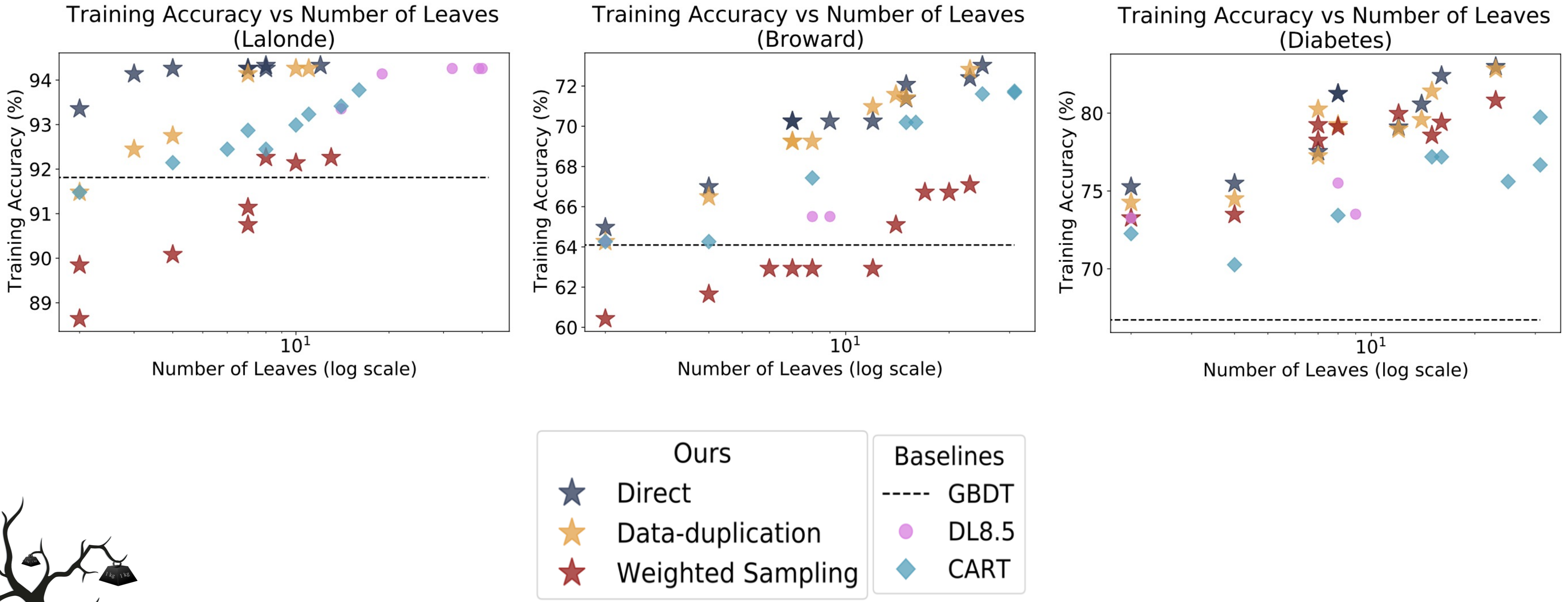
Experiments: Training Time vs. Training Accuracy



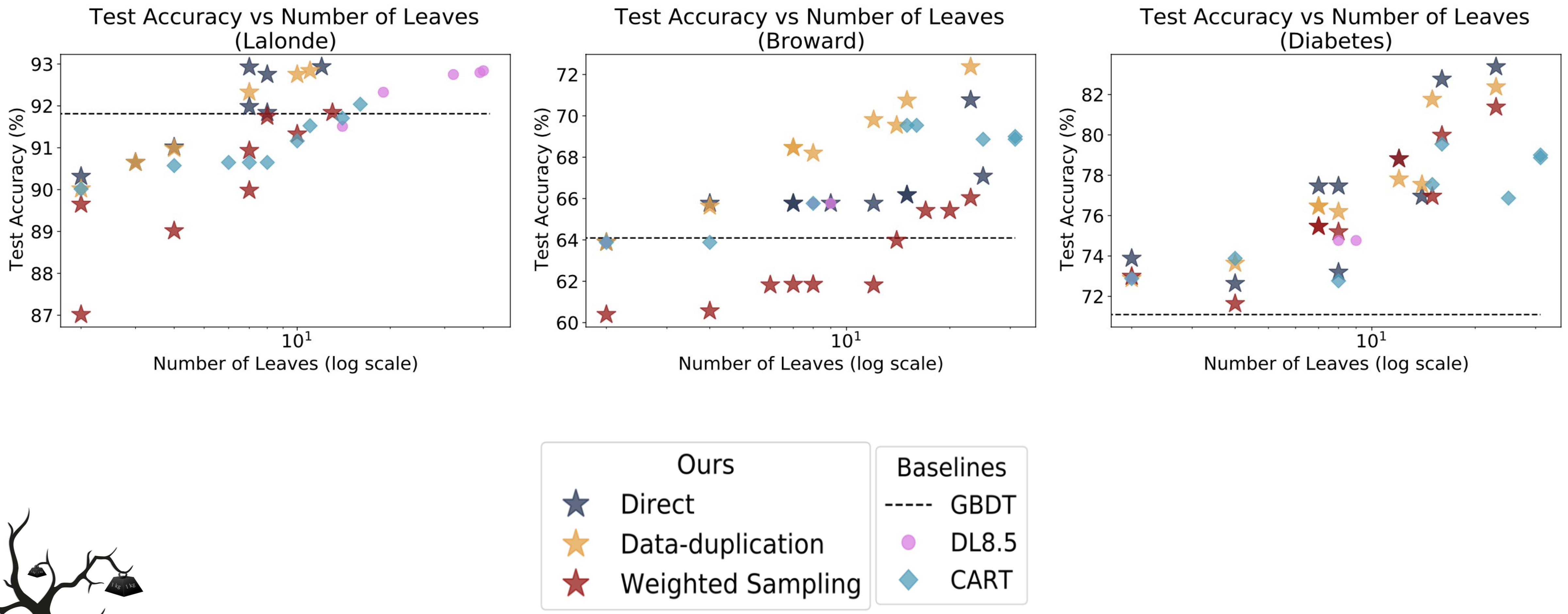
Experiments: Training Time vs. Test Accuracy



Experiments: Sparsity vs. Training Accuracy

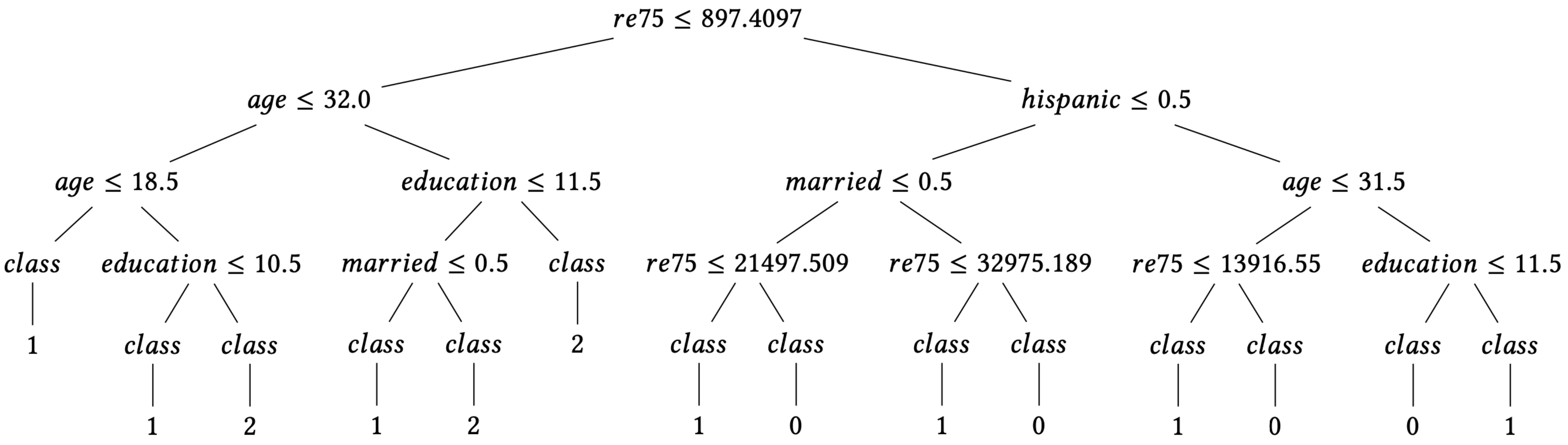


Experiments: Sparsity vs. Test Accuracy



How Can Our Approach be Used for Policy Making?

- Lalonde: 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.
- We use MALTS model [5] to estimate the missing outcome by matching.



[5] MALTS: Matching After Learning to Stretch. Parikh et al. (JMLR 2022)

Conclusion

Our contributions are:

- ✶ We suggest an effective approach to directly optimize the weighted loss function.
- ✶ To improve scalability, we transform weights to integer values and use data duplication to transform the weighted decision tree optimization problem into an unweighted counterpart.
- ✶ To scale to much larger datasets, we suggest a randomized procedure that samples each data point with a probability proportional to its weight.
- ✶ We present theoretical guarantees on the quality of proposed methods.

Code and Datasets: https://github.com/ubc-systopia/gosdt-guesses

Experiments: Sample Size vs. Training Accuracy