
A Scalable MIP-based Method for Learning Optimal Multivariate Decision Trees

Haoran Zhu, Pavankumar Murali, Dzung T. Phan, Lam M. Nguyen, Jayant R. Kalagnanam
IBM Research, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA
haoran@ibm.com, pavanm@us.ibm.com, phandu@us.ibm.com,
LamNguyen.MLTD@ibm.com, jayant@us.ibm.com

Abstract

Several recent publications report advances in training optimal decision trees (ODT) using mixed-integer programs (MIP), due to algorithmic advances in integer programming and a growing interest in addressing the inherent suboptimality of heuristic approaches such as CART. In this paper, we propose a novel MIP formulation, based on a 1-norm support vector machine model, to train a multivariate ODT for classification problems. We provide cutting plane techniques that tighten the linear relaxation of the MIP formulation, in order to improve run times to reach optimality. Using 36 data-sets from the University of California Irvine Machine Learning Repository, we demonstrate that our formulation outperforms its counterparts in the literature by an average of about 10% in terms of mean out-of-sample testing accuracy across the data-sets. We provide a scalable framework to train multivariate ODT on large data-sets by introducing a novel linear programming (LP) based data selection method to choose a subset of the data for training. Our method is able to routinely handle large data-sets with more than 7,000 sample points and outperform heuristics methods and other MIP based techniques. We present results on data-sets containing up to 245,000 samples. Existing MIP-based methods do not scale well on training data-sets beyond 5,500 samples.

1 Introduction

Decision tree models have been used extensively in machine learning (ML), mainly due to their transparency which allows users to derive interpretation on the results. Standard heuristics, such as CART [6], ID3 [17] and C4.5 [18], help balance gains in accuracy with training times for large-scale problems. However, these greedy top-down approaches determine the split at each node one-at-a-time without considering future splits at subsequent nodes. This means that splits at nodes further down in the tree might affect generalizability due to weak performance. Pruning is typically employed to address this issue, but this means that the training happens in two steps – first, top-down training and then pruning to identify better (stronger) splits. A better approach would be a one-shot training of the entire tree that determines splits at each node with full knowledge of all future splits, while optimizing, say, the misclassification rate [4]. The obtained decision tree is usually referred to as an *optimal decision tree (ODT)* in the literature.

The discrete nature of decisions involved in training a decision tree has inspired researchers in the field of Operations Research to encode the process using a mixed-integer programming (MIP) framework [5, 12, 19, 22, 23]. This has been further motivated by algorithmic advances in integer optimization [4]. An MIP-based ODT training method is able to learn the entire decision tree in a single step, allowing each branching rule to be determined with full knowledge of all the remaining rules. Papers on ODT have used optimality criteria such as the average testing length [14], training accuracy, the combination of training accuracy and model interpretability [4, 13], and the combination of training

accuracy and fairness [1]. In a recent paper, Aghaei et al. [2] propose a strong max-flow based MIP formulation to train univariate ODT for binary classification. The flexibility given by the choice of different objective functions, as well as linear constraints in the MIP model, also allows us to train optimal decision trees under various optimality criteria.

Yet another drawback of heuristic approaches is the difficulty in training multivariate (or oblique) decision trees, wherein splits at a node use multiple variables, or hyperplanes. While multivariate splits are much stronger than univariate (or axis-parallel) splits, they are more complicated because the splitting decision at each node cannot be enumerated. Many approaches have been proposed to train multivariate decision trees including the use of support vector machine (SVM) [3], logistic regression [21] and Householder transformation [25]. As noted in [4], these approaches do not perform well on large-scale data-sets as they also rely on top-down induction for the training process.

The *first contribution* of this paper is a new MIP formulation, SVM1-ODT, for training multivariate decision trees. Our formulation differs from others in the literature – we use a 1-norm SVM to maximize the number of correctly classified instances and to maximize the margin between clusters at the leaf nodes. We show that this formulation produces ODTs with a higher out-of-sample accuracy compared to the ODTs trained from state-of-the-art MIP models and heuristic methods on 20 data-sets selected from the UCI ML repository. In Sect. 4, we report the testing performance of the ODT trained using our formulation and show that it has an average improvement of 6-10% in mean out-of-sample accuracy for a decision tree of depth 2, and an average improvement of 17-20% for a decision tree of depth 3.

The *second contribution* of this paper is towards increasing tractability of the MIP-based ODT training for very large data-sets that are typical in real-world applications. It is imperative to bear in mind that the tractability of MIPs limits the size of the training data that can be used. Prior MIP-based ODT training formulations [4, 23] are intractable for large-sized data-sets (more than 5000 samples) since the number of variables and constraints increase linearly with the size of the training data. We address tractability in two steps. First, we tighten the LP-relaxation of SVM1-ODT by providing new cutting planes and getting rid of the big-M constant. Second, we propose an efficient linear programming (LP) -based data-selection method to be used prior to training the tree. This step is comprised of selecting a subset of data points that maximizes the information captured from the entire data-set.

Our method is able to routinely handle large data-sets with more than 7,000 sample points. We present results on data-sets containing up to 245,000 samples. Existing MIP-based methods do not scale well on training data beyond 5,000 samples, and do not provide a significant improvement over a heuristic approach. For large-scale data-sets, when SVM1-ODT is used along with the LP-based data selection method, our results indicate that the resulting decision trees offer higher training and testing accuracy, compared to CART (see Sect. 4, Figure 2). However, solely using any MIP-based formulation (including ours) without data selection can rarely outperform CART, due to the model becoming intractable resulting in the MIP solver failing to find any better feasible solution than the initially provided warm-start solution. This indicates that any loss of information from data-selection is more than adequately compensated by the use of optimal decision trees (using the SVM1-ODT formulation).

2 MIP formulation for training multivariate ODT for classification

In this section, we present our formulation to train an optimal multivariate classification tree using a data-set comprising numerical features, and for general data-sets containing categorical features, we propose an extension of such formulation in the supplementary material. For any $n \in \mathbb{Z}_+$, let $[n] := \{1, 2, \dots, n\}$ denote a finite set of data points, $[Y] = \{1, 2, \dots, Y\}$ be a set of class labels, and $[d] = \{1, 2, \dots, d\}$ be the index set of all features. Our formulation is established for the balanced binary tree with depth D . Let the set of *branch nodes* of the tree be denoted by $\mathcal{B} := \{1, \dots, 2^D - 1\}$, and the set of *leaf nodes* be denoted by $\mathcal{L} := \{2^D, \dots, 2^{D+1} - 1\}$. Similar to [4], let $A_R(l)$ and $A_L(l)$ denote the sets of ancestors of leaf node l whose right and left branches, respectively, are on the path from the root node to leaf node l .

Next, we define the variables to be used. Each data point $i \in [n]$ is denoted by (\mathbf{x}_i, y_i) , where \mathbf{x}_i is a d -dimensional vector, and $y_i \in [Y]$. Since we train multivariate trees, we use a branching hyperplane at branch node b , denoted by $\langle \mathbf{h}_b, \mathbf{x}_i \rangle = g_b$, where g_b is the bias term in the hyperplane. Indicator binary variable $c_i = 1$ when i is misclassified and 0 otherwise. Indicator binary variable $e_{il} = 1$

when i enters leaf node l . Variable $\hat{y}_i \in [Y]$ denotes the predicted label for i , and the decision variable $u_l \in [Y]$ is the label assigned to leaf node l . We let m_{ib} denote the slack variable for the soft margin for each point i corresponding to a hyperplane $\langle \mathbf{h}_b, \mathbf{x}_i \rangle = g_b$ used in the SVM-type model (3). The objective for the learning problem shown in (1) attempts to minimize the total misclassification ($\sum_i c_i$), the 1-norm SVM margin ($\sum_b \|\mathbf{h}_b\|_1$) and the sum of slack variables for classification ambiguity subtracted from the soft margin ($\sum_{i,b} m_{ib}$). Additionally, $\sum_b \|\mathbf{h}_b\|_1$ helps promote sparsity in the decision hyperplanes constructed at the branch nodes of a decision tree during the training process.

Then, **SVM1-ODT** can be expressed as follows:

$$\min \sum_{i \in [n]} c_i + \alpha_1 \sum_{i \in [n], b \in \mathcal{B}} m_{ib} + \alpha_2 \sum_{b \in \mathcal{B}} \|\mathbf{h}_b\|_1 \quad (1a)$$

$$\text{s.t. } (y_i - Y)c_i \leq y_i - \hat{y}_i \leq (y_i - 1)c_i, \forall i \in [n] \quad (1b)$$

$$\hat{y}_i = \sum_{l \in \mathcal{L}} w_{il}, \forall i \in [n] \quad (1c)$$

$$w_{il} \geq e_{il}, u_l - w_{il} + e_{il} \geq 1, \forall i \in [n], l \in \mathcal{L} \quad (1d)$$

$$Y e_{il} + u_l - w_{il} \leq Y, w_{il} \leq Y e_{il}, \forall i \in [n], l \in \mathcal{L} \quad (1e)$$

$$g_b - \sum_{j \in [d]} h_{bj} x_{ij} = p_{ib}^+ - p_{ib}^-, \forall i \in [n], b \in \mathcal{B} \quad (1f)$$

$$p_{ib}^+ \leq M(1 - e_{il}), \forall i \in [n], l \in \mathcal{L}, b \in A_R(l) \quad (1g)$$

$$p_{ib}^- + m_{ib} \geq \epsilon e_{il}, \forall i \in [n], l \in \mathcal{L}, b \in A_R(l) \quad (1h)$$

$$p_{ib}^- \leq M(1 - e_{il}), \forall i \in [n], l \in \mathcal{L}, b \in A_L(l) \quad (1i)$$

$$p_{ib}^+ + m_{ib} \geq \epsilon e_{il}, \forall i \in [n], l \in \mathcal{L}, b \in A_L(l) \quad (1j)$$

$$\sum_{l \in \mathcal{L}} e_{il} = 1, \forall i \in [n] \quad (1k)$$

$$\hat{y}_i, w_{il}, h_{bj}, g_b \in \mathbb{R}, p_{ib}^+, p_{ib}^-, m_{ib} \in \mathbb{R}_+, e_{il}, c_i \in \{0, 1\}, u_l \in \{1, \dots, Y\}. \quad (1l)$$

We notice that \mathbf{h}_b , g_b and u_l are main decision variables to characterize a decision tree, while \hat{y}_i , w_{il} , p_{ib}^+ , p_{ib}^- , m_{ib} , e_{il} and c_i are derived variables for the MIP model.

Constraint (1b) expresses the relationship between c_i and \hat{y}_i . If $\hat{y}_i = y_i$, then $c_i = 0$ since c_i is minimized. If $\hat{y}_i \neq y_i$ and $\hat{y}_i < y_i$, then $(y_i - 1)c_i \geq y_i - \hat{y}_i \geq 1$, thus $c_i = 1$. Similarly, if $\hat{y}_i > y_i$, then $(y_i - Y)c_i \leq -1$, thus $c_i = 1$. In a balanced binary tree with a fixed depth, the branching rule is given by: i goes to the left branch of b if $\langle \mathbf{h}_b, \mathbf{x}_i \rangle \leq g_b$, and goes to the right side otherwise. Predicted class $\hat{y}_i = \sum_{l \in \mathcal{L}} u_l \cdot e_{il}$. Since $u_l \cdot e_{il}$ is bilinear, we perform McCormick relaxation [15] of this term using an additional variable w_{il} such that $w_{il} = u_l \cdot e_{il}$. Since e_{il} is binary, this McCormick relaxation is exact. That is to say, $\hat{y}_i = \sum_{l \in \mathcal{L}} u_l \cdot e_{il}$ if and only if (1c)-(1e) hold for some extra variables w . Since u_l and e_{il} are integer variables, it follows that \hat{y}_i also integral. Constraints (1f), (1g) and (1i) formulate the branching rule at each node $b \in \mathcal{B}$: if i goes to the left branch at node b , then $g_b - \sum_{j \in [d]} h_{bj} x_{ij} \geq 0$, and if it goes to the right side, then $g_b - \sum_{j \in [d]} h_{bj} x_{ij} \leq 0$. As per MIP convention, we formulate this relationship by separating $g_b - \sum_{j \in [d]} h_{bj} x_{ij}$ into a pair of complementary variables p_{ib}^+ and p_{ib}^- (meaning p_{ib}^+ and p_{ib}^- cannot both be strictly positive at the same time), and forcing one of these two variables to be 0 through the big-M method [26]. We should remark that this is not exactly the same as our branching rule: when i goes to the right branch, it should satisfy $g_b - \sum_{j \in [d]} h_{bj} x_{ij} < 0$ strictly. The only special case is when $p_{ib}^+ = p_{ib}^- = 0$. However, due to the two constraints (1h), (1j), and the penalizing term $m_{ib} \geq 0$, this phenomenon cannot occur. Constraint (1k) enforces that each i should be assigned to exactly one leaf node. For a given dataset, model parameters ϵ , α_1 and α_2 are tuned via cross-validation. In the following section, we provide explanations for constraints (1h), (1j) and objective function (1a), and some strategies to tighten SVM1-ODT.

2.1 Multi-hyperplane SVM model for ODT

When constructing a branching hyperplane, we normally want to maximize the shortest distance from this hyperplane to its closest data points. For any branching node associated with a hyperplane $\langle \mathbf{h}_b, \mathbf{x}_i \rangle = g_b$ by fixing other parts of the tree, we can view the process of learning the hyperplane as constructing a binary classifier over data points $\{(\mathbf{x}_i, \bar{y}_i)\}$ that reach the node. The artificial label $\bar{y}_i \in \{\text{left}, \text{right}\}$ ($\equiv \{-1, +1\}$) is derived from the child of the node: \mathbf{x}_i goes to the left or the right; that is determined by e_{il} . This problem is reduced to an SVM problem for each branching node. Applying the 1-norm SVM model with soft margin [16, 27] to the node b gives us

$$\begin{aligned} \min \quad & \frac{1}{2} \|\mathbf{h}_b\|_1 + \alpha_1 \sum_{i \in [n]} m_{ib} \\ \text{s.t.} \quad & |g_b - \sum_{j \in [d]} h_{bj} x_{ij}| + m_{ib} \geq \epsilon e_{il}, \forall i \in [n], l \in \mathcal{L}_b. \end{aligned} \quad (2)$$

Here \mathcal{L}_b is the set of leaf nodes that have node b as an ancestor and m_{ib} denotes the slack variable for soft margin. We slightly modify the original 1-norm SVM model by using a small constant ϵe_{il} in (2) instead of e_{il} to prevent variables from getting too big. The constraint is only active when $e_{il} = 1$, namely data i enters the branching node b , and $e_{il} = 1$ implicitly encodes \bar{y}_i .

We use 1-norm, instead of the Euclidean norm, primarily because it can be linearized. The sparsity for \mathbf{h}_b targeted heuristically by including the 1-norm term [27] in the objective allows for feature selection at each branching node. As we noted previously, we can express the term $g_b - \sum_{j \in [d]} h_{bj} x_{ij}$ as the difference of two positive complementary variables, and the absolute value $|g_b - \sum_{j \in [d]} h_{bj} x_{ij}|$ just equals one of these two complementary variables, depending on which branch such a data point enters. When taking the sum over all branching nodes, we have the following *multi-hyperplane SVM* problem to force data points close to the center of the corresponding sub-region at each leaf node $l \in \mathcal{L}$:

$$\begin{aligned} \min \quad & \sum_{b \in A_L(l) \cup A_R(l)} \frac{1}{2} \|\mathbf{h}_b\|_1 + \alpha_1 \sum_{i \in [n], b \in A_L(l) \cup A_R(l)} m_{ib} \\ \text{s.t.} \quad & |g_b - \sum_{j \in [d]} h_{bj} x_{ij}| + m_{ib} \geq \epsilon e_{il}, \\ & \forall i \in [n], b \in A_L(l) \cup A_R(l). \end{aligned} \quad (3)$$

Note that $\bigcup_{l \in \mathcal{L}} (A_L(l) \cup A_R(l)) = \mathcal{B}$. By combining (3) over all leaf nodes $l \in \mathcal{L}$ we obtain:

$$\begin{aligned} \min \quad & \sum_{b \in \mathcal{B}} \frac{1}{2} \|\mathbf{h}_b\|_1 + \alpha_1 \sum_{i \in [n], b \in \mathcal{B}} m_{ib} \\ \text{s.t.} \quad & (1f) - (1j). \end{aligned} \quad (4)$$

Adding the misclassification term $\sum_i c_i$ back into the objective function, and assigning some regularization parameters, we end up getting the desired objective function in (1a).

In LP, the minimized absolute value term $|h_{bj}|$ can be easily formulated as an extra variable h'_{bj} and two linear constraints: $h'_{bj} = h_{bj}^+ + h_{bj}^-$, $h_{bj} = h_{bj}^+ - h_{bj}^-$, $h_{bj}^+, h_{bj}^- \geq 0$. We highlight the major difference here from another recent work [4] in using MIP to train ODT. First, in their formulation (OCT-H), they consider penalizing the number of variables used across all branch nodes in the tree, in order to encourage model simplicity. Namely, their minimized objective is: $\sum_i c_i + \alpha_1 \sum_b \|\mathbf{h}_b\|_0$. However, this requires some additional binary variables for each h_{bj} , which makes the MIP model even harder to solve. In fact, there is empirical evidence that using the 1-norm helps with model sparsity (e.g., LASSO [20], 1-norm SVM [27]). For that reason, we do not bother adding another 0-norm regularization term into our objective. Secondly, unlike in [4], we use variables u_l to denote the assigned label on each leaf, where $u_l \in [Y]$. The next theorem shows, in SVM1-ODT (1), integer \mathbf{u} variables can be equivalently relaxed to be continuous between $[1, Y]$. This relaxation is important in terms of optimization tractability.

Theorem 1. *Every integer variable $u_l, l \in \mathcal{L}$, in (1) can be relaxed to be continuous in $[1, Y]$.*

We note that all proofs are delegated to the supplementary material.

2.2 Strategies to tighten MIP formulation

For MIPs, it is well-known that avoiding the use of a big-M constant can enable us to obtain a tighter linear relaxation of the formulation, which helps an optimization algorithm such as branch-and-bound converge faster to a global solution. The next theorem shows that the big-M constant M can be

fixed to be 1 in (1) for SVM1-ODT, with the idea being to disperse the numerical issues between parameters by re-scaling. The numerical instability for a very small ϵ , as in the theorem below, should be easier to handle by an MIP solver than a very big M .

Theorem 2. Let $(\mathbf{h}'_b, g'_b)_{b \in \mathcal{B}}, (u'_l)_{l \in \mathcal{L}}, (\hat{y}'_i, c'_i)_{i \in [n]}, (w'_{il}, e'_{il})_{i \in [n], l \in \mathcal{L}}, (p'_{ib^+}, p'_{ib^-}, m'_{ib})_{i \in [n], b \in \mathcal{B}}$ be a feasible solution to (1) with parameters $(\alpha_1, \alpha_2, M, \epsilon)$. Then, it is also an optimal solution to (1) with parameters $(\alpha_1, \alpha_2, M, \epsilon)$, if and only if $(\mathbf{h}'_b/M, g'_b/M)_{b \in \mathcal{B}}, (u'_l)_{l \in \mathcal{L}}, (\hat{y}'_i, c'_i)_{i \in [n]}, (w'_{il}, e'_{il})_{i \in [n], l \in \mathcal{L}}, (p'_{ib^+}/M, p'_{ib^-}/M, m'_{ib}/M)_{i \in [n], b \in \mathcal{B}}$ is an optimal solution to (1) with parameters $(M\alpha_1, M\alpha_2, 1, \epsilon/M)$.

Note that $\{(\mathbf{h}'_b, g'_b)_{b \in \mathcal{B}}, (u'_l)_{l \in \mathcal{L}}\}$ and $\{(\mathbf{h}'_b/M, g'_b/M)_{b \in \mathcal{B}}, (u'_l)_{l \in \mathcal{L}}\}$ represent the same decision tree, since $\langle \mathbf{h}'_b, \mathbf{h} \rangle \leq g'_b$ is equivalent to $\langle \mathbf{h}'_b/M, \mathbf{x} \rangle \leq g'_b/M$, at each branch node $b \in \mathcal{B}$.

In MIP parlance, a *cutting-plane* (also called a *cut*) is a linear constraint that is not part of the original model and does not eliminate any feasible integer solutions. Pioneered by Gomory [10, 11], cutting-plane methods, as well as branch-and-cut methods, are among the most successful techniques for solving MIP problems in practice. Numerous types of cutting-planes have been studied in integer programming literature and several of them are incorporated in commercial solvers (see, e.g., [7, 26]). Even though the state-of-the-art MIP solvers can automatically generate cutting-planes during the solving process, these cuts usually do not take the specific structure of the model into consideration. Therefore, most commercial solvers allow the inclusion of user cuts, which are added externally by users in order to further tighten the MIP model. In this section, we propose a series of cuts for SVM1-ODT, and they are added once at the beginning before invoking a MIP solver. For the ease of notation, we denote by \mathcal{N}_k the set of data points with the same dependent variable value k , i.e., $\mathcal{N}_k := \{i \in [n] : y_i = k\}$, for any $k \in [Y]$.

Theorem 3. Given a set $I \subseteq [n]$ with $|I \cap \mathcal{N}_k| \leq 1$ for any $k \in [Y]$. Then for any $L \subseteq \mathcal{L}$, the inequality

$$\sum_{i \in I} c_i \geq \sum_{i \in I, l \in L} e_{il} - |L| \quad (5)$$

is a valid cutting-plane for SVM1-ODT (1).

Here the index set I is composed by arbitrarily picking at most one data point from each class \mathcal{N}_k .

Theorem 3 admits $\Omega(n^Y \cdot 2^{2^D})$ number of cuts. Next, we list cuts that will be added later as user cuts for our numerical experiments. They all help provide lower bounds for the term $\sum_{i \in [n]} c_i$, which also appear in the objective function of our SVM1-ODT formulation. Our cutting planes are added once at the beginning before invoking a MIP solver.

Proposition 1. Let $\{|\mathcal{N}_k| \mid k \in [Y]\} = \{s_1, \dots, s_Y\}$ with $s_1 \leq s_2 \leq \dots \leq s_Y$. Then the following inequalities

1. $\forall l \in \mathcal{L}, \sum_{i \in [n]} c_i \geq \sum_{i \in [n]} e_{il} - s_Y;$
2. $\forall l \in \mathcal{L}, \sum_{i \in [n]} (c_i + e_{il}) \geq s_1 \cdot (Y - 2^D + 1);$
3. $\sum_{i \in [n]} c_i \geq s_1 + \dots + s_{Y-2^D}$ if $Y > 2^D$.

are all valid cutting-planes for SVM1-ODT (1).

Note that the second inequality is only added when $Y \geq 2^D$, and the last lower bound inequality is only added when $Y > 2^D$. As trivial as the last lower bound inequality might seem, in some cases it can be quite helpful. During the MIP solving process, when the current best objective value meets this lower bound value, optimality can be guaranteed and the solver will terminate the branch-and-bound process. Therefore, a tightness of the lower bound has a significant impact on run time.

3 LP-based data-selection procedure

As mentioned previously, our main focus is to be able to train ODT over very large training data-sets. For the purpose of scalability, we rely on a data-selection method prior to the actual training process using SVM1-ODT.

The outline for our procedure is as follows: first, we use a decision tree trained using a heuristic (e.g., CART) as an initial solution. Next, data-selection is performed on clusters represented by the data points with the same dependent values at each leaf node. Finally, we merge all the data subsets selected from each cluster as the new training data, and use SVM1-ODT to train a classification tree on this data-set. In each cluster, our data-selection is motivated by the following simple heuristic: suppose for a data subset I_0 all the points in $\text{conv}\{\mathbf{x}_i \mid i \in I_0\}$ are correctly classified as label y . Then, we can drop out all the data points that lie in the interior of $\text{conv}\{\mathbf{x}_i \mid i \in I_0\}$ from our training set, since by assigning $\{\mathbf{x}_i \mid i \in I_0\}$ to the same leaf node and labeling it with y , we will also correctly classify all the remaining data points inside their convex combination. With that in mind, a data subset I_0 is selected as per the following two criteria: (1) the points within the convex hull $\text{conv}(\{\mathbf{x}_i \mid i \in I_0\})$ are as many as possible; and (2) $|I_0|$ is as small as possible. In each cluster $\mathcal{N} \subseteq [n]$, the following 0-1 LP can be defined to do data-selection:

$$\begin{aligned}
\min \quad & \mathbf{f}^T \mathbf{a} - \mathbf{g}^T \mathbf{b} \\
\text{s.t.} \quad & -\epsilon' \cdot \mathbf{1} \leq b_j \mathbf{x}_j - \sum_{i \neq j} \lambda_{ji} \mathbf{x}_i \leq \epsilon' \cdot \mathbf{1}, \forall j \in \mathcal{N} \\
& \sum_{i \neq j} \lambda_{ji} = b_j, \forall j \in \mathcal{N} \\
& 0 \leq \lambda_{ji} \leq a_i, \forall i \neq j \in \mathcal{N} \\
& a_j + b_j \leq 1, \forall j \in \mathcal{N} \\
& a_j, b_j \in \{0, 1\}, \forall j \in \mathcal{N}.
\end{aligned} \tag{6}$$

Here \mathbf{f}, \mathbf{g} are two parameter vectors with non-negative components. Data point \mathbf{x}_i is selected if $a_i = 1$. Data point \mathbf{x}_j is contained in the convex combination of selected data points if $b_j = 1$. When $\epsilon' = 0$, for any $j \in \mathcal{N}$ with $b_j = 1$, the first two constraints express \mathbf{x}_j as the convex combination of points in $\{\mathbf{x}_i \mid \lambda_{ji} > 0\}$. Here we introduce a small constant ϵ' to allow some perturbation. The third inequality $0 \leq \lambda_{ji} \leq a_i$ means we can only use selected data points, which are those with $a_i = 1$, to express other data points. The last constraint $a_j + b_j \leq 1$ ensures that any selected data point cannot be expressed as a convex combination of other selected data points. Depending on the choice of \mathbf{f}, \mathbf{g} and ϵ' , we have many different variants of (6). In the next section, we describe one variant of data-selection. We discuss *balanced* data-selection in the supplementary material.

3.1 Selecting approximal extreme points

We notice that the original 0-1 LP can be formulated to maximize the number of data points inside the convex hull of selected data points by selecting $\mathbf{f} = \mathbf{0}$ and $\mathbf{g} = \mathbf{1}$. This special case of (6) is used because choosing these values allows us to decompose it into N smaller LPs while maximizing the points inside the convex hull. By projecting out variable \mathbf{a} , the resulting 0-1 LP is equivalent to the following LP, as shown by the next result:

$$\begin{aligned}
\max \quad & \sum_{i \in \mathcal{N}} b_i \\
\text{s.t.} \quad & -\epsilon' \cdot \mathbf{1} \leq b_j \mathbf{x}_j - \sum_{i \neq j} \lambda_{ji} \mathbf{x}_i \leq \epsilon' \cdot \mathbf{1}, \forall j \in \mathcal{N} \\
& \sum_{i \neq j} \lambda_{ji} = b_j, \forall j \in \mathcal{N} \\
& 0 \leq b_j \leq 1, \forall j \in \mathcal{N}.
\end{aligned} \tag{7}$$

We note that such an LP is decomposable: it can be decomposed into $|\mathcal{N}|$ many small LPs, each with $d + 2$ constraints and $|\mathcal{N}|$ variables, and each can be solved in parallel.

Theorem 4. *The following hold.*

- 1) If $\epsilon' = 0$, then for any optimal solution $(\mathbf{b}, \bar{\lambda})$ of (7), there exists λ s.t. (\mathbf{b}, λ) is optimal solution of (6) with $\mathbf{f} = \mathbf{0}, \mathbf{g} = \mathbf{1}$, and vice versa;
- 2) If $\epsilon' > 0$, then for any optimal solution $(\mathbf{b}, \bar{\lambda})$ of (7), there exists λ s.t. $(\lfloor \mathbf{b} \rfloor, \lambda)$ is an optimal solution of (6) with $\mathbf{f} = \mathbf{0}, \mathbf{g} = \mathbf{1}$. Here, $\lfloor \mathbf{b} \rfloor$ is a vector with every component being $\lfloor b_i \rfloor$.

3.2 Data-selection algorithm

For each cluster \mathcal{N} , let $I_{\mathcal{N}} := \{j \in \mathcal{N} : b_j = 1\}$. Note that among those extreme points in $\{i \in \mathcal{N} : \exists j \in I_{\mathcal{N}}, \text{ s.t. } \lambda_{ji} > 0\}$, some of them may be outliers, which will result in λ_{ji} being a small number for all $j \in I_{\mathcal{N}}$. From the classic Carathéodory's theorem in convex analysis, for any point \mathbf{x}_j with $j \in I_{\mathcal{N}}$, it can be written as the convex combination of at most $d + 1$ extreme points. Then in that case, there must exist i with $\lambda_{ji} \geq \frac{1}{d+1}$. Denote $J_{\mathcal{N}} := \{i \in \mathcal{N} : \exists j \in I_{\mathcal{N}}, \text{ s.t. } \lambda_{ji} \geq$

$\frac{1}{d+1}\}$, $K_{\mathcal{N}} := \mathcal{N} \setminus (I_{\mathcal{N}} \cup J_{\mathcal{N}})$. One can show that the size of set $J_{\mathcal{N}}$ is upper bounded by $(d+1)|I_{\mathcal{N}}|$, so if $|I_{\mathcal{N}}|$ is small, $|J_{\mathcal{N}}|$ is also relatively small. For those data points in $K_{\mathcal{N}}$, we use a simple heuristic for data-selection inspired by support vector machines, wherein the most critical points for training are those closest to the classification hyperplane. That is, for some data-selection threshold number N and the boundary hyperplanes $h \in H$ (hyperplanes for each leaf node), we sort data point $i \in K_{\mathcal{N}}$ by $\min_{h \in H} \text{dist}(\mathbf{x}_i, h)$, in an increasing order, and select the first N points. This heuristic is called **hyperplane based data-selection** and is presented in Algorithm 1 below. The LP in (7) above is solved in the second step of Algorithm 1. When $|I_{\mathcal{N}}|$ is relatively large, we can simply select all the extreme points as the new training data. When $|I_{\mathcal{N}}|$ is relatively small, we can select those points with index in $J_{\mathcal{N}}$ as the new training data. The remaining data points in $K_{\mathcal{N}}$ are selected according to the above hyperplane based data-selection heuristic.

Algorithm 1 LP-based data-selection in each cluster \mathcal{N}

Given $\beta_1, \beta_2 \in (0, 1), \beta_2 < (d+1)(1-\beta_1)$;
Solve LP (7) and obtain the optimal solution $(\mathbf{b}, \bar{\lambda})$,
denote $I_{\mathcal{N}} := \{i \in \mathcal{N} : b_i = 1\}, \lambda = T(\bar{\lambda})$,
 $J_{\mathcal{N}} := \{i \in \mathcal{N} : \exists j \in I_{\mathcal{N}}, \text{ s.t. } \lambda_{ji} \geq \frac{1}{d+1}\}$,
 $K_{\mathcal{N}} := \mathcal{N} \setminus (I_{\mathcal{N}} \cup J_{\mathcal{N}})$;
if $|I_{\mathcal{N}}|/|\mathcal{N}| \geq 1 - \beta_1$ **then**
 Select $\mathcal{N} \setminus I_{\mathcal{N}}$ as training set;
else if $|J_{\mathcal{N}}| > \beta_2|\mathcal{N}|$ **then**
 Select $J_{\mathcal{N}}$ as training set;
else
 For $K_{\mathcal{N}}$, do **Hyperplane Data-selection** and pick the first $\beta_2|\mathcal{N}| - |J_{\mathcal{N}}|$ points, together with $J_{\mathcal{N}}$, as the selected new training set.
end if

In Algorithm 1, $T(\bar{\lambda})$ is the transformation we used in the proof of Theorem 4 to construct a feasible solution for (6), and β_1, β_2 are some pre-given threshold parameters. For MIP solvers such as CPLEX, the obtained optimal solution $(\mathbf{b}, \bar{\lambda})$ is a vertex in the associated polyhedron. In that case, $T(\bar{\lambda}) = \bar{\lambda}$.

For large data-sets, for (7), we observed that it took a considerable amount of time to import the constraints into the LP solver and solve it. However, since LP (7) can be decomposed into $|\mathcal{N}|$ number of much smaller LPs, the computational process can be accelerated dramatically.

4 Numerical experiments

We present results mainly from two types of numerical experiments to evaluate the performance of our ODT training procedure: (1) benchmark the mean out-of-sample performance of the ODT trained using SVM1-ODT on medium-sized data-sets ($n \leq 7000$), w.r.t. its counterparts in literature; and (2) benchmark the mean out-of-sample performance of the ODT trained using SVM1-ODT together with our data-selection procedure on large-scale data-sets ($7,000 \leq n \leq 245,000$), w.r.t. CART and OCT-H [4]. For benchmarking, we use data-sets from the UCI Machine Learning Repository [9].

Accuracy of multivariate ODT: We tested the accuracy of the ODT trained using our SVM1-ODT against baseline methods CART and OCT-H [4].

We used the same training-testing split as in [4], which is 75% of the entire data-set as training set, and the rest 25% as testing set, with 5-fold cross-validation. The time limit was set to be 15 or 30 minutes for medium-sized data-sets, and for larger data-sets we increased it up to 4 hours to ensure that the solver could make sufficient progress. Due to intractability of MIP models and the loss in interpretability for deep decision trees we only train ODT for small tree depths, similar to [4, 12, 23]. With the exception of Table 1, all results shown in this section and in the supplementary material are for $D = 2$.

For the following numerical results, our SVM1-ODT formulation is abbreviated as ‘‘S1O’’, ‘‘OCT-H’’ is the MIP formulation to train multivariate ODT in [4], ‘‘Fair’’ is the MIP formulation from [1] without the fairness term in objective, and ‘‘BinOCT’’ is from [23]. We implemented all these MIP approaches in Python 3.6 and solved them using CPLEX 12.9.0 [8]. We invoked the *DecisionTreeClassifier*

Table 1: Performance on data-sets with more than 4 classes using $D = 3$. The numbers inside the bracket ‘()’ for CART and OCT-H are the numerical results reported in [4].

tree depth $D = 3$			
data-set	Dermatology	Heart-disease	Image
n	358	297	210
d	34	13	19
Y	6	5	7
testing accuracy (%)			
SIO	98.9	65.3	85.7
CART	76.1(78.7)	55.6 (54.9)	57.1 (52.5)
OCT-H	82.6 (83.4)	56.2 (54.7)	59.4 (57.4)
Fair	86.4	47.2	63.3
training accuracy (%)			
SIO	100	90.2	100
CART	80.7	68.0	57.1
OCT-H	89.4	81.9	82.7
Fair	100	92.3	100

implementation from scikit-learn to train a decision tree using CART, using default parameter settings. For all methods, the maximum tree depth was set to be the same as our SVM1-ODT.

In Figure (1a), we compare the mean out-of-sample accuracy of the ODT trained from several different MIP formulations. Here the labels on the x-axis represent the names of the data-sets, followed by the data size. In Figure (1b), we have more comparison over the counterparts in literature, along with the BinOCT from [23]. Table 2 shows detailed results about the ODT training, including the running time and training accuracy of different methods. Moreover, we also tested a few data-sets with more than 4 classes using tree depth 3, with time limit being 30 minutes, the results are more exciting: For our tested 3 data-sets, we obtained 20.4 percentage points average improvement over CART, 17.2 percentage points improvement over OCT-H, and 17.7 percentage points improvement over Fair. We list the detailed results in Table 1.

Training multivariate ODT on large data-sets: We use SVM1-ODT together with the LP-based data-selection (denoted by ‘‘S10-DS’’) to train multivariate ODT on large data-sets, which has never been attempted by prior MIP-based training methods in the literature. The time limit for these experiments is 4 hours. Figure 2 depicts the performance of our method w.r.t. CART and OCT-H. For solving S10 and OCT-H, we use the decision tree trained using CART as a warm start solution for CPLEX, as in [4, 24]. For OCT-H, we observe that within the time limit, the solver is either not able to find any new feasible solution other than the one provided as warm start, implying the decision tree trained using OCT-H has the same performance as the one from CART, or the solver simply fails to construct the MIP formulation, which is depicted by the missing bars in the Figure 2. This figure basically means that solely relying on any MIP formulation to train ODT using large data-sets will result in no feasible solution being found within the time limit.

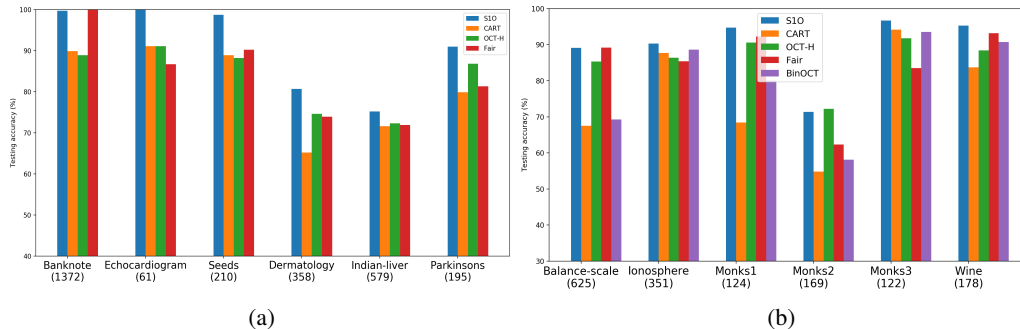


Figure 1: Accuracy comparison for multivariate ODT trained on medium-sized data-sets w.r.t (1a): OCT-H and Fair ; (1b): OCT-H, Fair and BinOCT [23], $D = 2$. In-depth results in Tables 3 and 5 in supplementary material.

Table 2: Accuracy and running time on medium-sized data-sets, for $D=2$. The numbers inside the bracket ‘()’ for CART and OCT-H are the numerical results reported in [4].

data-set	Iris	Congress	Spectf-heart	Breast-cancer	Heart-disease	Image	Hayes-roth
n	150	232	80	683	297	210	132
d	4	16	44	9	13	19	4
Y	3	2	2	2	5	7	3
testing accuracy (%)							
S1O	98.6	98.0	83.3	97.6	69.9	55.0	71.9
CART	92.4 (92.4)	93.5 (98.6)	72.0 (69.0)	91.1 (92.3)	57.5 (54.1)	42.9 (38.9)	55.8 (52.7)
OCT-H	94.4 (95.1)	94.8 (98.6)	75.0 (67.0)	96.1 (97.0)	56.7 (54.7)	49.8 (49.1)	61.2 (61.2)
Fair	90.0	91.4	57.0	95.4	56.7	46.9	72.2
training accuracy (%)							
S1O	98.9	98.5	85	97.3	75.3	56.7	76.8
CART	96.4	96.3	88.0	93.4	58.8	42.9	60.0
OCT-H	99.5	96.2	92.5	95.3	60.5	48.0	77.4
Fair	100	100	100	99.7	68.4	56.6	82.8
running time (s)							
S1O	8.17	1.48	0.42	517.4	900	900	900
OCT-H	1.56	727.3	900	900	900	900	900
Fair	11.14	1.82	0.23	803.5	900	713.7	900

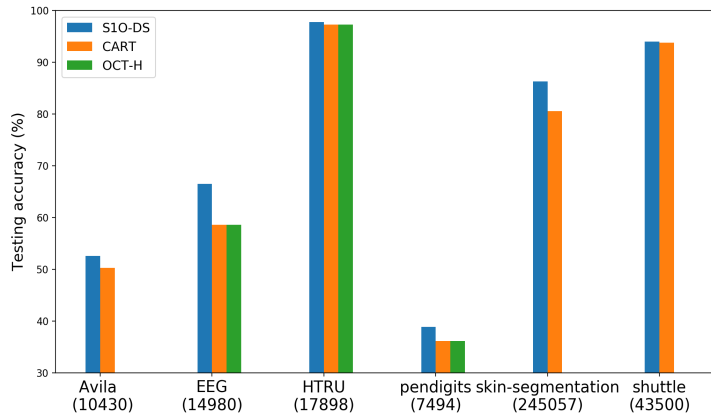


Figure 2: Comparison for large data-sets, $D=2$. In-depth results in Table 7 in supplementary material.

5 Conclusions

We propose a novel MIP-based method to train an optimal multivariate classification tree, which has better generalization behavior compared to state-of-the-art MIP-based methods. Additionally, in order to train ODT on very large data-sets, we devise an LP-based data-selection method. Numerical experiments suggest that the combination of these two can enable us to obtain a decision tree with better out-of-sample accuracy than CART and other comparable MIP-based methods, while solely using any MIP-based training method will fail to do that almost certainly. In the current setup, data selection occurs prior to training using SVM1-ODT. So, once a data subset has been selected, it is used at every branch node to determine optimal branching rules. A natural extension of this methodology could be a combined model for ODT training and data selection, wherein the branching rules learned at the each layer, is applied to the entire data-set and data selection is redone at every node in the subsequent layer prior to branching.

Broader impact

Ensemble methods such as random forests and gradient boosting methods such as XGBoost, and LightGBM typically perform well, in terms of scalability and out-of-sample accuracy, for large-scale classification problems. However, these methods suffer from low interpretability and are incapable of modeling fairness. We hope the scalable MIP-based framework proposed in this paper proves to be seminal in addressing applicability of ODTs to large-scale real-world problems, while relying on the decision tree structure to preserve interpretability. The MIP framework might especially come in handy for sequential or joint prediction-optimization models, wherein the problem structure could be utilized to devise decomposition-based solution procedures. Alternatively, the proposed approach could be used to train a classification tree to provide insights on the behavior of a black-box model.

Funding statement

The first author of this paper was funded by IBM Research during his internship there.

References

- [1] Sina Aghaei, Mohammad Javad Azizi, and Phebe Vayanos. Learning optimal and fair decision trees for non-discriminative decision-making. In *AAAI*, 2019.
- [2] Sina Aghaei, Andres Gomez, and Phebe Vayanos. Learning optimal classification trees: Strong max-flow formulations. *arXiv preprint arXiv:2002.09142*, 2020.
- [3] Kristin P Bennett and JA Blue. A support vector machine approach to decision trees. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, volume 3, pages 2396–2401. IEEE, 1998.
- [4] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.
- [5] Christian Bessiere, Emmanuel Hebrard, and Barry O’Sullivan. Minimising decision tree size as combinatorial optimisation. In *International Conference on Principles and Practice of Constraint Programming*, pages 173–187. Springer, 2009.
- [6] Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. Classification and regression trees. *wadsworth int. Group*, 37(15):237–251, 1984.
- [7] Gérard Cornuéjols. Valid inequalities for mixed integer linear programs. *Mathematical Programming*, 112(1):3–44, 2008.
- [8] IBM ILOG Cplex. V12. 1: User’s manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.
- [9] Dua Dheeru and E Karra Taniskidou. Uci machine learning repository. 2017.
- [10] Ralph E Gomory. Solving linear programming problems in integers. *Combinatorial Analysis*, 10:211–215, 1960.
- [11] Ralph E Gomory. Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem. In *50 Years of Integer Programming 1958-2008*, pages 77–103. Springer, 2010.
- [12] Oktay Günlük, Jayant Kalagnanam, Matt Menickelly, and Katya Scheinberg. Optimal decision trees for categorical data via integer programming. *arXiv preprint arXiv:1612.03225*, 2018.
- [13] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. In *Advances in Neural Information Processing Systems*, pages 7265–7273, 2019.
- [14] Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.

- [15] Garth P McCormick. Computability of global solutions to factorable nonconvex programs: Part I—convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976.
- [16] Mangasarian Olvi and Glenn Fung. Data selection for support vector machine classifiers. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2000.
- [17] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
- [18] J Ross Quinlan. Improved use of continuous attributes in c4. 5. *Journal of artificial intelligence research*, 4:77–90, 1996.
- [19] Jason Rhuggenaath, Yingqian Zhang, Alp Akcay, Uzay Kaymak, and Sicco Verwer. Learning fuzzy decision trees using integer programming. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2018.
- [20] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.
- [21] Alfred Kar Yin Truong. *Fast growing and interpretable oblique trees via logistic regression models*. PhD thesis, Oxford University, UK, 2009.
- [22] Sicco Verwer and Yingqian Zhang. Learning decision trees with flexible constraints and objectives using integer optimization. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 94–103. Springer, 2017.
- [23] Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. In *33rd AAAI Conference on Artificial Intelligence*, 2019.
- [24] Sicco Verwer, Yingqian Zhang, and Qing Chuan Ye. Auction optimization using regression trees and linear models as integer programs. *Artificial Intelligence*, 244:368–395, 2017.
- [25] DC Wickramarachchi, BL Robertson, Marco Reale, CJ Price, and J Brown. Hhcart: An oblique decision tree. *Computational Statistics & Data Analysis*, 96:12–23, 2016.
- [26] Laurence A Wolsey. *Integer programming*, volume 52. John Wiley & Sons, 1998.
- [27] Ji Zhu, Saharon Rosset, Robert Tibshirani, and Trevor J Hastie. 1-norm support vector machines. In *Advances in neural information processing systems*, pages 49–56, 2004.

A Scalable Mixed-Integer Programming Based Framework for Optimal Decision Trees Supplementary Material, NeurIPS 2020

A Supplementary material for Section 2

Theorem 1. *Every integer variable $u_l, l \in \mathcal{L}$ in (1) can be relaxed to be continuous in $[1, Y]$.*

Proof of Theorem 1. It suffices to show, for any extreme point V of $\text{conv}(\bar{P})$, where \bar{P} is the feasible region in (1) while all variables u_l are relaxed to $[1, Y]$, the \mathbf{u} components of V are still integer. We denote $V(u_i)$ to be the u_i component in extreme point V , analogously we have $V(c_i), V(h_{b,j})$ and so on.

Arbitrarily pick $l \in \mathcal{L}$. If for all $i \in [n]$, $V(e_{il}) = 0$, meaning there is no data point entering leaf node l , then we can see that $V(u_l)$ can be any value in $[1, Y]$ and this would maintain the feasibility of point V . Since V is an extreme point, $V(u_l)$ has to be either 1 or Y in this case, which are both integer. Now we assume there exists some $i \in [n]$ such that $V(e_{il}) = 1$, then from (1c)-(1e), we see that $V(u_l) = V(w_{il}) = V(\hat{y}_i)$. From (1b): $(y_i - Y)V(c_i) \leq y_i - V(\hat{y}_i) \leq (y_i - 1)V(c_i)$, here $V(c_i) \in \{0, 1\}$, we get: when $V(c_i) = 0$, then $V(\hat{y}_i) = y_i$, which implies $V(u_l) = y_i$, it is an integer. If $V(c_i) = 1$, then (1b) degenerates to $V(\hat{y}_i) \in [1, Y]$. From the extreme point assumption on V , we know in this case $V(\hat{y}_i)$ should be either 1 or Y , then from $V(u_l) = V(\hat{y}_i)$, we know $V(u_l)$ is still integral. \square

Theorem 2. *Let $(\mathbf{h}'_b, g'_b)_{b \in \mathcal{B}}, (u'_l)_{l \in \mathcal{L}}, (\hat{y}'_i, c'_i)_{i \in [n]}, (w'_{il}, e'_{il})_{i \in [n], l \in \mathcal{L}}, (p'_{ib}, p'_{ib}, m'_{ib})_{i \in [n], b \in \mathcal{B}}$ be a feasible solution to (1) with parameters $(\alpha_1, \alpha_2, M, \epsilon)$. Then it is further an optimal solution to (1) with parameters $(\alpha_1, \alpha_2, M, \epsilon)$, if and only if $(\mathbf{h}'_b/M, g'_b/M)_{b \in \mathcal{B}}, (u'_l)_{l \in \mathcal{L}}, (\hat{y}'_i, c'_i)_{i \in [n]}, (w'_{il}, e'_{il})_{i \in [n], l \in \mathcal{L}}, (p'_{ib}/M, p'_{ib}/M, m'_{ib}/M)_{i \in [n], b \in \mathcal{B}}$ is an optimal solution to (1) with parameters $(M\alpha_1, M\alpha_2, 1, \epsilon/M)$.*

Proof of Theorem 2. Let $(\mathbf{h}'_b, g'_b)_{b \in \mathcal{B}}, (u'_l)_{l \in \mathcal{L}}, (\hat{y}'_i, c'_i)_{i \in [n]}, (w'_{il}, e'_{il})_{i \in [n], l \in \mathcal{L}}, (p'_{ib}, p'_{ib}, m'_{ib})_{i \in [n], b \in \mathcal{B}}$ be an arbitrary feasible solution to (1) with parameters $(\alpha_1, \alpha_2, M, \epsilon)$. So (1b)-(1e), (1k) hold, and

$$\begin{aligned} g'_b - \sum_{j \in [d]} h'_{bj} \cdot x_{ij} &= p'_{ib} - p'_{ib}, \forall i \in [n], b \in \mathcal{B} \\ p'_{ib} &\leq M(1 - e'_{il}), \forall i \in [n], l \in \mathcal{L}, b \in A_R(l) \\ p'_{ib} + m'_{ib} &\geq \epsilon e'_{il}, \forall i \in [n], l \in \mathcal{L}, b \in A_R(l) \\ p'_{ib} &\leq M(1 - e'_{il}), \forall i \in [n], l \in \mathcal{L}, b \in A_L(l) \\ p'_{ib} + m'_{ib} &\geq \epsilon e'_{il}, \forall i \in [n], l \in \mathcal{L}, b \in A_L(l), \end{aligned}$$

which is equivalent to:

$$\begin{aligned} g'_b/M - \sum_{j \in [d]} h'_{bj}/M \cdot x_{ij} &= p'_{ib}/M - p'_{ib}/M, \forall i \in [n], b \in \mathcal{B} \\ p'_{ib}/M &\leq 1 - e'_{il}, \forall i \in [n], l \in \mathcal{L}, b \in A_R(l) \\ p'_{ib}/M + m'_{ib}/M &\geq \epsilon/M e'_{il}, \forall i \in [n], l \in \mathcal{L}, b \in A_R(l) \\ p'_{ib}/M &\leq 1 - e'_{il}, \forall i \in [n], l \in \mathcal{L}, b \in A_L(l) \\ p'_{ib}/M + m'_{ib}/M &\geq \epsilon/M e'_{il}, \forall i \in [n], l \in \mathcal{L}, b \in A_L(l), \end{aligned}$$

Note that constraints (1b)-(1e), (1k) do not involve variables $(h_b, g_b)_{b \in \mathcal{B}}, (p_{ib}^+, p_{ib}^-, m_{ib})_{i \in [n], b \in \mathcal{B}}$. Therefore, we know that solution $(h'_b, g'_b)_{b \in \mathcal{B}}, (u'_l)_{l \in \mathcal{L}}, (\hat{y}'_i, c'_i)_{i \in [n]}, (w'_{il}, e'_{il})_{i \in [n], l \in \mathcal{L}}, (p_{ib}^+, p_{ib}^-, m'_{ib})_{i \in [n], b \in \mathcal{B}}$ is feasible to (1) with parameters (M, ϵ) , if and only if, $(h'_b/M, g'_b/M)_{b \in \mathcal{B}}, (u'_l)_{l \in \mathcal{L}}, (\hat{y}'_i, c'_i)_{i \in [n]}, (w'_{il}, e'_{il})_{i \in [n], l \in \mathcal{L}}, (p_{ib}^+/M, p_{ib}^-/M, m'_{ib}/M)_{i \in [n], b \in \mathcal{B}}$ is feasible to (1) with parameters $(1, \epsilon/M)$.

Furthermore, if $(h'_b, g'_b)_{b \in \mathcal{B}}, \dots$ is the optimal solution to (1) with parameters $(\alpha_1, \alpha_2, M, \epsilon)$, then for any feasible solution $(h''_b, g''_b)_{b \in \mathcal{B}}, \dots$ to (1) with parameters (M, ϵ) , there is

$$\sum_i c'_i + \alpha_1 \sum_{i,b} m'_{ib} + \alpha_2 \sum_{b,j} |h'_{bj}| \leq \sum_i c''_i + \alpha_1 \sum_{i,b} m''_{ib} + \alpha_2 \sum_{b,j} |h''_{bj}|.$$

Since $(h'_b/M, g'_b/M)_{b \in \mathcal{B}}, (u'_l)_{l \in \mathcal{L}}, (\hat{y}'_i, c'_i)_{i \in [n]}, (w'_{il}, e'_{il})_{i \in [n], l \in \mathcal{L}}, (p_{ib}^+/M, p_{ib}^-/M, m'_{ib}/M)_{i \in [n], b \in \mathcal{B}}$ and $(h''_b/M, g''_b/M)_{b \in \mathcal{B}}, (u''_l)_{l \in \mathcal{L}}, (\hat{y}''_i, c''_i)_{i \in [n]}, (w''_{il}, e''_{il})_{i \in [n], l \in \mathcal{L}}, (p_{ib}^+/M, p_{ib}^-/M, m''_{ib}/M)_{i \in [n], b \in \mathcal{B}}$ are both feasible to (1) with parameters $(1, \epsilon/M)$, and the objective function of (1) with parameters $(M\alpha_1, M\alpha_2)$ is $\sum_i c_i + M\alpha_1 \sum_{i,b} m_{ib} + M\alpha_2 \sum_{b,j} |h_{bj}|$, therefore:

$$\begin{aligned} & \sum_i c'_i + M\alpha_1 \sum_{i,b} m'_{ib}/M + M\alpha_2 \sum_{b,j} |h'_{bj}|/M \\ &= \sum_i c'_i + \alpha_1 \sum_{i,b} m'_{ib} + \alpha_2 \sum_{b,j} |h'_{bj}| \\ &\leq \sum_i c''_i + \alpha_1 \sum_{i,b} m''_{ib} + \alpha_2 \sum_{b,j} |h''_{bj}| \\ &= \sum_i c''_i + M\alpha_1 \sum_{i,b} m''_{ib}/M + M\alpha_2 \sum_{b,j} |h''_{bj}|/M. \end{aligned}$$

Hence $(h'_b/M, g'_b/M)_{b \in \mathcal{B}}, (u'_l)_{l \in \mathcal{L}}, (\hat{y}'_i, c'_i)_{i \in [n]}, (w'_{il}, e'_{il})_{i \in [n], l \in \mathcal{L}}, (p_{ib}^+/M, p_{ib}^-/M, m'_{ib}/M)_{i \in [n], b \in \mathcal{B}}$ is an optimal solution to (1) with parameters $(M\alpha_1, M\alpha_2, 1, \epsilon/M)$. The other direction is exactly the same. \square

A.1 Cutting-planes for SVM1-ODT

In this subsection, we provide proofs for Theorem 3 and Proposition 1. Note that we denote by \mathcal{N}_k the set of data points with the same dependent value k , i.e., $\mathcal{N}_k := \{i \in [n] : y_i = k\}$, for any $k \in [Y]$.

Theorem 3. *Given a set $I \subseteq [n]$ with $|I \cap \mathcal{N}_k| \leq 1$ for any $k \in [Y]$. Then for any $L \subseteq \mathcal{L}$, the inequality*

$$\sum_{i \in I} c_i \geq \sum_{i \in I, l \in L} e_{il} - |L|$$

is a valid cutting-plane to SVM1-ODT (1).

Here the index set I is composed by arbitrarily picking at most one data point from each class \mathcal{N}_k .

Proof of Theorem 3. First, we consider the case that $\sum_{l \in L} e_{il} = 1$ for all $i \in I$. This means for any data point $i \in I$, it is classified into leaf node among L . Then inequality (5) reduces to: $\sum_{i \in I} c_i \geq |I| - |L|$. It trivially holds when $|L| \geq |I|$. When $|I| > |L|$, according to the property that $|I \cap \mathcal{N}_k| \leq 1$ for any $k \in [Y]$, we know the data points in I have exactly $|I|$ many different classes. Since all of those data points are entering leaf nodes in L , according to the Pigeon Hole Principle, we know at least $|I| - |L|$ of them are misclassified.

Now we consider general case. We can divide I into two parts: $I = I_1 \cup I_2$, where $I_1 := \{i \in I : \sum_{l \in L} e_{il} = 1\}$, $I_2 := \{i \in I : \sum_{l \in L} e_{il} = 0\}$. Then from our above discussion, we have $\sum_{i \in I_1} c_i \geq \sum_{i \in I_1, l \in L} e_{il} - |L|$. From the definition of I_2 , there is $\sum_{i \in I_2, l \in L} e_{il} = 0$. Hence:

$$\sum_{i \in I} c_i \geq \sum_{i \in I_1} c_i \geq \sum_{i \in I_1, l \in L} e_{il} - |L|$$

$$= \sum_{i \in I_1, l \in L} e_{il} - |L| + \sum_{i \in I_2, l \in L} e_{il} = \sum_{i \in I, l \in L} e_{il} - |L|.$$

We complete the proof. \square

Proposition 1. *Let $\{|\mathcal{N}_k| \mid k \in [Y]\} = \{s_1, \dots, s_k\}$ with $s_1 \leq s_2 \leq \dots \leq s_Y$. Then the following inequalities*

1. $\forall l \in \mathcal{L}, \sum_{i \in [n]} c_i \geq \sum_{i \in [n]} e_{il} - s_Y$;
2. $\forall l \in \mathcal{L}, \sum_{i \in [n]} (c_i + e_{il}) \geq s_1 \cdot (Y - 2^D + 1)$;
3. $\sum_{i \in [n]} c_i \geq s_1 + \dots + s_{Y-2^D}$ if $Y > 2^D$.

are all valid cutting-planes to SVM1-ODT (1).

Proof of Proposition 1. We prove the validity of these inequalities individually.

1. Since $s_1 \leq \dots \leq s_Y$, we can partition the set $[n]$ into s_Y different disjoint I s that all have property $|I \cap \mathcal{N}_k| \leq 1$ for any $k \in [Y]$ as in Theorem 3. Select $\mathcal{L} = \{l\}$ for an arbitrary $l \in \mathcal{L}$. Then, for each of the above set I we obtain a cut from (5). Combining all these S_Y cuts together, we would obtain the desired inequality $\sum_{i \in [n]} c_i \geq \sum_{i \in [n]} e_{il} - s_Y$.
2. For any $l \in \mathcal{L}$, denote $L := \mathcal{L} \setminus \{l\}$. Then for index set I with $|I \cap \mathcal{N}_k| \leq 1$ for any $k \in [Y]$, Theorem 3 gives us the inequality $\sum_{i \in I} c_i \geq \sum_{i \in I, l' \in L} e_{il'} - |L|$. From (1k) of SVM1-ODT, there is $\sum_{l' \in L} e_{il'} = 1 - e_{il}$. Hence we obtain the inequality

$$\sum_{i \in I} (c_i + e_{il}) \geq |I| - 2^D + 1.$$

Then we construct s_1 number of disjoint I s each with cardinality $|Y|$, by arbitrarily picking one element from each class \mathcal{N}_k without replacement. This process can be done for exactly s_1 many times. Therefore, we obtain s_1 many inequalities for different index set I :

$$\sum_{i \in I} (c_i + e_{il}) \geq |Y| - 2^D + 1.$$

For those $i \in [n]$ which are not contained in any of these s_1 many I s, we use the trivial lower bound inequality $c_i + e_{il} \geq 0$. Lastly, we combine all those inequalities together, which leads to:

$$\sum_{i \in [n]} (c_i + e_{il}) \geq s_1 \cdot (Y - 2^D + 1).$$

3. When $Y > 2^D$, since there are more classes than the number of leaf nodes, and all data points have to be categorized into exactly one of those leaf nodes, by the Pigeonhole Principle, we know the number of misclassification is at least $s_1 + \dots + s_{Y-2^D}$.

\square

A.2 SVM1-ODT for data-set with Categorical Features

Usually in practice, when dealing with data-set with both numerical and categorical features, people will do some feature transformation in pre-processing, e.g., one-hot encoding etc. It is doing fine for most heuristic-based training methods, but since here our training method is MIP-based, the tractability performance can be very dimension-sensitive. Most feature transformation increases the data dimension greatly. In this subsection we are going to talk about the modifications of

the formulation SVM1-ODT, in order to train the initial data-set directly without doing feature transformation. First we add the following new constraints:

$$\sum_{j \in \mathcal{F}_c} h_{bj} \leq 1, \forall b \in \mathcal{B} \quad (8a)$$

$$s_{bjv} \leq h_{bj}, \forall b \in \mathcal{B}, j \in \mathcal{F}_c, v \in \mathcal{V}_j \quad (8b)$$

$$\sum_{j \in \mathcal{F}_c} \sum_{v \in \mathcal{V}_j} s_{bjv} \mathbb{I}(x_{i,j} = v) \geq e_{il} + \sum_{j \in \mathcal{F}_c} h_{bj} - 1, \quad (8c)$$

$$\forall i \in [n], l \in \mathcal{L}, b \in A_L(l)$$

$$\sum_{j \in \mathcal{F}_c} \sum_{v \in \mathcal{V}_j} s_{bjv} \mathbb{I}(x_{i,j} = v) \leq 2 - e_{il} - \sum_{j \in \mathcal{F}_c} h_{bj}, \quad (8d)$$

$$\forall i \in [n], l \in \mathcal{L}, b \in A_R(l)$$

$$\sum_{j \in \mathcal{F}_c} h_{bj} - 1 \leq h_{bj} \leq 1 - \sum_{j \in \mathcal{F}_c} h_{bj}, \quad (8e)$$

$$\forall b \in \mathcal{B}, j \in \mathcal{F}_q$$

$$s_{bjv}, h_{bj} \in \{0, 1\}, \forall b \in \mathcal{B}, j \in \mathcal{F}_c, v \in \mathcal{V}_j \quad (8f)$$

Here $\mathcal{F}_c \subseteq [d]$ denotes the index set of categorical features, and $\mathcal{F}_q = [d] \setminus \mathcal{F}_c$ denotes the index set of numerical features. $\mathbb{I}(x_{i,j} = v)$ is the indicator function about whether $x_{i,j} = v$ or not. For $j \in \mathcal{F}_c$, \mathcal{V}_j denotes the set of all possible values at categorical feature j .

For our above formulation, we are assuming: At each branch node of the decision tree, the branching rule cannot be based on both categorical features and numerical features, and if it is based on categorical feature, then the branching rule is only provided by one single categorical feature. These constraints are formulated by (8a) and (8e) individually, where $h_{bj} \in \{0, 1\}$ for $j \in \mathcal{F}_c$ characterizes whether the branching rule at node b is based on categorical feature j or not. Furthermore, we use another binary variable s_{bjv} to denote the conditions that data points must satisfy in order to go left at that node b : If the j -th component of data point \mathbf{x}_i has value v such that $s_{bjv} = 1$, then this data point goes to the left branch. Here we allow multiple different values $v \in \mathcal{V}_j$ such that $s_{bjv} = 1$. Clearly s_{bjv} can be positive only when categorical feature j is chosen as the branching feature. So we have constraint (8b). Note that this branching rule for categorical features have also been considered in [1], through different formulation. Now we explain how to formulate this categorical branching rule into linear constraints. Still, we start from the leaf node. When $e_{il} = 1$, we know the branching rule at each $b \in A_L(l) \cup A_R(l)$ is applied to data point \mathbf{x}_i . To be specific, when $b \in A_L(l)$ and $\sum_{j \in \mathcal{F}_c} h_{bj} = 1$, there must exists some feature $j \in \mathcal{F}_c$ and $v \in \mathcal{V}_j$ such that $\mathbf{x}_{i,j} = v$ and $s_{bjv} = 1$. Analogously, when $b \in A_R(l)$ and $\sum_{j \in \mathcal{F}_c} h_{bj} = 1$, the previous situation cannot happen. Then one can easily see that these two implications are formulated by (8c) and (8d). Cases of $\sum_{j \in \mathcal{F}_c} h_{bj} = 0$ are not discussed here since in that case the branching rule is given by the branching hyperplane on numerical features, which was discusses in Sect. 2.

Furthermore we should mention that, to train data-set with both numerical and categorical features, we will also modify some constraints in the original formulation (1): (1h) and (1j) should be changed to $p_{ib}^- + m_{ib} \geq \epsilon(e_{il} - \sum_{j \in \mathcal{F}_c} h_{bj})$ and $p_{ib}^+ + m_{ib} \geq \epsilon(e_{il} - \sum_{j \in \mathcal{F}_c} h_{bj})$. This is because those two constraints are only considered when the branching rule is not given by categorical features, which means $\sum_{j \in \mathcal{F}_c} h_{bj} = 0$.

B Supplementary material for Section 3

Theorem 4.

- 1) If $\epsilon' = 0$, then for any optimal solution $(\mathbf{b}, \bar{\lambda})$ of (7), there exists λ s.t. (\mathbf{b}, λ) is optimal solution of (6) with $\mathbf{f} = \mathbf{0}, \mathbf{g} = \mathbf{1}$, and vice versa;
- 2) If $\epsilon' > 0$, then for any optimal solution $(\mathbf{b}, \bar{\lambda})$ of (7), there exists λ s.t. $(\lfloor \mathbf{b} \rfloor, \lambda)$ is an optimal solution of (6) with $\mathbf{f} = \mathbf{0}, \mathbf{g} = \mathbf{1}$. Here, $\lfloor \mathbf{b} \rfloor$ is a vector with every component being $\lfloor b_i \rfloor$.

Proof of Theorem 4. We consider two cases for ϵ' .

Case 1 ($\epsilon' = 0$): First we show: If $(\mathbf{b}, \bar{\lambda})$ is an optimal solution in (7), then there exists λ such that (\mathbf{b}, λ) is optimal in (6). Notice that when $\epsilon' = 0$, $\mathbf{b} \in \{0, 1\}^{|\mathcal{N}|}$. This is because if $b_j \in (0, 1)$, we can multiply b_j and all $\lambda_{j,i}$ by $\frac{1}{b_j}$, the new solution is still feasible, with higher objective value, which contradicts to the optimal assumption of \mathbf{b} . Assuming there exists some $i \in \mathcal{N}$ such that $b_i = 1$ and exists some $j \neq i \in \mathcal{N}$ such that $\bar{\lambda}_{j,i} > 0$. Since $b_j = \sum_i \bar{\lambda}_{j,i}$, we know $b_j = 1$. Now we transform the original j -th row of $\bar{\lambda}$ in this way: replace $\bar{\lambda}_{j,i}$ by 0, replace $\bar{\lambda}_{j,i'}$ by $\frac{\bar{\lambda}_{j,i}\bar{\lambda}_{i'i'} + \bar{\lambda}_{j,i'}}{1 - \bar{\lambda}_{j,i}\bar{\lambda}_{i'j}}$. Here $i' \neq i, j$. As long as there exists $i \neq j \in \mathcal{N}$ such that $b_i = 1, \lambda_{j,i} > 0$, we can do the above transformation to get rid of it. Therefore after doing this transformation for finitely many times, we obtain a new λ with $0 \leq \lambda_{j,i} \leq 1 - b_i, \forall i \neq j \in \mathcal{N}$. It remains to show, after each transformation, the first two equations in (7) still hold. The first equation holds because $\mathbf{x}_j = \bar{\lambda}_{j,i}\mathbf{x}_i + \sum_{i' \neq j, i} \bar{\lambda}_{j,i'}\mathbf{x}_{i'} = \bar{\lambda}_{j,i}(\sum_{i' \neq i} \bar{\lambda}_{i'i'}\mathbf{x}_{i'}) + \sum_{i' \neq j, i} \bar{\lambda}_{j,i'}\mathbf{x}_{i'} = \bar{\lambda}_{j,i}\lambda_{ij}\mathbf{x}_j + \sum_{i' \neq j, i} (\bar{\lambda}_{j,i}\bar{\lambda}_{i'i'} + \bar{\lambda}_{j,i'})\mathbf{x}_{i'}$. Hence $\mathbf{x}_j = \sum_{i' \neq j, i} \frac{\bar{\lambda}_{j,i}\bar{\lambda}_{i'i'} + \bar{\lambda}_{j,i'}}{1 - \bar{\lambda}_{j,i}\bar{\lambda}_{i'j}}\mathbf{x}_{i'}$. For the second equation we just need to check $1 = \sum_{i' \neq j, i} \frac{\bar{\lambda}_{j,i}\bar{\lambda}_{i'i'} + \bar{\lambda}_{j,i'}}{1 - \bar{\lambda}_{j,i}\bar{\lambda}_{i'j}}$. This is true because $\sum_{i' \neq j, i} (\bar{\lambda}_{j,i}\bar{\lambda}_{i'i'} + \bar{\lambda}_{j,i'}) / (1 - \bar{\lambda}_{j,i}\bar{\lambda}_{i'j}) = (\sum_{i' \neq j, i} (\bar{\lambda}_{j,i}\bar{\lambda}_{i'i'} + \bar{\lambda}_{j,i'}))(1 - \bar{\lambda}_{j,i}\bar{\lambda}_{i'j}) = (\bar{\lambda}_{j,i}(\sum_{i' \neq j, i} \bar{\lambda}_{i'i'}) + \sum_{i' \neq j, i} \bar{\lambda}_{j,i'})(1 - \bar{\lambda}_{j,i}\bar{\lambda}_{i'j}) = (\bar{\lambda}_{j,i}(1 - \bar{\lambda}_{i'j}) + (1 - \bar{\lambda}_{j,i}))(1 - \bar{\lambda}_{j,i}\bar{\lambda}_{i'j}) = 1$. Therefore, we have shown for any optimal solution $(\mathbf{b}, \bar{\lambda})$ of (7), there exists λ s.t. (\mathbf{b}, λ) is feasible in (6). The optimality of such (\mathbf{b}, λ) in (6) automatically follows from the fact that (7) is a relaxation over (6). Now, we show the other direction. It suffices to show: If (\mathbf{b}, λ) is an optimal solution of (6), then (\mathbf{b}, λ) is also an optimal solution of (7). Clearly (\mathbf{b}, λ) is feasible in (7). If it is not the optimal solution, meaning there exists $(\mathbf{b}', \bar{\lambda})$ in (7), such that $\mathbf{1}^T \mathbf{b}' > \mathbf{1}^T \mathbf{b}$. From what we just showed, there exists λ' such that (\mathbf{b}', λ') is feasible in (6), with $\mathbf{1}^T \mathbf{b}' > \mathbf{1}^T \mathbf{b}$, which gives the contradiction.

Case 2 ($\epsilon' > 0$): Since $\lfloor \mathbf{b}^* \rfloor$ is a binary vector, follow the same argument as in above, we can construct λ such that $(\lfloor \mathbf{b}^* \rfloor, \lambda)$ is a feasible solution to (6). To show it is also optimal, it suffices to show, any feasible solution in (6), has objective value no larger than $\sum_{i \in \mathcal{N}} \lfloor b_i^* \rfloor$. Assume $(b_i)_{i \in \mathcal{N}}$ is part of one feasible solution in (6). Then apparently it is also part of one feasible solution in (7), since (7) is simply the relaxation of (6). Now we show $b_i \leq b_i^*$ for all i : when $b_i = 0$ it is trivially true, when $b_i = 1$ then b_i^* should also be 1, since otherwise we can also increase b_i^* to 1 while remain feasible, and this contradicts to the optimality assumption of \mathbf{b}^* . Hence: $\sum_i \lfloor b_i \rfloor \leq \sum_i \lfloor b_i^* \rfloor$. Since \mathbf{b} is binary, then $\sum_i \lfloor b_i \rfloor = \sum_i b_i$, and this concludes the proof. \square

B.1 Balanced Data-selection

In this section we talk about another variant of (7), under the same data-selection framework as we mentioned in Sect. 3, called *balanced data-selection*. Sometimes when doing data-selection, it might not be a good idea to pick all the extreme points or discard as many points inside the convex hull as possible. One example is Figure 3: In this data-set, most of data points lie inside the rectangle, some of them lie on the circle, and a few of them lie inside the circle while outside the rectangle. If we only care about maximizing the number of points inside the convex hull of entire data points, then we would end up picking all points on the circle, however in this case it is obvious that picking those 4 vertices of the rectangle is a better option since they capture the data distribution better, even though these 4 points are not extreme points of the entire data-set.

Motivated by the above example, we realize that in many cases the balance between the number of selected points (those with $a_i = 1$ in (6)) and the number of points inside the convex hull of selected points (points with $b_i = 1$) is important. This can be represented by choosing nonzero objective \mathbf{f} and \mathbf{g} . One obvious choice here is $\mathbf{f} = \mathbf{g} = \mathbf{1}$. However, once we pick nonzero objective \mathbf{f} , we cannot simply project out \mathbf{a} variables by replacing $a_i = 1 - b_i$ any more, and 0-1 LP (6) might also seem intractable to solve optimally in large data-set case. But since this 0-1 LP was proposed by heuristic, we do not have to solve it to optimality. Here we can approximately solve it by looking at the optimal solution of its corresponding linear

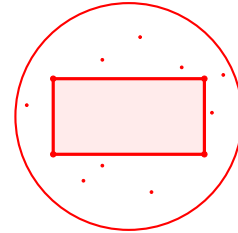


Figure 3: In this case, the balanced data-selection is more preferable.

relaxation:

$$\begin{aligned}
& \max \quad \sum_{i \in \mathcal{N}} (b_i - a_i) \\
& \text{s.t.} \quad b_j \mathbf{x}_j = \sum_{i \in \mathcal{N}, i \neq j} \lambda_{ji} \mathbf{x}_i, \forall j \in \mathcal{N} \\
& \quad \sum_{i \in \mathcal{N}, i \neq j} \lambda_{ji} = b_j, \forall j \in \mathcal{N} \\
& \quad 0 \leq \lambda_{ji} \leq a_i, \forall i \neq j \in \mathcal{N} \\
& \quad a_j + b_j \leq 1, \forall j \in \mathcal{N} \\
& \quad a_j, b_j \in [0, 1].
\end{aligned} \tag{9}$$

(9) is the natural linear relaxation for (6) with $\epsilon' = 0$. Then we have the following observations:

Remark 1. Assume $(\mathbf{a}', \mathbf{b}', \lambda')$ is the optimal solution of (9). Then:

- If $b'_i = 0$ for some $i \in \mathcal{N}$, then \mathbf{x}_i cannot be written as the convex combination of some other points;
- If $a'_i = 0$ for some $i \in \mathcal{N}$, then $b'_i \in \{0, 1\}$;
- If $a'_i > 0, b'_i > 0$ for some $i \in \mathcal{N}$, then $a'_i + b'_i = 1$.

Hence we know that, by enforcing $b_i = 0$ in (6) for those i with $b'_i = 0$, it will not change the optimal solution; For those points \mathbf{x}_i with $b'_i = 1$ and \mathbf{x}_j with $b'_j \in (0, 1)$, we expect that it is easier to express \mathbf{x}_i as the convex combination of other points than \mathbf{x}_j , so we greedily assign $b_i = 1, a_i = 0$ in (6). The tricky part is about those point with $a'_i > 0, b'_i > 0$, which is used to express other points and can also be expressed by other points in the optimal solution of (9). So we implement the original 0-1 LP (6) to handle those ‘‘ambiguous’’ points. In other words, we will do the following steps:

Algorithm 2 Approximately get a balanced data-selection solution

- Solve** the LP relaxation (9), and denote $(\mathbf{a}', \mathbf{b}', \lambda)$ to be the optimal solution, $I_0 := \{i \in \mathcal{N} : b'_i = 0\}, I_1 := \{i \in \mathcal{N} : b'_i = 1\}$;
- Assign** $b_i = 0$ for $i \in I_0, b_i = 1$ for $i \in I_1$ in (6) with $\mathbf{f} = \mathbf{g} = \mathbf{1}$, and solve it;
- Pick** the index support of the \mathbf{a} components in the optimal solution as the selected data subset.
-

B.2 Iterative ODT training algorithm

Provided with the framework of using MIP to train ODT and from the nature of our data-selection method, in this subsection we want to propose a generic iterative method to continuously obtain more accurate ODTs from subsets of data-set. We are introducing this iterative training method corresponding to (7), where $\mathbf{f} = \mathbf{0}, \mathbf{g} = \mathbf{1}$. Note that the balanced data-selection can also be applied to this iterative method. We should also mention that this iterative method shares the same spirit as the classic E/M algorithm in statistics.

Algorithm 3 Iterative ODT training method based on data-selection

- Initialize** data-selection parameter ϵ' , SVM1-ODT training parameter $\alpha_1, \alpha_2, \epsilon, M$, and initial heuristically generated decision tree T ;
- for** $t = 0, 1, 2, \dots$ **do**
- Cluster \mathcal{N} is picked as: the data points of $[n]$ assigned into each leaf node of T , which are also been **correctly classified**. Denote the collection of **incorrectly classified data** to be I ;
- Solve** the data-selection sub-problem (7) for each cluster \mathcal{N} . Denote $\bar{I}_{\mathcal{N}} := \{j \in \mathcal{N} : b_j = 1\}, J_{\mathcal{N}} := \{i \notin \bar{I}_{\mathcal{N}} : \exists j \in \bar{I}_{\mathcal{N}}, \text{ s.t. } \lambda_{ji} > 0\}, K_{\mathcal{N}} := \mathcal{N} \setminus (\bar{I}_{\mathcal{N}} \cup J_{\mathcal{N}})$. Then denote J to be the collection of all $J_{\mathcal{N}}$, and K to be the collection of all $K_{\mathcal{N}}$;
- Input** the data subset $I \cup J \cup K$ into MIP (1), and replacing the term $\sum_{i \in I \cup J \cup K} c_i$ in objective function by: $\sum_{j \in J} (|I| + 1)c_j + \sum_{i \in I \cup K} c_i$, and solve the SVM1-ODT;
- Assign** T to be decision tree corresponding to the optimal (best current feasible) solution of the previous SVM1-ODT, and iterate the loop.
- end for**
-

For this iterative method, we have the following statement.

Proposition 2. When $\alpha_1 = \alpha_2 = 0$, and the SVM1-ODT in each iteration of Algorithm 3 is solved to optimality, then the decision tree obtained at each iteration would have higher accuracy over the entire training data than the previous decision trees.

Proof. For decision tree T_0 , denote I to be the collection of incorrectly classified data in T_0 . After solving the data-selection sub-problem (7) for each cluster, we denote \bar{I} to be the collection of data point i in each cluster which has $b_i = 1$, and J, K as defined in the Algorithm 3. Clearly $[n] = I \cup \bar{I} \cup J \cup K$, and the accuracy of T_0 is $1 - \frac{|I|}{n}$. Now we assume T_1 to be the tree obtained at the next iteration, which minimizes the objective $\sum_{j \in J} (|I| + 1)c_j + \sum_{i \in I \cup K} c_i$. Since T_0 is a feasible solution, with objective value $|I|$, then we must have, for decision tree T_1 , $c_j = 0$ for all $j \in J$, and $\sum_{i \in I \cup K} c_i \leq |I|$. In other words, tree T_1 correctly classifies all data points in J , and the misclassification number over $I \cup K$ is at most $|I|$ many. According to the construction of J and \bar{I} , we know that once all data points in J are correctly classified, then all data points in \bar{I} are also correctly classified, since data point in \bar{I} is contained in the convex hull of points in J . Also because $[n] = I \cup \bar{I} \cup J \cup K$, we know the training accuracy of T_1 over $[n]$ is just $1 - \frac{\sum_{i \in I \cup K} c_i}{n}$, which is at least $1 - \frac{|I|}{n}$, the accuracy of T_0 . \square

C Additional numerical results and detailed records

Table 3: Accuracy and running time on medium-sized data-sets, for tree depth D=2. The numbers after ‘/’ for CART and OCT-H are the numerical results reported in [4].

data-set	Banknote- authen	Echocar- diogram	Seeds	Dermat- ology	Indian- liver	Parkinsons
n	1372	61	210	358	579	195
d	4	10	7	34	10	21
Y	2	2	3	6	2	2
testing accuracy (%)						
S1O	99.7	100	98.7	80.7	75.2	91.0
CART	89.9 /89.0	91.1 /74.7	88.9 /87.2	65.2 /65.4	71.6 /71.7	79.9 /84.1
OCT-H	88.9 /91.5	91.1 /77.3	88.2 /90.6	74.6 /74.2	72.3 /72.6	86.8 /84.9
Fair	100	86.7	90.2	73.9	71.9	81.3
training accuracy (%)						
S1O	100	100	100	81.1	79.6	100
CART	91.7	100	92.5	67.0	71.4	88.2
OCT-H	87.2	100	94.2	75.9	75.4	92.0
Fair	100	100	100	81.1	81.3	100
running time (s)						
S1O	900	5.1	39.3	900	900	900
OCT-H	900	0.09	519	900	900	900
Fair	94.8	0.28	55	231	600	19.6

Table 4: Accuracy and running time on medium-sized data-sets, for tree depth $D=2$. The numbers after ‘/’ for CART and OCT-H are the numerical results reported in [4].

data-set	sonar	survival	Hepatitis	Relax
n	208	306	80	182
d	60	3	19	12
Y	2	2	2	2
testing accuracy (%)				
S1O	82.4	75.1	89.5	73.6
CART	69.3 /70.4	73.4 /73.2	80.1 /83.0	68.2 /71.1
OCT-H	74.6 /70.0	73.3 /73.0	84.2 /81.0	66.0 /70.7
Fair	70.6	74.1	84.2	73.3
training accuracy (%)				
S1O	100	74.2	100	74.9
CART	79.4	77.1	94.5	74.7
OCT-H	89.0	74.7	99.1	77.5
Fair	100	80.0	100	90.5
running time (s)				
S1O	900	900	44	900
OCT-H	900	900	107	900
Fair	3.14	900	0.3	900

Table 5: Testing results for multivariate ODT on medium-sized data-sets, $D=2$. The numbers after ‘/’ for CART and OCT-H are the numerical results reported in [4].

data-set	Balance-scale	Ionosphere	Monks-1	Monks-2	Monks-3	Wine
n	625	351	124	169	122	178
d	4	34	6	6	6	13
Y	3	2	2	2	2	3
testing accuracy (%)						
S1O	89.1	90.3	94.7	71.4	96.7	95.3
CART	67.5 /64.5	87.7 /87.8	68.4 /57.4	54.8 /60.9	94.2 /94.2	83.7 /81.3
BinOCT	69.3	88.6	80.0	58.1	93.5	90.7
OCT-H	85.3 /87.6	86.4 /86.2	90.6 /93.5	72.2 /75.8	91.8 /92.3	88.4 /91.1
Fair	89.2	85.4	92.3	62.3	83.5	93.2
training accuracy (%)						
S1O	86.5	100	98.1	87.7	92.4	100
CART	71.7	91.0	76.8	65.2	93.5	94.1
BinOCT	73.3	91.1	83.5	69.8	93.8	97.3
OCT-H	82.9	94.2	94.9	79.2	90.0	97.0
Fair	89.8	100	100	96.2	97.4	100

Table 6: More testing results for medium-sized data-sets, $D=2$. Here “S1O-DS” refers to the combination of SVM1-ODT and LP-based data-selection method.

data-set	statlog	spambase	Thyroid- disease-ann- thyroid	Wall-following- robot-2	seismic
n	4435	4601	3772	5456	2584
d	36	57	21	2	20
Y	6	2	3	4	2
testing accuracy (%)					
S1O-DS	74.2	89.1	96.3	93.7	93.3
CART	63.7 /63.2	84.3 /84.2	97.5 /95.6	93.7/94.0	93.3 /93.3
OCT-H	63.7 /63.2	87.2 /85.7	92.8 /92.5	93.7/94.0	93.3 /93.3
Fair	63.7	86.1	93.0	93.7	93.3
training accuracy (%)					
S1O-DS	74.1	88.9	96.7	93.7	93.3
CART	63.5	86.7	98.4	93.7	93.3
OCT-H	63.5	90.3	94.2	93.7	93.3
Fair	63.5	91.4	95.8	93.7	93.3
running time (s)					
S1O-DS	900	900	900	2.08	0.16
percentage of selected data points (%)					
DS	7.0	10.0	12	1.0	10.0
Parameters set					
ϵ	0.1	0.02	0.01	0.0008	0.01
α_1	1000000	1000	1000	30	1000
α_2	0.1	0.1	0.1	0.01	0.1
ϵ'	0.0	0.0	0.0	0.0	0.0
β_1	0.3	0.3	0.1	0.4	0.4
β_2	0.07	0.1	0.12	1.0	0.1

Table 7: Testing results for large-scale data-sets, with tree depth $D = 2$, time limit is set to be $4h$.

data-set	Avila	EEG	HTRU	pendigits	skin-segmentation	shuttle
n	10430	14980	17898	7494	245057	43500
d	10	14	8	16	3	9
Y	12	2	2	10	2	7
testing accuracy (%)						
S1O-DS	52.6	66.5	97.8	38.9	86.3	94.0
CART	50.3	58.6	97.3	36.2	80.6	93.8
OCT-H	N/A	58.6	97.3	36.2	N/A	N/A
training accuracy (%)						
S1O-DS	55.0	67.1	97.5	38.9	87.1	94.1
CART	50.7	60.3	97.8	36.5	81.0	94.0
OCT-H	N/A	60.3	97.8	36.5	N/A	N/A
percentage of selected data points (%)						
DS	4.5	2.0	3	7.1	0.42	2.0
Parameters set						
ϵ	0.01	0.04	0.5	0.01	0.03	0.007
α_1	1000	1000	1000	1000	1000	1000
α_2	0.1	0.1	0.1	0.1	0.1	0.1
ϵ'	0.1	0.0	0.0	0.3	0.0	0.0
β_1	0.2	0.1	0.1	0.1	0.2	0.1
β_2	0.05	0.02	0.03	0.1	0.05	0.02

Table 8: Testing results for large-scale data-sets, with tree depth $D = 3$, time limit is set to be $4h$.

data-set	Avila	EEG	HTRU	pendigits	skin-segmentation	shuttle
n	10430	14980	17898	7494	245057	43500
d	10	14	8	16	3	9
Y	12	2	2	10	2	7
testing accuracy (%)						
S1O-DS	55.8	66.5	97.9	62.5	94.9	99.5
CART	53.5	64.2	98.1	57.9	87.1	99.7
OCT-H	N/A	N/A	N/A	57.9	N/A	N/A
training accuracy (%)						
S1O-DS	58.1	70.6	97.4	62.2	94.7	99.5
CART	56.2	65.3	97.8	58.4	87.6	99.7
OCT-H	N/A	N/A	N/A	58.4	N/A	N/A
percentage of selected data points (%)						
DS	2.0	2.0	1.8	2.8	0.39	1.5
Parameters set						
ϵ	0.02	0.02	0.02	0.01	0.03	0.008
α_1	1000	1000	100000	1000	1000	1000
α_2	0.1	0.1	0.01	0.1	0.1	0.1
ϵ'	0.0	0.0	0.1	0.3	0.0	0.0
β_1	0.2	0.1	0.2	0.1	0.2	0.1
β_2	0.02	0.02	0.04	0.08	0.03	0.015