New optimization models for optimal classification trees

Zacharie Alès, Valentine Huré, Amélie Lambert

New optimization models for optimal classification trees

Zacharie Ales, Valentine Huré, Amélie Lambert

November 22, 2022

Abstract

Interpretability is a growing concept in Machine Learning. Decision-making algorithms are more and more used in healthcare, finance or other high stakes contexts. Therefore, the need for algorithms whose decisions are understandable is of the utmost importance. Intrinsically interpretable classifiers such as decision trees are often seen as less accurate than black box models such as neural networks. For decision trees, state-of-the-art methods are recursive heuristics (e.g. CART) that may fail to find underlying characteristics in datasets. Recently, linear formulations were introduced to model the problem of the construction of the best decision tree for a given dataset. Notably, a MIO formulation, introduced by Bertsimas and al., has shown better accuracy than CART. However this model does not scale up to datasets with more than 1000 data points. Our work focuses on improvements of MIOs that speed up their resolution in order to handle larger problems. We present a quadratic formulation of the MIP devised by Bertsimas and al. as well as its linearization and another that extends a flow-formulation (from binary dataset to real-value dataset). We prove that our new formulations have stronger continuous relaxation than the MIP introduced by Bertsimas and al.. Finally, our experiments show that they have a significantly smaller resolution time than the MIP of Bertsimas and al. while maintaining or improving performances on test sets.

Keywords: combinatorial optimization, optimal classification trees, mixed binary programming, quadratic programming, linearizations

1 Introduction

A supervised classification problem considers a dataset $(X_i, y_i)_{i\in\overline{\mathcal{I}}}$ such that $X_i = (x_{i,j})_{j\in\mathcal{J}} \in \mathbb{R}^{|\mathcal{J}|}$ is the features vector of data $i \in \overline{\mathcal{I}}$ and $y_i \in \mathcal{K}$ is its associated class. The objective is then to determine a classification function $\mathcal{C} : \mathbb{R}^{|\mathcal{J}|} \mapsto \mathcal{K}$, called a *classifier* to best predict the class of new data from their features vectors. To that end, the data are partitioned into two subsets: the *training set* $\mathcal{I}$ used to train the classifier and the *test set* $\mathcal{I}_t$ used for the evaluation. The evaluation consists in computing the percentage of exact predictions over $\mathcal{I}_t$: $\frac{|\{i\in\mathcal{I}_t \mid \mathcal{C}(X_i)=y_i\}|}{|\mathcal{I}_t|}$.

The work presented in this article aims to provide classifiers which both have good performances on the training set and are *interpretable*. Interpretability is a concept that is increasingly considered in supervised classification (Carvalho, Pereira, & Cardoso, 2019). Although several definitions of this concept can be considered, it can be summarised as the ability to explain or to present in understandable terms to a human how a classifier works (Doshi-Velez & Kim, 2017). There is currently no clear metric to measure interpretability but rather a consensus on which models are (or are not) interpretable (J. Zhou, Gandomi, Chen, & Holzinger, 2021). A variety of reasons explain the rising interest of interpretability. For example, in the General Data Protection Regulation (GDRP), the notion of a *right to explanation* was introduced (Goodman & Flaxman, 2017; Wachter, Mittelstadt, & Floridi, 2017). Interpretability may also allow users to have more confidence in the results of a classifier which is particularly important when taking sensitive decisions (e.g. medical or legal applications).

Interpretability and prediction performances are usually conflicting goals as the most efficient classifiers tend to be very complex (e.g., deep neural networks). This has been amplified by the fact that, for the last decades, research in supervised classification has mainly focused on performances rather than interpretability. Two main approaches can generally be considered to bring interpretability in supervised classification. The first approach, applied to the most complex classifiers, consists in developping post-hoc models that can, to some extent, explain a classifier predictions. For example, in LIME (Ribeiro, Singh, & Guestrin, 2016) (Local Interpratable Model-Agnostic Explanation) the classifier is used to predict the class of data in the neighbourhood of data from the training set. A simpler classifier is then generated based on these predictions. Counterfactual explanations can also explain a prediction by giving the smallest change of the features vector X_i that would have changed the outcome of data i prediction (Wachter, Mittelstadt, & Russell, 2017). The second approach consists in considering classifiers with a human-understandable decision process, in other words classifiers that are intrinsically interpretable (Rudin, 2019). This article falls into this second category as we consider decision tree classifiers. In order to both obtain interpretability and good performances, we focus on the exact resolution of the training problem.

A decision tree is an oriented binary tree $\mathcal{T} = (\mathcal{N} \cup \mathcal{L}, E)$ which associates a split function $f_t : \mathbb{R}^{|\mathcal{J}|} \to \{true, false\}$ to each of its internal node $t \in \mathcal{N}$ and a class $k \in \mathcal{K}$ to each of its leaves $\ell \in \mathcal{L}$. A data $i \in \overline{\mathcal{I}}$ is classified by following a path from the root to a leaf. This path is determined by applying the split functions of each internal node reached to the features vector X_i. Data i follows the left branch of node $t \in \mathcal{N}$ if $f_t(X_i)$ is true and the right branch otherwise. The class predicted by the classifier is the one associated with the leaf reached by data i.

The split functions are generally linear functions $a^\intercal X_i < b$ with $a \in \mathbb{R}^{|\mathcal{J}|}$ and $b \in \mathbb{R}$. When the vectors a are restricted to be binary unit vectors, the tree is said to have *axis-aligned* splits. Otherwise, we say that the splits are *oblique*.

The problem of the construction of an optimal decision tree is NP-complete (Laurent & Rivest, 1976) and the most used approaches are greedy heuristics such as CART (Breiman, Friedman, Olshen, & Stone, 1984). This algorithm starts by a tree composed only of its root node. At each step one leaf is transformed into an internal node which split function is determined by optimizing an impurity measure (e.g., the Gini impurity (Jost, 2006)) over all the training data reaching it. This process is recursively applied to the two new generated leaves unless the maximum depth is reached, or when all the data reaching a leaf belong to the same class. This approach is fast but does not provide any guarantee on the performances of the obtained tree compared to an optimal one.

Recently, exact solution methods for the construction of an optimal decision tree were introduced. For instance, a dynamic programming algorithm called $MurTree$ that is limited to binary features vectors (i.e. $X_i \in \{0,1\}^{|\mathcal{J}|}$) was introduced in (Demirović et al., 2022). Mathematical programs that build Optimal Classification Trees (OCT) have also been introduced (Bertsimas & Dunn, 2017; Aghaei, Gomez, & Vayanos, 2020; Verwer & Zhang, 2017). In particular, Bertsimas and al. (Bertsimas & Dunn, 2017) proposed two Mixed Integer Linear Programs (MILP). The first one, that we call (T) in this paper, considers *axis-aligned* splits while the second, called here (T^H) is an extension to the case of *oblique* splits. These two formulations handle datasets with real-valued features (i.e. $X_i \in \mathbb{R}^{|\mathcal{J}|}$). A formulation based on flows, that we call (F^b), was introduced by Aghaei and al. (Aghaei et al., 2020). This model only applies to *axis-aligned* split functions and binary features vectors (i.e. $X_i \in \{0,1\}^{|\mathcal{J}|}$). The authors proved that (F^b) provides a stronger continuous relaxation than a variant of problem (T) dedicated to binary features vectors and that it can be solved through a Benders decomposition. For large datasets, solving these MILPs with standard solvers is still a challenge, thus to support scaling-up, models (T) and (T^H) are actually solved heuristically. Note that all these formulations consider two conflicting objectives. The first objective corresponds to the number of classification errors on the training set while the second objective represents the complexity of the tree. This complexity is represented by the number of internal nodes which perform a split in the axis-aligned case and by the

total number of non-zero coefficients in the splits of the tree in the oblique case. A first motivation to minimize this second objective is that it improves the interpretability of the classifier. It can also reduce *overfitting* which occurs when a classifier fits too well to the training set and learns details that may not extrapolate well to the test set. A similar notion of complexity called *sparsity* is also considered for a variant of the classification trees in which the path of the data is stochastic (Blanquero, Carrizosa, Molero-Río, & Morales, 2020).

We propose in this paper several new formulations of the problem of determining the optimal decision tree based on formulations (T) and (F^b). A first contribution is the introduction of a new way to count the number of classification errors in (T). For this, we build a quadratic formulation (Q) that we linearize in two different ways. We obtain two MILPs, (QF) and (QG), which we compare from the continuous relaxation point of view. In particular, we prove that these two new models provide better continuous relaxation bounds than (T). We also prove that our models can be easily extended to the *oblique* case. Then, we propose an extension (F^H) of model (F^b) to the *oblique* case which is also able to handle non-binary features vectors (i.e. $X_i \in \mathbb{R}^{|\mathcal{J}|}$). Then, we prove that (F^H) has a stronger relaxation than (T^H), and can even provide better optimal solutions. All these formulations generally have several solution which are optimal for the training set. To select one that will have the best performances on the *test set*, we propose a heuristic that locally shifts the split functions as far as possible from the data. We also provide an algorithm that fits the parameters (e.g. the depth of the tree or the weight of the second objective) of a formulation by solving less MIPs than the algorithm proposed by Bertsmias and al. (Bertsimas & Dunn, 2017). Finally, we provide extensive experimental results.

The paper is organised as follows. In Section 2, we present related works and in particular the formulations (T) and (F^b). In Section 3, we present our new quadratic formulations of (T) together with two linearizations. Then, in Section 4, we show how to extend model (F^b) to real-valued datasets. In Section 5, we describe how we select an optimal solution and we fit the parameters of our formulations. Finally, in Section 6, we present computational results where we compare our approaches to state-of-the-art methods.

2 Related Works

We present in this section formulations (T) and (F^b) on which our work is based. Without loss of generality, we consider that the value of each feature belongs to $[0, 1]$.

2.1 Optimal Classification Tree

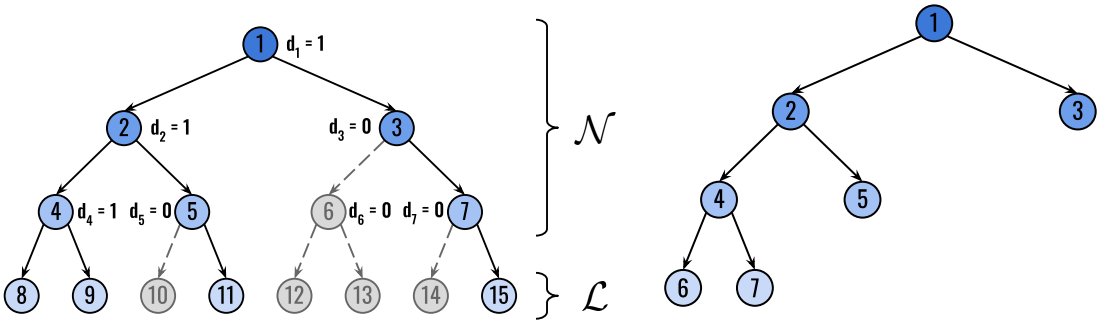
Bertsimas et al. (Bertsimas & Dunn, 2017) introduced two models. The first one computes trees where the split functions, $a^\intercal X_i < b$, are axis-aligned splits (i.e. $a_j \in \{0, 1\}$ and $\sum_{j\in\mathcal{J}} a_j = 1$) while in the second oblique splits are allowed (i.e. $a_j \in [-1, 1]$ and $\sum_{j\in\mathcal{J}} |a_j| \leq 1$).

2.1.1 The axis-aligned case

The basic idea is to introduce variables that define the structure of the tree. Then, to each leaf is associated a class $k \in \mathcal{K}$. Finally, each data i, is assigned to the leaf it falls in according to its path in the tree. Formulation $(T_{\alpha,\beta,\delta})$ considers three parameters which control the structure of the tree: δ the depth of the tree, α the penalty that limits the number of splitting nodes, and β the minimal number of data reaching a leaf. In the following, we denote by $r \in \mathcal{N}$ the root of the tree, and by $a(t)$ the direct ancestor of node $t \in \mathcal{N}\backslash\{r\}$. Let P_ℓ be the path from r to a leaf $\ell \in \mathcal{L}$, we denote by $A_L(\ell)$ ($A_R(\ell)$ respectively), the subset of P_ℓ whose left (right resp.) child is in P_ℓ.

Since the structure of an optimal tree of depth δ is not known a priori, $(T_{\alpha,\beta,\delta})$ associates variables to each node of the complete tree (i.e. $|\mathcal{N}| = 2^\delta - 1$ and $|\mathcal{L}| = 2^\delta$). The split function of each internal node $t \in \mathcal{N}$ is determined by $|\mathcal{J}|$ variables $a_{j,t} \in \{0,1\}$ and one variable $b_t \in [0,1]$. The class predicted by a leaf $\ell \in \mathcal{L}$ is determined by variables $c_{k,\ell}$ which are equal to 1 if and only if class k is assigned to leaf ℓ. A data i assigned to leaf ℓ has to satisfy $\sum_{j\in\mathcal{J}} a_{j,t_r} x_{i,j} \geq b_{t_r}$ for all $t_r \in A_R(\ell)$, and $\sum_{j\in\mathcal{J}} a_{j,t_l} x_{i,j} < b_{t_l}$ for all $t_l \in A_L(\ell)$. To satisfy the latter strict inequalities, a vector $\mu \in \mathbb{R}^{|\mathcal{J}|}$ as well as two scalars $\mu^- = \min_{j\in\mathcal{J}}(\mu_j)$ and $\mu^+ = \max_{j\in\mathcal{J}}(\mu_j)$ are introduced. Each μ_j is the smallest positive interval between values of the training data on feature j (i.e. $\mu_j = \min\{|x_{i_1,j} - x_{i_2,j}|$, s.t. $x_{i_1,j} \neq x_{i_2,j}, i_1, i_2 \in \mathcal{I}^2\}$). To follow the path of $X_i \in \mathcal{I}$ in the tree, binary variables $z_{i,\ell}$ that indicate if X_i reaches leaf ℓ are introduced.

To avoid overfitting, solutions representing non-complete trees are allowed, i.e. some nodes may not apply a split if the trade-off between decreasing classification errors and increasing complexity is not beneficial. We refer to nodes that apply a split as *active*. Variable d_t is equal to 1 if and only if node $t \in \mathcal{N}$ is active. In the model, all data reaching an *inactive* node $t \in \mathcal{N}$ is sent to its right branch by setting $a_{j,t}$ and b_t to 0 for all $j \in \mathcal{J}$. Thus, some leaves may not be reached by any data. We denote leaves reached by data as *opened*. Variable l_ℓ is equal to 1 if and only if leaf $\ell \in \mathcal{L}$ is opened. Let us illustrate these notations by considering a solution in which the variables d_t take the values represented in Figure 1a. Four of the splitting nodes are inactive (3, 5, 6, and 7) and four of the leaves are closed (10, 12, 13, 14) leading to a decision tree which structure is represented in Figure 1b. Since all the data reaching an inactive node are sent to its right branch, leaf 3 in Figure 1b will be assigned the class predicted by node 15 in Figure 1a.



(a) Value of variables d_t.

(b) Structure of the corresponding decision tree.

Figure 1: Link between the value of variables d_t of $(T_{\alpha,\beta,\delta})$ and the structure of the decision tree obtained.

Additional variables are required to count the number of misclassifications. Variable N_ℓ is equal to the number of data reaching leaf $\ell \in \mathcal{L}$ and variable $N_{k,\ell}$ is the number of these data which class is $k \in \mathcal{K}$. Finally, variables L_ℓ is equal to the number of misclassifications in leaf $\ell \in \mathcal{L}$.

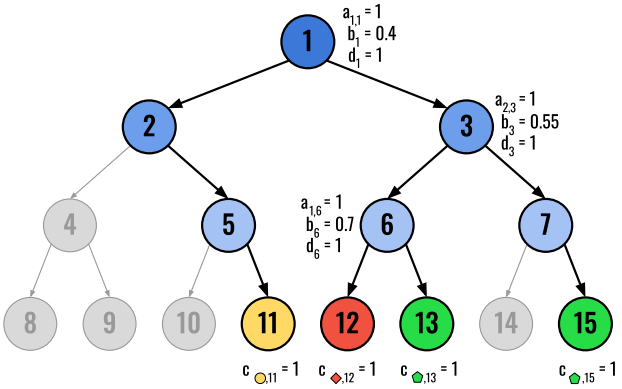
The objective function is a weighted sum of two criteria. The first criterion minimizes the number of misclassifications weighted by a constant $\hat{L}$ that corresponds to the number of misclassifications for an optimal tree of depth $\delta = 0$. The second one minimizes the number of active nodes weighted by parameter $\alpha \geq 0$, and allows to control both the classifier interpretability and the overfitting by penalizing the number of active nodes. Indeed, reducing the number of split functions make the classifier easier to understand. Moreover, performing too many splits, may produce a tree which is very good on the training set but generalizes poorly to the test set. Note that α should be smaller than 1, otherwise, due to constant $\hat{L}$, the solution will always be a tree reduced to its root. Model

$(T_{\alpha,\beta,\delta})$ is formulated as follows:

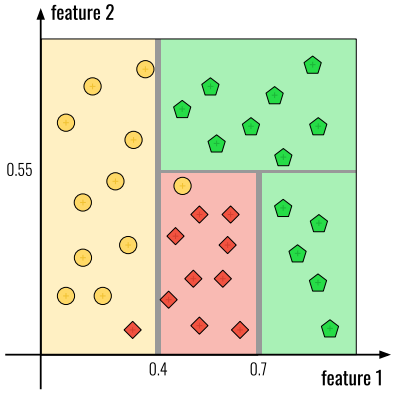
$$
(T_{\alpha,\beta,\delta})\begin{cases}
\min \ \dfrac{1}{\hat{L}}\displaystyle\sum_{\ell\in\mathcal{L}} L_\ell + \alpha \sum_{t\in\mathcal{N}} d_t & \\
\text{s.t.} \displaystyle\sum_{j\in\mathcal{J}} a_{j,t} = d_t & t\in\mathcal{N} \quad (1)\\
0 \le b_t \le d_t & t\in\mathcal{N} \quad (2)\\
d_t \le d_{a(t)} & t\in\mathcal{N}\setminus\{r\} \quad (3)\\
\displaystyle\sum_{k\in\mathcal{K}} c_{k,\ell} = l_\ell & \ell\in\mathcal{L} \quad (4)\\
\displaystyle\sum_{i\in\mathcal{I}} z_{i,\ell} \ge \beta l_\ell & \ell\in\mathcal{L} \quad (5)\\
z_{i,\ell} \le l_\ell & \ell\in\mathcal{L}, i\in\mathcal{I} \quad (6)\\
\displaystyle\sum_{\ell\in\mathcal{L}} z_{i,\ell} = 1 & i\in\mathcal{I} \quad (7)\\
\displaystyle\sum_{j\in\mathcal{J}} a_{j,t}\left(x_{i,j}+\mu_j-\mu^-\right)+\mu^- \le b_t + (1+\mu^+)(1-z_{i,\ell}) & i\in\mathcal{I}, \ell\in\mathcal{L}, t\in A_L(\ell) \quad (8)\\
\displaystyle\sum_{j\in\mathcal{J}} a_{j,t}x_{i,j} \ge b_t - (1-z_{i,\ell}) & i\in\mathcal{I}, \ell\in\mathcal{L}, t\in A_R(\ell) \quad (9)\\
N_{k,\ell} = \displaystyle\sum_{i\in\mathcal{I}, y_i=k} z_{i,\ell} & \ell\in\mathcal{L}, k\in\mathcal{K} \quad (10)\\
N_\ell = \displaystyle\sum_{i\in\mathcal{I}} z_{i,\ell} & \ell\in\mathcal{L} \quad (11)\\
L_\ell \ge N_\ell - N_{k,\ell} - |\mathcal{I}|(1-c_{k,\ell}) & \ell\in\mathcal{L}, k\in\mathcal{K} \quad (12)\\
L_\ell \le N_\ell - N_{k,\ell} + |\mathcal{I}|c_{k,\ell} & \ell\in\mathcal{L}, k\in\mathcal{K} \quad (13)\\
L_\ell \ge 0 & \ell\in\mathcal{L} \quad (14)\\
a_{j,t}\in\{0,1\}, d_t\in\{0,1\} & t\in\mathcal{N}, j\in\mathcal{J} \quad (15)\\
c_{k,\ell}\in\{0,1\}, l_\ell\in\{0,1\}, z_{i,\ell}\in\{0,1\} & \ell\in\mathcal{L}, k\in\mathcal{K}, i\in\mathcal{I} \quad (16)
\end{cases}
$$

The first sets of constraints fix the structure of the tree. Constraints (1) and (2) ensure that the variables defining the split function of $t \in \mathcal{N}$ vanish when t is inactive. Constraints (3) inactivate all descendants of an inactived node. Constraints (4) ensure that one and only one class is assigned to each opened node. The second part of the model follows the path of the data in the tree. Constraints (5) and (6) open a leaf only if at least β data reach it. Constraints (7) ensure that each data is assigned to one and only one leaf, and Constraints (8) - (9) that the path of each data is consistent with the split functions. The last set of constraints counts the number of misclassifications L_ℓ (Constraints (12) and (14)), by first fixing the counting variables $N_{k,\ell}$ and N_ℓ with Constraints (10) and (11), respectively. Note that since the misclassifications are minimized, an optimal solution always assigns to a leaf ℓ the most represented class among the data reaching ℓ.

As an example, Figure 2a represents the optimal solution of (T) for the training set represented in Figure 2b. Each split function on a feature $j \in \mathcal{J}$ is represented by a grey line of thickness μ_j. In this example, we have $\delta = 3$, $\alpha = 0.1$ and $\beta = 4$, and the optimal tree makes 2 classification errors. Observe that the missclassified red square could be properly classified by adding a split function on node 2, but this could be seen as overfitting. This solution is thus cut off by Constraint (5) since $\beta = 4$.



(a) Value of variables a, b, c, and d that are non-zero in the solution.



(b) The training set. For any point in $[0, 1]^2$, the background color corresponds to the class predicted by the decision tree.

Figure 2: Optimal tree obtained by $(T_{0.1,\,4,\,3})$ on a training set containing 3 classes, 2 features and 34 data.

2.1.2 The oblique case

An extension to the oblique case was also proposed in (Bertsimas & Dunn, 2017) which allows split functions that use a linear combination of more than one feature. We label the associated model as $(T^H_{\alpha,\beta,\delta,\mu})$ since oblique splits are in fact hyperplanes.

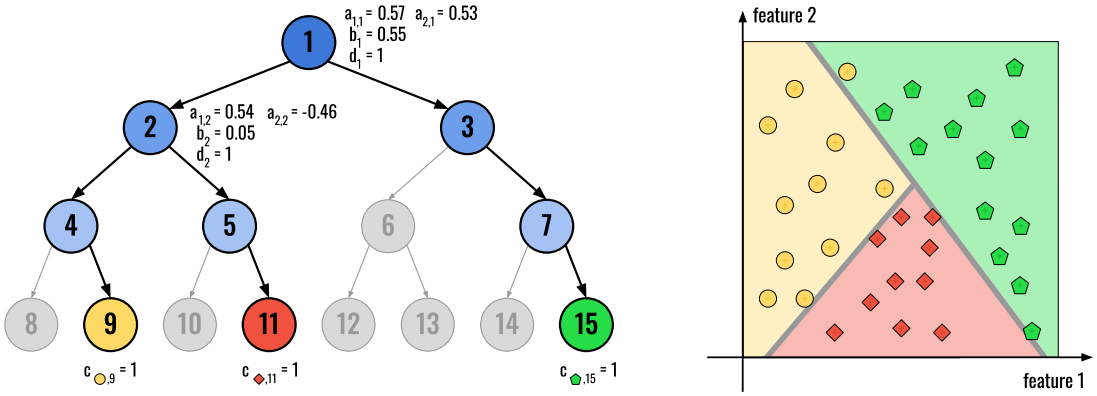
We present the differences with Formulation $(T_{\alpha,\beta,\delta})$. In this extension, variables $a_{j,t}$ are now defined in $[-1, 1]$ rather than in $\{0, 1\}$. As in the previous model, a node $t \in \mathcal{N}$ is inactive if and only if its split function only has null coefficients. To model it, the authors use the constraints $\sum_{j\in\mathcal{J}} |a_{j,t}| \leq d_t$ that they linearized with auxiliary variables $\hat{a}_{j,t} = |a_{j,t}|$. Then, the strict inequalities of the split functions are now handled by a new parameter $\mu \in \mathbb{R}$. Fixing a relevant value to parameter μ is a difficult task. If it is too small, in particular smaller than the accuracy of the solver, it may cause numerical issues. Note that in this formulation, it is not possible to generate split functions within the interval $[b_t - \mu, b_t]$, so if μ is chosen too large this may affect the objective value of an optimal solution. Finally, since a split function can now involve up to $|\mathcal{J}|$ non-zero coefficients, minimizing the number of active node is not a relevant second objective anymore. Instead, the authors minimize the number of non-zero coefficients $\{a_{j,t}\}_{j\in\mathcal{J},\, t\in\mathcal{N}}$ using auxiliary binary variables $s_{j,t}$ equal to 1 if and

only if $a_{j,t} \neq 0$. Formulation $(T^H_{\alpha,\beta,\delta,\mu})$ is the following:

$$
(T^H_{\alpha,\beta,\delta,\mu}) \begin{cases}
\min \dfrac{1}{\hat{L}} \displaystyle\sum_{\ell \in \mathcal{L}} L_\ell + \alpha \sum_{t \in \mathcal{N}, j \in \mathcal{J}} s_{j,t} & & \\
\text{s.t.} (3)-(7), (10)-(14), (16) & & \\
\displaystyle\sum_{j \in \mathcal{J}} \hat{a}_{j,t} \leq d_t & t \in \mathcal{N} & (17) \\
-\hat{a}_{j,t} \leq a_{j,t} \leq \hat{a}_{j,t} & j \in \mathcal{J}, t \in \mathcal{N} & (18) \\
-s_{j,t} \leq a_{j,t} \leq s_{j,t} & j \in \mathcal{J}, t \in \mathcal{N} & (19) \\
s_{j,t} \leq d_t & j \in \mathcal{J}, t \in \mathcal{N} & (20) \\
\displaystyle\sum_{j \in \mathcal{J}} s_{j,t} \geq d_t & t \in \mathcal{N} & (21) \\
-d_t \leq b_t \leq d_t & t \in \mathcal{N} & (22) \\
\displaystyle\sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} + \mu \leq b_t + (2+\mu)(1 - z_{i,\ell}) & i \in \mathcal{I}, \ell \in \mathcal{L}, t \in A_L(\ell) & (23) \\
\displaystyle\sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} \geq b_t - 2(1 - z_{i,\ell}) & i \in \mathcal{I}, \ell \in \mathcal{L}, t \in A_R(\ell) & (24) \\
s_{j,t} \in \{0,1\}, d_t \in \{0,1\} & t \in \mathcal{N}, j \in \mathcal{J} & (25)
\end{cases}
$$

In $(T^H_{\alpha,\beta,\delta,\mu})$, Constraints (17) and (18) ensure that $\hat{a}_{j,t} = |a_{j,t}|$. The definition of variables $s_{j,t}$ and their link with d_t are enforced by constraints (19) - (21). Constraints (22) enables variable b_t to be non-positive, and Constraints (23) and (24) define oblique split functions.

Figure 3a represents an optimal solution of (T^H) on the training set of Figure 3b (same as in Figure 2b). In this example, we have $\mu = 0.02$, $\delta = 3$, $\alpha = 0.04$ and $\beta = 4$, and all split functions have a thickness of μ. Observe that for the same value of parameters δ, α and β, and for $\mu \leq \min_{j \in \mathcal{J}} \mu_j$, all the feasible solutions of $(T_{\alpha,\beta,\delta})$ are always feasible for $(T^H_{\alpha,\beta,\delta,\mu})$.



(a) Value of variables a, b, c, and d that are non-zero in the solution.

(b) The training set.

Figure 3: Optimal tree obtained by $(T^H_{0.04,\,4,\,3,\,0.02})$ on a training set containing 3 classes, 2 features and 34 data.

2.2 A Strong Max-Flow Formulation

We now present an alternative linear program that is based on max-flow formulation (Aghaei et al., 2020). It only handles binary features vectors (i.e. $X_i \in \{0,1\}^{\mathcal{J}}$), and axis-aligned split functions. The

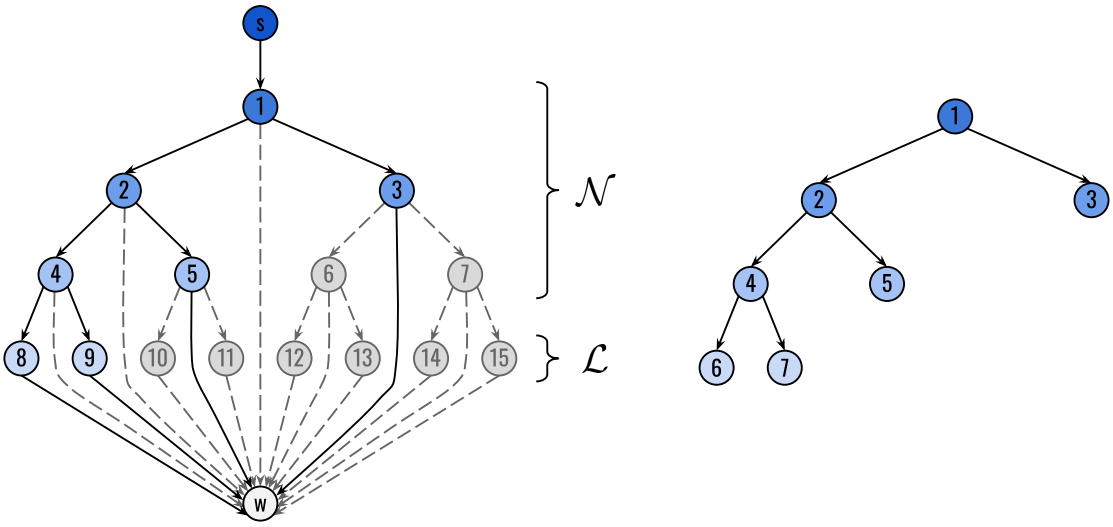
main idea of this formulation is to associate a unitary flow to each data i that is correctly classified.

The authors first define a flow network $\mathcal{G} = (\mathcal{V} = \mathcal{N} \cup \mathcal{L} \cup \{s, w\}, \mathcal{E} = E \cup \{(s, r), (t, w)_{t \in \mathcal{N} \cup \mathcal{L}}\})$, where a source s is connected by an arc (s, r) to the root r of the tree, and a sink w is connected by an arc (t, w) to all nodes $t \in \mathcal{N} \cup \mathcal{L}$ (see example in Figure 4a). In this model any node $t \in \mathcal{N} \cup \mathcal{L}$ can predict a class $k \in \mathcal{K}$, and the binary variable $g_{k,t}$ indicates if node t is assigned to class k. Then, to model the split functions, a binary variable $h_{j,t}$ states if the split function of node $t \in \mathcal{N}$ is performed on feature $j \in \mathcal{J}$. If it is, a data i reaching node t will go to its left child $l(t)$ if $x_{i,j} = 0$, and to its right child $r(t)$ if $x_{i,j} = 1$. Let P_i, be the path of a data $i \in \mathcal{I}$ in $\mathcal{G}$. A binary flow $\{u^i_{v,v'}\}_{(v,v') \in \mathcal{E}}$ is associated to i, and is equal to 1 if and only if i is correctly classified and $(v, v') \in P_i$. The class assigned to i being that of the direct ancestor of w in P_i.

The objective function is similar to that of $(T_{\alpha,\beta,\delta})$, but the parameter which weights the criterion is a scalar $\lambda \in [0, 1[$. As in model $(T_{\alpha,\beta,\delta})$, the depth of the tree δ is a parameter of this formulation, but there is no parameter β controlling the minimal number of data reaching a leaf. Formulation of is as follows:

$$
(F^b_{\delta,\lambda}) \begin{cases}
\max (1-\lambda) \displaystyle\sum_{i \in \mathcal{I}} u^i_{s,r} - \lambda \sum_{t \in \mathcal{N}} \sum_{j \in \mathcal{J}} h_{j,t} & \\
\text{s.t.} \displaystyle\sum_{j \in \mathcal{J}} h_{j,t} + \sum_{k \in \mathcal{K}} g_{k,t} = 1 & t \in \mathcal{N} & (26) \\
\displaystyle\sum_{k \in \mathcal{K}} g_{k,\ell} = 1 & \ell \in \mathcal{L} & (27) \\
u^i_{a(t),t} = u^i_{t,l(t)} + u^i_{t,r(t)} + u^i_{t,w} & t \in \mathcal{N}, i \in \mathcal{I} & (28) \\
u^i_{a(\ell),\ell} = u^i_{\ell,w} & t \in \mathcal{L}, i \in \mathcal{I} & (29) \\
u^i_{t,l(t)} \leq \displaystyle\sum_{j \in \mathcal{J}: x_{i,j}=0} h_{j,t} & t \in \mathcal{N}, i \in \mathcal{I} & (30) \\
u^i_{t,r(t)} \leq \displaystyle\sum_{j \in \mathcal{J}: x_{i,j}=1} h_{j,t} & t \in \mathcal{N}, i \in \mathcal{I} & (31) \\
u^i_{t,w} \leq g_{y_i,t} & i \in \mathcal{I}, t \in \mathcal{N} \cup \mathcal{L} & (32) \\
h_{j,t} \in \{0, 1\} & t \in \mathcal{N}, j \in \mathcal{J} & (33) \\
g_{k,t} \in \{0, 1\} & t \in \mathcal{N} \cup \mathcal{L}, k \in \mathcal{K} & (34) \\
u^i_{t,t'} \in \{0, 1\} & i \in I, (t, t') \in \mathcal{E} & (35)
\end{cases}
$$

Constraints (26) ensure that each node either performs a split or predicts a class, and Constraints (27) that one and only one class is assigned to each leaf. Constraints (28) and (29) hold for the flow conservation. Constraints (30) and (31) ensure the consistency of the split functions. Constraints (32) impose that the flow of a misclassified data vanishes. Note that when a node $t \in \mathcal{N}$ is inactive, any data reaching it necessarily follows the arc (t, w). This is illustrated in the flow network represented in Figure 4a. In this solution of (F^b), nodes 3 and 5 are inactive and, consequently, the flow of the data cannot go through arcs $(3, 6)$, $(3, 7)$, $(5, 10)$ and $(5, 11)$. This leads to a decision tree which structure is represented in Figure 4b.



(a) The flow network. Grayed arcs and vertices are not reached by any training data.

(b) Structure of the corresponding decision tree.

Figure 4: Link between the flow of the data in the graph of a solution of Formulation (F^b) and the structure of the decision tree obtained.

In (Aghaei et al., 2020), it is proven that $(F^b_{\delta,\lambda})$ has a stronger relaxation than a restriction of $(T_{\alpha,\beta,\delta})$ to the case of binary feature vectors (i.e. $X_i \in \{0,1\}^{|\mathcal{J}|}$).

3 A new quadratic formulation based on $(T_{\alpha,\beta,\delta})$

We propose in this section a new non linear modelisation of both problems $(T_{\alpha,\beta,\delta})$ and $(T^H_{\alpha,\beta,\delta,\mu})$ by rewritting some of their constraints in order to reduce their sizes.

3.1 A quadratic formulation for the axis-aligned case

In $(T_{\alpha,\beta,\delta})$, variables L_ℓ that model the number of misclassified data reaching leaf $\ell \in \mathcal{L}$, are fixed thanks to N_ℓ, the total number of data reaching leaf ℓ, and $N_{k,\ell}$, the number of data of class $k \in \mathcal{K}$ reaching leaf ℓ. More formally:

$$L_\ell = N_\ell - \max_{k \in \mathcal{K}} N_{k,\ell} \tag{36}$$

This equality is true since the class assigned to leaf ℓ in an optimal solution of $(T_{\alpha,\beta,\delta})$ is precisely $\arg\max_{k \in \mathcal{K}} N_{k,\ell}$. In $(T_{\alpha,\beta,\delta})$, Equation (36) is linearized thanks to Constraints (10)-(14).

We introduce a new writting of variable L_ℓ in order to define a more compact formulation. Recall that variables $c_{k,\ell} = 1$ if class $k \in \mathcal{K}$ is assigned to leaf $\ell \in \mathcal{L}$, and variables $z_{i,\ell} = 1$ if data i reaches leaf ℓ. For a given $k \in \mathcal{K}$, let $\mathcal{I}_k$ be the subset of $\mathcal{I}$ restricted to data of class k (i.e., $\mathcal{I}_k = \{i \in \mathcal{I} \mid y_i = k\}$). Variable L_ℓ can be expressed as:

$$L_\ell = \sum_{k\in\mathcal{K}} c_{k,\ell}\,(N_\ell - N_{k,\ell}) = \sum_{k\in\mathcal{K}}(c_{k,\ell} \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell}) = \sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} c_{k,\ell} z_{i,\ell} \tag{37}$$

We propose to use Equation (37) to model the number of misclassified data, which we directly include into the objective function. Doing this allows us to reduce the size of the formulation by removing variables L_ℓ, N_ℓ, $N_{k,\ell}$ and Constraints (10)-(14). We obtain the following quadratic formulation:

$$(Q_{\alpha,\beta,\delta})\begin{cases} \min \quad F_{\hat{L},\alpha}(c,z,d) = \dfrac{1}{\hat{L}}\displaystyle\sum_{\ell\in\mathcal{L}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} c_{k,\ell} z_{i,\ell} + \alpha \sum_{t\in\mathcal{N}} d_t & \\ \text{s.t.} \quad (1)-(9), (15), (16) & \\ \qquad l_\ell \le d_t & \ell\in\mathcal{L},\ t\in A_L(\ell) \end{cases} \tag{38}$$

where Constraints (38) are valid inequalities that strengthen our formulation by ensuring that a non *active* node does not send the data through its left branch. The quadratic objective function $F_{\hat{L},\alpha}$ of $(Q_{\alpha,\beta,\delta})$ is a non-convex function which we propose to convexify through two different approaches: the linearization of Fortet (Fortet, 1959) and that of Glover (Glover, 1975). Further, we compare the continuous relaxation of the two corresponding linear programs.

Fortet's linearization (Fortet, 1959) consists in replacing each bi-linear term $z_{i,\ell}c_{k,\ell}$ by an auxiliary variable $\theta_{i,k,\ell}$. The equality $\theta_{i,k,\ell} = z_{i,\ell}c_{k,\ell}$ is then enforced by the following sets of linear inequalities:

$$\begin{cases} \theta_{i,k,\ell} \le c_{k,\ell} & i\in\mathcal{I}\setminus\mathcal{I}_k,\ k\in\mathcal{K}, \ell\in\mathcal{L} \quad (39)\\ \theta_{i,k,\ell} \le z_{i,\ell} & i\in\mathcal{I}\setminus\mathcal{I}_k,\ k\in\mathcal{K}, \ell\in\mathcal{L} \quad (40)\\ \theta_{i,k,\ell} \ge c_{k,\ell} + z_{i,\ell} - 1 & i\in\mathcal{I}\setminus\mathcal{I}_k,\ k\in\mathcal{K}, \ell\in\mathcal{L} \quad (41)\\ \theta_{i,k,\ell} \ge 0 & i\in\mathcal{I}\setminus\mathcal{I}_k,\ k\in\mathcal{K}, \ell\in\mathcal{L} \quad (42) \end{cases}$$

It is easy to verify that inequalities (39)-(42) are equivalent to $\theta_{i,k,\ell} = z_{i,\ell}c_{k,\ell}$ as $z_{i,\ell}$ and $c_{k,\ell}$ are binary variables. Moreover, since in $(Q_{\alpha,\beta,\delta})$, the products $z_{i,\ell}c_{k,\ell}$ are only involved into the objective function, and are weighted with non-negative coefficients, Constraints (39) and (40) are not necessary to get the equivalence. We finally obtain the following linear reformulation of $(Q_{\alpha,\beta,\delta})$:

$$(QF_{\alpha,\beta,\delta})\begin{cases} \min \quad \mathrm{FL}^1_{\hat{L},\alpha}(\theta,d) = \frac{1}{\hat{L}}\displaystyle\sum_{\ell\in\mathcal{L}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} \theta_{i,k,\ell} + \alpha \sum_{\ell\in\mathcal{N}} d_\ell \\ \text{s.t.} \quad (1)\text{-}(9),\ (15),\ (16),\ (38),\ (41),\ (42) \end{cases}$$

To further reduce the size of the formulation, we propose to use Glover's procedure (Glover, 1975) to linearize the products $c_{k,\ell}\Big(\displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell}\Big)$ by use of auxiliary variables $\Theta_{k,\ell}$. It is easy to prove that $|\mathcal{I}\setminus\mathcal{I}_k|$ is a valid upper bound for $\displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell}$. Using these bounds and the fact that $\displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell} \ge 0$, the equality $\Theta_{k,\ell} = c_{k,\ell}\Big(\displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell}\Big)$ is then enforced by the following set of linear inequalities:

$$\begin{cases} \Theta_{k,\ell} \le |\mathcal{I}\setminus\mathcal{I}_k| c_{k,\ell} & k\in\mathcal{K},\ \ell\in\mathcal{L} \quad (43)\\ \Theta_{k,\ell} \le \displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell} & k\in\mathcal{K},\ \ell\in\mathcal{L} \quad (44)\\ \Theta_{k,\ell} \ge 0 & k\in\mathcal{K},\ \ell\in\mathcal{L} \quad (45)\\ \Theta_{k,\ell} \ge \displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell} - |\mathcal{I}\setminus\mathcal{I}_k|(1-c_{k,\ell}) & k\in\mathcal{K},\ \ell\in\mathcal{L} \quad (46) \end{cases}$$

Here again we only need inequalities (45) and (46) to get an equivalent linear reformulation:

$$(QG_{\alpha,\beta,\delta})\begin{cases}\min & FL^2_{\hat{L},\alpha}(\Theta,d)=\dfrac{1}{\hat{L}}\displaystyle\sum_{k\in K}\sum_{\ell\in\mathcal{L}}\Theta_{k,\ell}+\alpha\sum_{t\in\mathcal{N}}d_t\\ \text{s.t.} & (1)-(9),(15),(16),(38),(45),(46)\end{cases}$$

We now compare our two linear reformulations of $(Q_{\alpha,\beta,\delta})$. We first observe that the size of $(QG_{\alpha,\beta,\delta})$ (i.e. $\mathcal{O}(|\mathcal{K}|\times|\mathcal{L}|)$ auxiliary variables and constraints) is reduced by a factor of $|\mathcal{I}|$ in comparison to the size of $(QF_{\alpha,\beta,\delta})$ (i.e. $\mathcal{O}(|\mathcal{K}|\times|\mathcal{L}|\times|\mathcal{I}|)$ auxiliary variables and constraints). Since $|\mathcal{I}|$ is higher than both $|\mathcal{K}|$ and $|\mathcal{L}|$ in non-trivial problems, this reduction can be significant. Let us denote by $v(P)$ the optimal value of a problem (P), and $\overline{P}$ the continuous relaxation of problem P. We state in Proposition 1 that problem $(QF_{\alpha,\beta,\delta})$ has a better continuous relaxation than problem $(QG_{\alpha,\beta,\delta})$.

Proposition 1. $v(\overline{QF}_{\alpha,\beta,\delta})\geq v(\overline{QG}_{\alpha,\beta,\delta})$.

Proof. Let $(d^*,a^*,b^*,l^*,c^*,z^*,\theta^*)$ be an optimal solution of $(\overline{QF}_{\alpha,\beta,\delta})$. We build a feasible solution $(d=d^*,a=a^*,b=b^*,l=l^*,c=c^*,z=z^*,\Theta)$ of $(\overline{QG}_{\alpha,\beta,\delta})$ such that $\Theta_{k,\ell}=\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}\theta^*_{i,k,\ell}$ for all $(\ell,k)\in(\mathcal{L}\times\mathcal{K})$. The only constraints of $(\overline{QG}_{\alpha,\beta,\delta})$ which are not obviously satisfied by this solution are Constraints (46). Let us prove that they are satisfied:

$$\begin{array}{rcl}\Theta_{k,\ell}=\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}\theta^*_{i,k,\ell} & \geq & \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}\left(c^*_{k,\ell}+z^*_{i,\ell}-1\right)\\ & \geq & |\mathcal{I}\setminus\mathcal{I}_k|c_{k,\ell}+\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}z^*_{i,\ell}-|\mathcal{I}|+|\mathcal{I}_k|\\ & \geq & \displaystyle\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}z_{i,\ell}-|\mathcal{I}\setminus\mathcal{I}_k|(1-c_{k,\ell})\end{array}$$

We now compare the objective values $\Delta=FL^2_{\hat{L},\alpha}(\Theta,d)-FL^1_{\hat{L},\alpha}(\theta^*,d^*)$:

$$\Delta=\frac{1}{\hat{L}}\sum_{k\in K}\sum_{\ell\in\mathcal{L}}\Theta_{k,\ell}-\frac{1}{\hat{L}}\sum_{\ell\in\mathcal{L}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}\theta^*_{i,k,\ell}=\frac{1}{\hat{L}}\Big(\sum_{k\in K}\sum_{\ell\in\mathcal{L}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}\theta^*_{i,k,\ell}-\sum_{\ell\in\mathcal{L}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}\theta^*_{i,k,\ell}\Big)=0$$

□

We want to compare the value of $(\overline{QG}_{\alpha,\beta,\delta})$ and $(\overline{QF}_{\alpha,\beta,\delta})$ to the value of $(\overline{T}_{\alpha,\beta,\delta})$. To do so, we first show that the value of $(\overline{T}_{\alpha,\beta,\delta})$ is always 0.

Lemma 1. $v(\overline{T}_{\alpha,\beta,\delta})=0$

Proof. Since all the variables are positive and weighted by positive constants, necessarily $v(\overline{T}_{\alpha,\beta,\delta})\geq 0$. Let ℓ_r denote the right-most leaf of the tree. We build the following solution of value 0:

- $a_{j,t}=b_t=d_t=0\quad\forall j\in\mathcal{J},t\in\mathcal{N}$;
- $z_{i,\ell_r}=l_{\ell_r}=1$, and $z_{i,\ell}=l_\ell=0\quad\forall i\in\mathcal{I},\ell\in\mathcal{L},\ell\neq\ell_r$;
- $N_{\ell_r}=|\mathcal{I}|$, and $N_\ell=0\quad\forall\ell\in\mathcal{L},\ell\neq\ell_r$;
- $N_{k,\ell_r}=|\mathcal{I}_k|$, and $N_{k,\ell}=0\quad\forall k\in\mathcal{K},\ell\in\mathcal{L},\ell\neq\ell_r$;
- $c_{k,\ell_r}=\frac{|\mathcal{I}_k|}{|\mathcal{I}|}$, and $c_{k,\ell_r}=0\quad\forall k\in\mathcal{K},\ell\in\mathcal{L},\ell\neq\ell_r$;

- $L_\ell = 0 \quad \forall \ell \in \mathcal{L}$.

Constraints (1)-(7) and (10)-(14) are obviously satisfied, and Constraints (8) and (9) are verified since all data go to the right-most leaf. The value of this solution is 0. □

We state in Proposition 2 that our new linear formulations provide better continuous relaxation bounds than the one of $(T_{\alpha,\beta,\delta})$.

Proposition 2. *If* $\alpha > 0$, $v(\overline{QF}_{\alpha,\beta,\delta}) \geq v(\overline{QG}_{\alpha,\beta,\delta}) > v(\overline{T}_{\alpha,\beta,\delta})$.

Proof. Considering Lemma 1, it remains to prove that $v(\overline{QG}_{\alpha,\beta,\delta}) > 0$. Let us distinguish two cases:

Case 1: If $\sum_{t\in\mathcal{N}} d_t > 0$, since α is positive, we necessarily have $v(\overline{QG}_{\alpha,\beta,\delta}) > 0$.

Case 2: If $\sum_{t\in\mathcal{N}} d_t = 0$, we have $v(\overline{QG}_{\alpha,\beta,\delta}) = \frac{1}{\hat{L}} \sum_{k\in K} \sum_{\ell\in\mathcal{L}} \Theta_{k,\ell}$. Constraints (1) and (2) impose for every node t and feature j that $a_{j,t} = b_t = 0$. By Constraints (38), we have that for all $\ell \in \mathcal{L} \setminus \{\ell_r\}$ the value of $z_{i,\ell}$ is 0 and $z_{i,\ell_r} = 1$. Therefore, $l_\ell = 1$ if and only if $\ell = \ell_r$, giving us $\sum_{k\in\mathcal{K}} c_{k,\ell_r} = 1$.

Since $(\overline{QG}_{\alpha,\beta,\delta})$ is a minimisation problem and the coefficients of $\theta_{\ell,k}$ in FL^2 are positive, Constraints (45) and (46) enforce $\Theta_{\ell,k} = 0 \ \forall \ell \neq \ell_r$. For ℓ_r we obtain:

$$\Theta_{\ell_r,k} = \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} z_{i,\ell_r} - |\mathcal{I}\setminus\mathcal{I}_k|(1 - c_{k,\ell_r}) = |\mathcal{I}\setminus\mathcal{I}_k|\, c_{k,\ell_r}$$

Thus, $v(\overline{QG}_{\alpha,\beta,\delta}) = \sum_{k\in\mathcal{K}} |\mathcal{I}\setminus\mathcal{I}_k| c_{k,\ell_r} > 0$ since $\sum_{k\in\mathcal{K}} c_{k,\ell_r} = 1$.

□

3.2 Handling the *oblique* case

Equation (37) remains true for *oblique* split functions, and thus the oblique extension of our three formulations is straightforward, since the differences are the same than those between $(T_{\alpha,\beta,\delta})$ and $(T^H_{\alpha,\beta,\delta,\mu})$. Thus, we build $(Q^H_{\alpha,\beta,\delta,\mu})$, $(QF^H_{\alpha,\beta,\delta,\mu})$ and $(QG^H_{\alpha,\beta,\delta,\mu})$ the oblique extensions of $(Q_{\alpha,\beta,\delta})$, $(QF_{\alpha,\beta,\delta})$ and $(QG_{\alpha,\beta,\delta})$, respectively:

$$(Q^H_{\alpha,\beta,\delta,\mu}) \begin{cases} \min & F^H_{\hat{L},\alpha}(c,z,s) = \frac{1}{\hat{L}} \sum_{k\in\mathcal{K}} \sum_{\ell\in\mathcal{L}} \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} c_{k,\ell} z_{i,\ell} + \alpha \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{N}} s_{j,t} \\ s.t. & (3) - (7)(16) - (25) \end{cases}$$

$$(QF^H_{\alpha,\beta,\delta,\mu}) \begin{cases} \min & \mathrm{FL}^{1H}_{\hat{L},\alpha}(\theta,s) = \frac{1}{\hat{L}} \sum_{k\in\mathcal{K}} \sum_{\ell\in\mathcal{L}} \sum_{i\in\mathcal{I}\setminus\mathcal{I}_k} \theta_{i,k,\ell} + \alpha \sum_{j\in\mathcal{J}} \sum_{t\in\mathcal{N}} s_{j,t} \\ s.t. & (3)\text{-}(7)\ (16)\text{-}(25)\ (41)\text{-}(42) \end{cases}$$

$$(QG^H_{\alpha,\beta,\delta,\mu})\begin{cases}\min & \mathrm{FL}^{1H}_{\hat{L},\alpha}(\Theta,s)=\frac{1}{\hat{L}}\sum_{k\in\mathcal{K}}\sum_{\ell\in\mathcal{L}}\Theta_{k,\ell}+\alpha\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{N}}s_{j,t}\\ s.t. & \text{(3)-(7) (16)-(25) (45)-(46)}\end{cases}$$

Similarly to the *axis-aligned* case, we compare in Proposition 3 the continuous relaxation values of $(QF^H_{\alpha,\beta,\delta,\mu})$, $(QG^H_{\alpha,\beta,\delta,\mu})$, and $(T^H_{\alpha,\beta,\delta,\mu})$, denoted by $(\overline{QF}^H_{\alpha,\beta,\delta,\mu})$, $(\overline{QG}^H_{\alpha,\beta,\delta,\mu})$, and $(\overline{T}^H_{\alpha,\beta,\delta,\mu})$, respectively. To do so, we show that the value of $(\overline{T}^H_{\alpha,\beta,\delta,\mu})$ is always 0.

Lemma 2. $v(\overline{T}^H_{\alpha,\beta,\delta,\mu})=0$

Proof. Since all the variables are positive and weighted by positive constants, we have $v(\overline{T}^H_{\alpha,\beta,\delta,\mu})\geq 0$. Let t_r denote the right-most leaf, we build the following solution of value 0:

- $a_{j,t}=\hat{a}_{j,t}=s_{j,t}=b_t=d_t=0 \quad \forall j\in\mathcal{J},t\in\mathcal{N}$;
- $z_{i,\ell_r}=l_{\ell_r}=1$, and $z_{i,\ell}=l_\ell=0 \quad \forall i\in\mathcal{I},\ell\in\mathcal{L},\ell\neq\ell_r$;
- $N_{\ell_r}=|\mathcal{I}|$, and $N_\ell=0 \quad \forall \ell\in\mathcal{L},\ell\neq\ell_r$;
- $N_{k,\ell_r}=|\mathcal{I}_k|$, and $N_{k,\ell}=0 \quad \forall k\in\mathcal{K},\ell\in\mathcal{L},\ell\neq\ell_r$;
- $c_{k,\ell_r}=\frac{\mathcal{I}_k}{|\mathcal{I}|}$, and $c_{k,\ell_r}=0 \quad \forall k\in\mathcal{K},\ell\in\mathcal{L},\ell\neq\ell_r$;
- $L_\ell=0 \quad \forall \ell\in\mathcal{L}$.

Constraints (3)-(7), (10)-(14), (17)-(22) are obviously satisfied, and Constraints (23)-(24) are verified since all data go to the right-most leaf. □

We state in Proposition 3 that our new linear formulations provide better continuous relaxation bounds than the original formulation $(T^H_{\alpha,\beta,\delta,\mu})$.

Proposition 3. *If* $\alpha>0$, $v(\overline{QF}^H_{\alpha,\beta,\delta,\mu})\geq v(\overline{QG}^H_{\alpha,\beta,\delta,\mu})>v(\overline{T}^H_{\alpha,\beta,\delta,\mu})$.

Proof. We first prove that $v(\overline{QF}^H_{\alpha,\beta,\delta,\mu})\geq v(\overline{QG}^H_{\alpha,\beta,\delta,\mu})$.
The proof is similar to that of Proposition 1. Let $(d^*,a^*,b^*,l^*,c^*,z^*,s^*,\theta^*)$ be an optimal solution of $(\overline{QF}^H_{\alpha,\beta,\delta,\mu})$. We build a feasible solution to $(\overline{QG}^H_{\alpha,\beta,\delta,\mu})$: $(d=d^*,a=a^*,b=b^*,l=l^*,c=c^*,z=z^*,s=s^*,\Theta)$, with for all $(\ell,k)\in(\mathcal{L}\times\mathcal{K})$ $\Theta_{k,\ell}=\sum_{i\in\mathcal{I}\setminus\mathcal{I}_k}\theta^*_{i,k,\ell}$. This solution obviously satisfies all constraints of $(\overline{QG}^H_{\alpha,\beta,\delta,\mu})$ and its objective value is equal to $v(\overline{QF}^H_{\alpha,\beta,\delta,\mu})$.

Secondly, we prove that $v(\overline{QG}^H_{\alpha,\beta,\delta,\mu})>v(\overline{T}^H_{\alpha,\beta,\delta,\mu})$. Considering Lemma 2, it remains to prove that $v(\overline{QG}^H_{\alpha,\beta,\delta,\mu})>0$. Let us distinguish two cases:

Case 1: If $\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{N}}s_{j,t}>0$, since α is positive, we necessarily have $v(\overline{QG}^H_{\alpha,\beta,\delta,\mu})>0$.

Case 2: If $\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{N}}s_{j,t}=0$, it means that all data are sent to the right-most leaf. With a similar reasoning as the proof of Proposition 2, we get that $v(\overline{QG}^H_{\alpha,\beta,\delta,\mu})>0$.

□

4 Extension of $(F^b_{\delta,\lambda})$ to the case of real valued data

Formulation $(F^b_{\delta,\lambda})$ only applies to datasets in which all features are binary (i.e. $X_i \in \{0,1\}^{|\mathcal{J}|}$). We propose in this section to extend this formulation to the case of real valued data (i.e. $X_i \in \mathbb{R}^{|\mathcal{J}|}$) for both the axis-aligned and oblique cases.

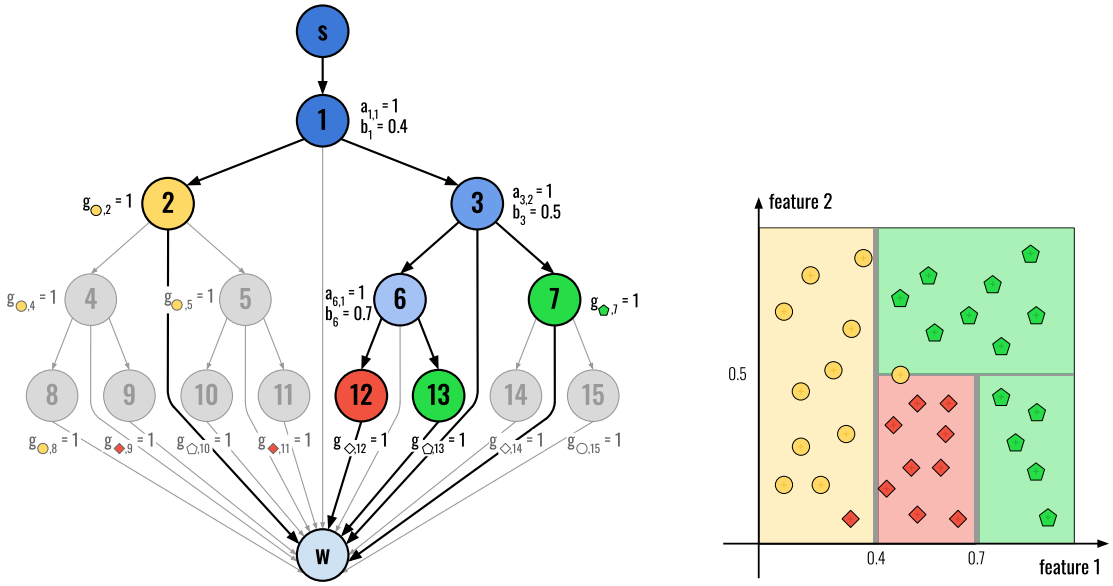
4.1 The axis-aligned case

For our extension, we keep the unitary flow variables $u^i_{t,t'}$ that model if data i is correctly classified and goes through arc (t,t'). Variables $h_{j,t}$ are not relevant anymore, we replace them by variables $a_{j,t}$ and b_t that model the split functions $\sum_{j\in\mathcal{J}} a_{j,t}x_{i,j} \geq b_t$ and $\sum_{j\in\mathcal{J}} a_{j,t}x_{i,j} < b_t$. The obtained formulation $(F_{\alpha,\delta})$ is the following:

$$
(F_{\alpha,\delta})\begin{cases}
\min \quad f^F_{\hat{L},\alpha}(u,a) = \dfrac{1}{\hat{L}}\left(|\mathcal{I}| - \displaystyle\sum_{i\in\mathcal{I}} u^i_{s,r}\right) + \alpha \displaystyle\sum_{t\in\mathcal{N}}\sum_{j\in\mathcal{J}} a_{j,t} & \\
\text{s.t.} \quad (27)-(29),(32),(34),(35) & \\
\displaystyle\sum_{j\in\mathcal{J}} a_{j,t} + \sum_{k\in\mathcal{K}} g_{k,t} = 1 & t\in\mathcal{N} \quad (47)\\
0 \leq b_t \leq \displaystyle\sum_{j\in\mathcal{J}} a_{j,t} & t\in\mathcal{N} \quad (48)\\
\displaystyle\sum_{j\in\mathcal{J}} a_{j,t}\left(x_{i,j} + \mu_j - \mu^-\right) + \mu^- \leq b_t + (1+\mu^+)(1-u^i_{t,l(t)}) & t\in\mathcal{N}, i\in\mathcal{I} \quad (49)\\
\displaystyle\sum_{j\in\mathcal{J}} a_{j,t}x_{i,j} \geq b_t - (1-u^i_{t,r(t)}) & t\in\mathcal{N}, i\in\mathcal{I} \quad (50)\\
u^i_{t,l(t)} \leq \displaystyle\sum_{j\in\mathcal{J}} a_{j,t} & i\in\mathcal{I}, t\in\mathcal{N} \quad (51)\\
u^i_{t,r(t)} \leq \displaystyle\sum_{j\in\mathcal{J}} a_{j,t} & i\in\mathcal{I}, t\in\mathcal{N} \quad (52)\\
a_{j,t} \in \{0,1\} & t\in\mathcal{N}, j\in\mathcal{J} \quad (53)
\end{cases}
$$

We keep in this extension Constraints (27) which force each leaf to be associated to a class, as well as the flow conservation constraints (28)-(29). As in $(F^b_{\delta,\lambda})$, a misclassified data will have a null flow by Constraints (32). Then, we adapt the capacity constraints (30)-(31) with Constraints (51)-(52). Note that Constraints (51) are redundant but are valid inequalities in the linear relaxation. Finally, to model the split functions, we rely on Formulation $(T_{\alpha,\beta,\delta})$ by adapting Constraints (8)-(9) to our case, where the difference is that variables $z_{i,\ell}$ are replaced by variables $u^i_{t,\ell(t)}$ ($u^i_{t,r(t)}$ respectively) in (49) ((50) resp.). We also adapt Constraints (26) into Constraints (47), that ensure that a node either predicts a class or performs a split. In this last case, these constraints additionally force $\sum_{j\in\mathcal{J}} a_{j,t}$ to be equal to 1. For the objective function, we have chosen to use the same parameters as in $(T_{\alpha,\beta,\delta})$, so we minimize a function weighted by parameters $\hat{L}$ and α.

Figure 5a represents an optimal solution of (F), for parameters $\alpha = 0.1$, and $\delta = 4$, on the same training set considered in Figures 2 and 3. Note that the missclassified right-most circle in Figure 5b is located on a split function of thickness μ_2 (i.e. $x_{i,2} \in]0.5-\mu_2, 0.5[$). This is possible in (F) since the flow of any misclassified data $i \in \mathcal{I}$ is null which disables the Constraints (49) and (50). Observe that the same split function is not feasible for (T) since each data is assigned to a path for which the branching rules must be satisfied.



(a) Value of variables a, b and g that are non-zero in the solution.

(b) The training set.

Figure 5: Optimal tree obtained from $(F_{0.1,\,4})$ on a training set containing 3 classes, 2 features, and 34 data.

The following proposition proves that, in the axis-aligned case, the optimal value of formulations $(T_{\alpha,\beta,\delta})$ and $(F_{\alpha,\delta})$ are identical.

Proposition 4. *If* $\beta = 0$, $v(T_{\alpha,\beta,\delta}) = v(F_{\alpha,\delta})$.

Proof. **1.** We first prove that $v(T_{\alpha,\beta,\delta}) \geq v(F_{\alpha,\delta})$. We denote by $Leaves(t)$ the set of leaves that can be reached from $t \in \mathcal{N}$ and by $leaf(t)$ the right-most leaf in $Leaves(t)$. Let $S_T = (a^*, b^*, c^*, d^*, l^*, L^*, N_t^*, N_{k,t}^*, z^*)$ be an optimal solution of $(T_{\alpha,\beta,\delta})$. Without loss of generality, let us assume that $d_t = 1$ if and only if node $t \in \mathcal{N}$ does split data, i.e. if $\sum_{i\in\mathcal{I}} \sum_{\ell\in Leaves(l(t))} z_{i,\ell} \geq 1$ and $\sum_{i\in\mathcal{I}} \sum_{\ell\in Leaves(r(t))} z_{i,\ell} \geq 1$. This is always possible since if there exists a node $t \in \mathcal{N}$ such that either $\sum_{i\in\mathcal{I}} \sum_{\ell\in Leaves(l(t))} z_{i,\ell} = 0$ or $\sum_{i\in\mathcal{I}} \sum_{\ell\in Leaves(r(t))} z_{i,\ell} = 0$ it is possible to set d_t to 0 without altering the predictions of the data. The solution remains optimal as the objective value cannot be increased by this transformation. We build a solution $S_F = (a, b, g, u)$ of $(F_{\alpha,\delta})$ as follows:

- $a_{j,t} = a^*_{j,t}$ and $b_t = b^*_t \quad \forall j \in \mathcal{J}, t \in \mathcal{N}$;
- $u^i_{t,t'}$ is defined differently depending on $(t, t') \in \mathcal{E}$:
  - $u^i_{s,r} = \sum_{\ell\in\mathcal{L}} z^*_{i,\ell} c^*_{y_i,\ell} \quad \forall i \in \mathcal{I}$;
  - $u^i_{\ell,w} = u^i_{a(\ell),\ell} = d^*_{a(\ell)} z^*_{i,\ell} u^i_{s,r} \quad \forall \ell \in \mathcal{L}, i \in \mathcal{I}$;
  - $u^i_{a(t),t} = d^*_{a(t)} u^i_{s,r} \sum_{\ell\in Leaves(t)} z^*_{i,\ell} \quad \forall i \in \mathcal{I}, t \in \mathcal{N} \setminus \{r\}$;
  - $u^i_{t,w} = d^*_{a(t)}(1 - d^*_t) z^*_{i,leaf(t)} u^i_{s,r} \quad \forall i \in \mathcal{I}, t \in \mathcal{N} \setminus \{r\}$;
  - $u^i_{r,w} = (1 - d^*_1) z^*_{i,leaf(r)} u^i_{s,r} \quad \forall i \in \mathcal{I}$;

- $g_{k,t}$ is defined differently depending on $t \in \mathcal{N} \cup \mathcal{L}$:
  - $g_{k,t} = (1 - d_t^*) d_{a(t)}^* c_{k,leaf(t)}^* + (1 - d_{a(t)}^*) \mathbf{1}[k = 1] \quad \forall t \in \mathcal{N} \setminus \{r\}, k \in \mathcal{K}$;
  - $g_{k,r} = (1 - d_1^*) c_{k,leaf(r)}^* \quad \forall k \in \mathcal{K}$;
  - $g_{k,\ell} = d_{a(\ell)}^* c_{k,\ell}^* + (1 - d_{a(\ell)}^*) \mathbf{1}[k = 1] \quad \forall \ell \in \mathcal{L}$.

Let us first note that the integrity of S_T. Moreover, Constraints (7) ensures the integrity of S_F, Constraints (29) are satisfied by definition, and Constraints (1) and (2) imply Constraints (48).

Constraints (49) **and** (50)**.** When $u_{t,l(t)}^i$ is equal to 0, the associated Constraint (49) is necessarily satisfied. From its definition we know that $u_{t,l(t)}^i = 1$ if and only if there exists a leaf $\ell \in Leaves(l(t))$ such that $z_{i,\ell}^* = 1$. In such cases Constraints (8) ensure that Constraints (49) are satisfied. The satisfaction of Constraints (50) can be proved similarly.

Constraints (27)**.** By definition of variables $g_{k,\ell}$, a Constraint (27) is satisfied if $d_{a(\ell)}^* = 0$. Otherwise, we know by assumption that there exists $i \in \mathcal{I}$ such that $z_{i,\ell}^* = 1$. From Constraints (4) and (6), we deduce that $\sum_{k \in \mathcal{K}} c_{k,\ell}^* = 1$ which leads to the same result.

Constraints (28)**.** By definition of variables $u_{t,t'}^i$, a Constraint (28) is satisfied if $d_{a(t)}^* = d_t^*$ or $u_{s,r}^i = 0$. Otherwise, because of Constraints (3), the only possible case is $d_{a(t)}^* = 1$, $d_t^* = 0$ and $u_{s,r}^i = 1$. By definition of variables $u_{t,t'}^i$, the right-hand side of the corresponding Constraint (28) is equal to $z_{i,leaf(t)}^*$ and because of Constraints (1) and (8), $z_{i,leaf(t)}^* = \sum_{\ell \in Leaves(t)} z_{i,\ell}^*$. As the left-hand side of the Constraint(28) is equal to $\sum_{\ell \in Leaves(t)} z_{i,\ell}^*$, by definition of variables $u_{t,t'}^i$, Constraints (28) are satisfied.

Constraints (47)**.** For a given $t \in \mathcal{N}$, if both $d_{a(t)}^*$ and d_t are equal to 1, the associated Constraint (47) is satisfied as $\sum_{k \in \mathcal{K}} g_{k,t} = 0$ by definition of $g_{k,t}$ and $\sum_{j \in \mathcal{J}} a_{j,t} = 1$ from Constraints (4) and (6). Otherwise, either $d_{a(t)}^*$ or d_t^* is equal to 0 and Constraints (1) and (3) ensure that $\sum_{j \in \mathcal{J}} a_{j,t} = 0$. If $d_{a(t)}^* = 0$, $\sum_{k \in \mathcal{K}} g_{k,t} = 1$ by definition of g. If $d_t^* = 0$, $\sum_{k \in \mathcal{K}} g_{k,t} = \sum_{k \in \mathcal{K}} c_{k,leaf(t)}^*$ which is also equal to 1. Thus, Constraints (47) are satisfied in all cases.

Constraints (51) **and** (52)**.** If $d_t^* = 1$, Constraints (51) and (52) are satisfied as Constraints (1) ensure that $\sum_{j \in \mathcal{J}} a_{j,t} = 1$. These constraints are also satisfied if $d_t^* = 0$ as it implies $u_{t,l(t)}^i = u_{t,r(t)}^i = 0$ by definition of these variables.

Constraints (32)**.** For any node $t \in \mathcal{N}$, these constraints are satisfied by definition of $u_{t,w}^i$ if either, $d_{a(t)} = 0$, $d_t = 1$, $z_{i,leaf(t)} = 0$ or $u_{s,r}^i = 0$. Otherwise, we know by definition of $g_{k,t}$ that $g_{k,t} = c_{k,leaf(t)}^*$ and by definition of $u_{s,r}^i$ that there exists $\overline{\ell} \in \mathcal{L}$ such that $z_{i,\overline{\ell}}^* = c_{y_i,\overline{\ell}}^* = 1$. Since $z_{i,leaf(t)} = 1$, we deduce from Constraints (7) that $\overline{\ell} = leaf(t)$. The associated Constraint (32) is also satisfied in that case as this leads to $g_{y_i,t} = 1$. A similar reasoning provides the same result for any $\ell \in \mathcal{L}$.

We now compare the objective values of S_T and S_F. Since the second criterion in both objectives are identical, we only compare the first criterion. For this, we consider the quadratic Equation (37):

$$L_\ell^* = \sum_{k \in \mathcal{K}} c_{k,\ell}^* \sum_{i \in \mathcal{I} \setminus \mathcal{I}_k} z_{i,\ell}^* \quad \forall \ell \in \mathcal{L}$$

The two solutions have the same value since:

$$\sum_{\ell \in \mathcal{L}} L_\ell^* = \sum_{\ell \in \mathcal{L}} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}} z_{i,\ell}^* c_{k,\ell}^* - \sum_{\ell \in \mathcal{L}} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}_k} z_{i,\ell}^* c_{k,\ell}^* = |\mathcal{I}| - \sum_{i \in \mathcal{I}} \sum_{\ell \in \mathcal{L}} z_{i,\ell}^* c_{y_i,\ell}^* = |\mathcal{I}| - \sum_{i \in \mathcal{I}} u_{s,r}^i$$

2. We now prove that $v(T_{\alpha,\beta,\delta}) \leq v(F_{\alpha,\delta})$. Let $S_F = (a^*, b^*, g^*, u^*)$ be an optimal solution of $(F_{\alpha,\delta})$. Without loss of generality, let us assume that $\sum_{j \in \mathcal{J}} a_{j,t}^* = 1$ if and only if node $t \in \mathcal{N}$ does split data. We build $S_T = (a, b, c, d, l, L_t, N_t, N_{k,t}, z)$ a feasible solution to $(T_{\alpha,\beta,\delta})$ with the same objective value:

- $d_t = \prod_{t' \in A(t) \cup \{t\}} \left(\sum_{j' \in \mathcal{J}} a^*_{j,t'} \right)$, $a_{j,t} = d_t a^*_{j,t}$ and $b_t = d_t b^*_t \quad \forall t \in \mathcal{N}, j \in \mathcal{J}$, where $A(t)$ is the set of nodes on the path from r to t (r included);
- $z_{i,\ell} = \prod_{t \in A_L(\ell)} \mathbf{1}[\sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} < b_t] \times \prod_{t \in A_R(\ell)} \mathbf{1}[\sum_{j \in \mathcal{J}} a_{j,t} x_{i,j} \geq b_t] \quad \forall \ell \in \mathcal{L}, i \in \mathcal{I}$;
- $l_\ell = \max_{i \in \mathcal{I}} z_{i,\ell}$, $N_\ell = \sum_{i \in \mathcal{I}} z_{i,\ell}$, and $N_{k,\ell} = \sum_{i \in \mathcal{I}_k} z_{i,\ell} \quad \forall \ell \in \mathcal{L}, k \in \mathcal{K}$;
- $c_{k,\ell} = d_{a(\ell)} g^*_{k,\ell} + \sum_{t \in \mathcal{N}:\ leaf(t) = \ell} B_t g^*_{k,t}, \forall \ell \in \mathcal{L}, k \in \mathcal{K}$ where $B_t = d_{a(t)}(1 - d_t)$, if $t \neq r$ and $B_r = (1 - d_r)$;
- $L_\ell = \sum_{k \in \mathcal{K}} (N_\ell - N_{k,\ell}) c_{k,\ell} \quad \forall \ell \in \mathcal{L}$.

Let us first note that the integrity of S_F ensures the integrity of S_T. Constraints (1), (2), (3), (5), (6), (10), (11) and (14) are satisfied by definition of S_T.

Constraints (4)**.** First note that Constraints (3) ensure that variables d_t are decreasing from the root to the leaf of every branch and that by definition of $c_{k,\ell}$ and Constraints (27), we have $\sum_{k \in \mathcal{K}} c_{k,\ell} = d_{a(\ell)} + \sum_{t \in \mathcal{N}: leaf(t) = \ell} B_t \sum_{k \in \mathcal{K}} g_{k,t}$. We consider three cases according to the possible values of B and d in the sub-branch $SB(\ell) = \{t \in \mathcal{N} \ : \ leaf(t) = \ell\}$ and prove that Constraints (4) are always satisfied:

- *Case 1: $B_t = 0$ for all $t \in SB(\ell)$ and $d_{a(\ell)} = 1$.*
  In that case, $d_t = 1$ for all $t \in SB(\ell)$. Therefore, $\sum_{k \in \mathcal{K}} c_{k,\ell} = 1$. Moreover, $d_{a(\ell)} = 1$ also ensures that $\sum_{j \in \mathcal{J}} a^*_{j,a(\ell)} = 1$ leading with our assumption to the existence of $i \in \mathcal{I}$ such that $z_{i,\ell} = 1$. Consequently, l_ℓ is also equal to 1.
- *Case 2: $B_t = 0$ for all $t \in SB(\ell)$ and $d_{a(\ell)} = 0$.*
  In that case $d_t = 0$ for all $t \in SB(\ell)$ and $r \notin SB(\ell)$. By definition of z and since for all $t \in SB(\ell)$ $d_{a(t)}$ is equal to 0, we obtain $\sum_{i \in \mathcal{I}} z_{i,\ell} = 0$. Therefore $\sum_{k \in \mathcal{K}} c_{k,\ell} = l_\ell = 0$.
- *Case 3: $\exists \bar{t} \in SB(\ell)$ such that $B_{\bar{t}} = 1$.*
  Here $\bar{t}$ is necessarily the first node in sub-branch $SB(\ell)$ such that $d_t = 0$. By definition of a, we have $\sum_{j \in \mathcal{J}} a_{j,\bar{t}} = 0$. Therefore, $\sum_{k \in \mathcal{K}} c_{k,\ell} = \sum_{k \in \mathcal{K}} g_{k,\bar{t}}$ which is equal to 1 from Constraints (47). From the definitions of a, b, z, our assumption and Constraints (49) and (50), we have $l_\ell = 1$.

Constraints (12) **and** (13)**.** Because the solution of $(F_{\alpha,\delta})$ is optimal, a class assigned to a node $t \in \mathcal{N} \cup \mathcal{L}$ is necessarily one of the most represented among the data reaching t. Therefore $c_{k,\ell}$ is equal to $argmax_{k \in \mathcal{K}} \sum_{i \in \mathcal{I}_k} z_{i,\ell}$. By definition of L_ℓ, N_ℓ and $N_{k,\ell}$, Constraints (12) and (13) are satisfied.

Constraints (8) **and** (9)**.** The provided solution may not satisfy all constraints (8) and (9). Indeed, unlike in $(T_{\alpha,\beta,\delta})$, Formulation $(F_{\alpha,\delta})$ does not follow the path of misclassified data. Consequently, there may exist a data $i \in \mathcal{I}$ missclassified by $(F_{\alpha,\delta})$ such that for a given leaf $\ell \in \mathcal{L}$ and a node $t \in A_L(\ell)$, $z_{i,\ell} = 1$ and $x_{i,j} \in]b_t - \mu_j, b_t]$. In that case the associated Constraint (8) is violated. However, it is easy to adjust the value of b_t so that Constraint (8) is satisfied. More precisely, let $j \in \mathcal{J}$ be the feature such that $a_{j,t} = 1$. By construction of μ_j, $x_{i,j}$ is the only value of the dataset for feature j in the interval $[x_{i,j}, x_{i,j} + \mu_j[$. Consequently, the constraint can be satisfied by setting the value of b_t to $x_{i,j} + \mu_j$.

By use of Equation (37) it follows that the two solutions have the same values.

□

We now prove that the continuous relaxation $(\overline{T}_{\alpha,\beta,\delta})$ of $(T_{\alpha,\beta,\delta})$ is worse than that of $(F_{\alpha,\delta})$.

Proposition 5. *If $\alpha > 0$, $\beta > 0$, and $|\mathcal{K}| > 1$, $v(\overline{F}_{\alpha,\delta}) > v(\overline{T}_{\alpha,\beta,\delta})$.*

Proof. By Lemma 1, it amounts to prove that $v(\overline{F}_{\alpha,\delta}) > 0$. Let us distinguish two cases:

Case 1: If $\sum_{(j,t)\in\mathcal{J}\times\mathcal{N}} a_{j,t} > 0$, since $\alpha > 0$, the value of the relaxation is necessarily positive.

Case 2: If $\sum_{(j,t)\in\mathcal{J}\times\mathcal{N}} a_{j,t} = 0$, Constraints (51) and (52) impose that $u^i_{t,t'} = 0$ for any arc (t,t') such that $t \neq s$ and $t' \neq w$. From the flow conservation constraints (28), we deduce that for all $i \in \mathcal{I}$, $u^i_{s,r} = u^i_{r,w}$. Considering Constraints (27) and (32), we have that, for any $(i_1,i_2) \in \mathcal{I}^2$ such that $y_{i_1} \neq y_{i_2}$:

$$u^{i_1}_{s,r} + u^{i_2}_{s,r} = u^{i_1}_{r,w} + u^{i_2}_{r,w} \leq g_{y_{i_1},r} + g_{y_{i_2},r} \leq 1$$

Therefore, $|\mathcal{I}| - \sum_{i\in\mathcal{I}} u^i_{s,r} > 0$.

□

4.2 Extension to the oblique case

We now extend Formulation $(F_{\alpha,\delta})$ to the case of oblique splits. Here again, we rely on Formulation $(T^H_{\alpha,\beta,\delta,\mu})$. In particular, we use variables $\hat{a}_{j,t} = |a_{j,t}|$, binary variables $s_{j,t}$ to count the number of $a_{j,t} \neq 0$, and binary variables d_t that indicate if node t is active, together with constraints (17)-(22) of $(T^H_{\alpha,\beta,\delta,\mu})$. We also use Constraints (27)-(29), (32) of $(F_{\alpha,\delta})$ leading to the following formulation $(F^H_{\alpha,\delta,\mu})$:

$$(F^H_{\alpha,\delta,\mu})\begin{cases}
\min \quad f^{FH}_{\hat{L},\alpha}(u,s) = \dfrac{1}{\hat{L}}\left(|\mathcal{I}| - \sum_{i\in\mathcal{I}} u^i_{s,r}\right) + \alpha \sum_{t\in\mathcal{N}}\sum_{j\in\mathcal{J}} s_{j,t} & \\
\text{s.t.} (17)-(22),(25),(27)-(29),(32),(34),(35) & \\
\quad d_t + \sum_{k\in\mathcal{K}} g_{k,t} = 1 & t\in\mathcal{N} \quad (54)\\
\quad \sum_{j\in\mathcal{J}} a_{j,t}x_{i,j} + \mu \leq b_t + (2+\mu)(1-u^i_{t,l(t)}) & t\in\mathcal{N}, i\in\mathcal{I} \quad (55)\\
\quad \sum_{j\in\mathcal{J}} a_{j,t}x_{i,j} \geq b_t - 2(1-u^i_{t,r(t)}) & t\in\mathcal{N}, i\in\mathcal{I} \quad (56)\\
\quad u^i_{t,l(t)} \leq d_t & i\in\mathcal{I}, t\in\mathcal{N} \quad (57)\\
\quad u^i_{t,r(t)} \leq d_t & i\in\mathcal{I}, t\in\mathcal{N} \quad (58)
\end{cases}$$

In this formulation $\sum_{j\in\mathcal{J}} a_{j,t}$ has no longer a binary value, we thus adapt Constraints (47), (51), (52) into Constraints (54), (57), (58) by use of binary variables d_t. Finally, to model the split functions, we rely on Formulation $(T^H_{\alpha,\beta,\delta,\mu})$ by adapting Constraints (23)-(24). Variables $z_{i,\ell}$ are replaced by variables $u^i_{t,l(t)}$ in (55) and by variables $u^i_{t,r(t)}$ in (56).For the objective function, we have chosen to use the same parameters as in $(T^H_{\alpha,\beta,\delta,\mu})$, so we minimize a function weighted by parameters $\hat{L}$ and α.

Proposition 6. *For $\delta > 0$, $\beta = 0$ and $\mu > 0$, $v(F^H_{\alpha,\delta,\mu}) \leq v(T^H_{\alpha,\beta,\delta,\mu})$. There exists datasets for which this inequality is strict if and only if $\alpha < 1$.*

Proof. Let $(a^*, \hat{a}^*, s^*, b^*, c^*, d^*, l^*, L^*, N_t^*, N_{k,t}^*, z^*)$ be an optimal solution of $(T^H_{\alpha,\beta,\delta,\mu})$. Without loss of generality, let us assume that $d_t = 1$ if and only if node $t \in \mathcal{N}$ does split data. We build $(a, \hat{a}, s, b, d, u, g)$ a feasible solution of $(F^H_{\alpha,\delta,\mu})$:

- $a_{j,t} = a^*_{j,t}$, $\hat{a}_{j,t} = \hat{a}^*_{j,t}$, $s_{j,t} = \tilde{s}_{j,t} \quad \forall j \in \mathcal{J}, t \in \mathcal{N}$;
- $b_t = b^*_t$, $d_t = d^*_t \quad \forall t \in \mathcal{N}$;
- $u^i_{t,t'}$ is defined differently depending on $(t, t') \in \mathcal{E}$:
  - $u^i_{s,r} = \sum_{\ell \in \mathcal{L}} z^*_{i,\ell} c^*_{y_i,\ell} \quad \forall i \in \mathcal{I}$;
  - $u^i_{\ell,w} = u^i_{a(\ell),\ell} = d^*_{a(\ell)} z^*_{i,\ell} u^i_{s,r} \quad \forall \ell \in \mathcal{L}, i \in \mathcal{I}$;
  - $u^i_{a(t),t} = d^*_{a(t)} u^i_{s,r} \sum_{\ell \in Leaves(t)} z^*_{i,\ell} \quad \forall i \in \mathcal{I}, t \in \mathcal{N} \setminus \{r\}$;
  - $u^i_{t,w} = d^*_{a(t)}(1 - d^*_t) z^*_{i,leaf(t)} u^i_{s,r} \quad \forall i \in \mathcal{I}, t \in \mathcal{N} \setminus \{r\}$;
  - $u^i_{r,w} = (1 - d^*_1) z^*_{i,leaf(r)} u^i_{s,r} \quad \forall i \in \mathcal{I}$;
- $g_{k,t}$ is defined differently depending on $t \in \mathcal{N} \cup \mathcal{L}$:
  - $g_{k,t} = (1 - d^*_t) d^*_{a(t)} c^*_{k,leaf(t)} + (1 - d^*_{a(t)}) \mathbf{1}[k = 1] \quad \forall t \in \mathcal{N} \setminus \{r\}, k \in \mathcal{K}$;
  - $g_{k,r} = (1 - d^*_1) c^*_{k,leaf(r)} \quad \forall k \in \mathcal{K}$;
  - $g_{k,\ell} = d^*_{a(\ell)} c^*_{k,\ell} + (1 - d^*_{a(\ell)}) \mathbf{1}[k = 1] \quad \forall \ell \in \mathcal{L}$.

With an very similar reasoning as used in the proof of Proposition 4, one can check that this solution satisfies all constraints of $(F^H_{\alpha,\delta,\mu})$ and that the two solutions have the same value. Consequently, $v(F^H_{\alpha,\delta,\mu}) \leq v(T^H_{\alpha,\beta,\delta,\mu})$.

If $\alpha < 1$, we now exhibit datasets for which $v(F^H_{\alpha,\delta,\mu}) < v(T^H_{\alpha,\beta,\delta,\mu})$. We consider a training set $\mathcal{I} = \{(X_{i_1}, k_1), (X_{i_2}, k_2), (X_{i_3}, k_3), (X_{i_4}, k_3)\}$ composed of three classes $\mathcal{K} = \{k_1, k_2, k_3\}$ such that:

- $x_{i_1,1} = 0$, $x_{i_2,1} = \frac{\overline{\mu}}{2}$, $x_{i_3,1} = \overline{\mu}$, $x_{i_4,1} = 1$;
- $x_{i_1,j} = x_{i_2,j} = x_{i_3,j} = x_{i_4,j} \quad \forall j \in \mathcal{J} \setminus \{1\}$.

We prove that data i_1 and i_3 necessarily reach the same leaf of a tree obtained by $(T^H_{\alpha,\beta,\delta,\mu})$ while they can be separated by $(F^H_{\alpha,\delta,\mu})$.

Let t be the highest node of a decision tree which split function separates i_1 and i_3. Let us assume without loss of generality that $\sum_{j \in \mathcal{J}} a_{j,t} x_{i_1,j}$ is lower than $\sum_{j \in \mathcal{J}} a_{j,t} x_{i_3,j}$. Consequently,

$$\begin{array}{ccc} \sum_{j \in \mathcal{J}} a_{j,t} x_{i_1,j} < & b_t & \leq \sum_{j \in \mathcal{J}} a_{j,t} x_{i_3,j} \\ & 0 < b_t - \sum_{j \in \mathcal{J} \setminus \{1\}} a_{j,t} x_{i_1,j} \leq a_{1,t} \overline{\mu} & \end{array} \tag{59}$$

In $(T^H_{\alpha,\beta,\delta,\mu})$, the Constraint (23) associated with node t and the leaf reached by data i_1 leads to

$$\overline{\mu} \leq b_t - \sum_{j \in \mathcal{J} \setminus \{1\}} a_{j,t} x_{i_1,j} \tag{60}$$

From inequalities (59) and (60) we deduce that

$$b_t - \sum_{j \in \mathcal{J} \setminus \{1\}} a_{j,t} x_{i_1,j} = \overline{\mu} \tag{61}$$

Given the features vectors of the data and the fact that t is the highest node which separates nodes i_1 and i_3, we deduce that data i_2 necessarily reaches node t. Consequently, i_2 either reaches the left or the right child of t. We prove that in both cases Equation (61) can not be satisfied, thus proving that i_1 and i_3 can not be separated in $(T^H_{\alpha,\beta,\delta,\mu})$.

If i_2 reaches $l(t)$, we deduce from the Constraint (23) associated with node t and the leaf reached by i_2 that:

$$\begin{aligned} \textstyle\sum_{j \in \mathcal{J}} a_{j,t} + \overline{\mu} &< b_t \\ a_{1,t} \tfrac{\overline{\mu}}{2} + \overline{\mu} &< \overline{\mu} \\ a_{1,t} \overline{\mu} &< 0 \end{aligned}$$

which is not possible as $a_{1,t}$ must be greater than 0 according to (59).

If i_2 reaches $r(t)$:

$$\begin{aligned} \textstyle\sum_{j \in \mathcal{J}} a_{j,t} x_{i_2,j} &\geq b_t \\ a_{1,t} \tfrac{\overline{\mu}}{2} &\geq \overline{\mu} \\ a_{1,t} &\geq 2 \end{aligned}$$

which is not either possible as $a_{1,t} \leq 1$.

Since the distances between i_1 and i_2 or i_2 and i_3 are both lower than the one between i_1 and i_3, $(T^H_{\alpha,\beta,\delta,\mu})$ can not either separate these data. Thus, all three data necessarily reach the same leaf of a tree obtained by $(T^H_{\alpha,\beta,\delta,\mu})$ which leads to at least 2 misclassifications.

However, i_1 and i_3 can be separated by a tree obtained with $(F^H_{\alpha,\delta,\mu})$. Indeed, if i_2 is missclassified, its flow is equal to 0 and all Constraints (49) and (50) involving this data are satisfied. A tree with a single split function $x_{1,t} \leq \overline{\mu}$ can, thus, be obtained by $(F^H_{\alpha,\delta,\mu})$ and leads to only one classification error. Consequently, if $\alpha = 0$, $v(F^H_{\alpha,\delta,\mu}) < v(T^H_{\alpha,\beta,\delta,\mu})$.

If α is in $]0, 1[$. Let T_r be the tree reduced to a root node which predicts the most represented class $\overline{k}$ in the training set (i.e., $\overline{k} = \arg\max_{k \in \mathcal{K}} |\mathcal{I}_k|$). The same reasoning applies for this value of α provided that we can ensure that T_r is not an optimal tree (otherwise $v(F^H_{\alpha,\delta,\mu}) = v(T^H_{\alpha,\beta,\delta,\mu}) = 1$). This can be obtained by adding to the dataset a sufficient number of data equal to (X_{i_1}, k_1) and (X_{i_4}, k_3).

If $\alpha \geq 1$, T_r is an optimal tree. Consequently, $v(F^H_{\alpha,\delta,\mu}) = v(T^H_{\alpha,\beta,\delta,\mu})$. □

In terms of value of the linear relaxation, we obtain a result similar to the one of the axis-aligned case.

Proposition 7. *If $\alpha > 0$, $\beta = 0$ and $|\mathcal{K}| > 1$, $v(\overline{F}^H_{\alpha,\delta,\mu}) > v(\overline{T}^H_{\alpha,\beta,\delta,\mu})$.*

Proof. By Lemma 2, it amounts to prove that $(\overline{F}^H_{\alpha,\delta,\mu}) > 0$. Let us distinguish two cases.

Case 1: If $\sum_{(t,j) \in \mathcal{N} \times \mathcal{J}} s_{j,t} > 0$, since α is positive, the value of the relaxation is necessarily positive.

Case 2: If $\sum_{(t,j)\in\mathcal{N}\times\mathcal{J}} s_{j,t} = 0$, the proof is the same in Case 2 in the proof of Proposition 5.

□

5 From MIP solution to classification tree

The models presented throughout this paper have a large set of optimal solutions and depend on several parameters (the depth of the tree δ, the minimal number of data in a leaf β, and the weight α of the second objective). In this section we present a post-processing method that selects a "good" optimal for a given value of the parameters (Section 5.1) and an algorithm that fits the parameters to a specific dataset (Section 5.2).

5.1 Post-processing for better performances

All the considered models have an infinite number of optimal solutions since the left-hand side b_t of any split is continuous. Consequently, b_t can take an infinite number of values without altering the path of the data. We observed experimentally that the optimal solutions often provide split functions which are very close to the data. However, similarly to the well-known SVM models (Vapnik, 1963), we want to maximize these distances to reduce the risk of having test data of a same class on both sides of a split function. For this, we propose a post-processing algorithm in which the splits of a given tree are shifted. The post-processing solves an optimization problem similar to that of Zhou et al. (W. Zhou, Zhang, & Jiao, 2002) introduced for linear SVM problems.

The post-processing is applied independently on each splitting node $t \in \mathcal{N}$. Let $\mathcal{I}_L$ ($\mathcal{I}_R$ resp.) be the set of data reaching the left (right resp.) child of node t. The point is to determine the split between $\mathcal{I}_L$ and $\mathcal{I}_R$ which is the furthest from all data in $\mathcal{I}_L \cup \mathcal{I}_R$. Note that in flow based formulations, we do not take into account missclassified for the post-processing since they are not used by the formulation to fix the coefficients of the split functions.

In the oblique case, the new split function of node t is obtained by solving a MILP. For all $j \in \mathcal{J}$, let $s^*_{j,t}$ be the value of variable $s_{j,t}$ before the post-processing. Variables $a_{j,t}$, b_t and $s_{j,t}$ characterise the shifted split function. Variable e_i is equal to the distance between data $i \in \mathcal{I}$ and the new split function and e_{min} is equal to $\min_{i\in\mathcal{I}} e_i$.

$$
(\mathrm{A})\begin{cases}
\max\limits_{a_{j,t},b_t,s_{j,t},e_i,e_{min}} & e_{min} & \\
\text{s.t.} & e_i \geq e_{min} & i \in \mathcal{I}_L \cup \mathcal{I}_R (62)\\
& e_i = b_t - \sum_{j\in\mathcal{J}} a_{j,t}x_{i,j} & i \in \mathcal{I}_L (63)\\
& e_i = \sum_{j\in\mathcal{J}} a_{j,t}x_{i,j} - b & i \in \mathcal{I}_R (64)\\
& -s_{j,t} \leq a_{j,t} \leq s_{j,t} & j \in \mathcal{J} (65)\\
& \sum\limits_{j\in\mathcal{J}} s_{j,t} \leq \sum\limits_{j\in\mathcal{J}} s^*_{j,t} & (66)\\
& b \in [-1,1] & (67)\\
& s_{j,t} \in \{0,1\} & j \in \mathcal{J} (68)
\end{cases}
$$

Constraints (62) ensure $e_{min} = \min_i e_i$. Constraints (63) ((64) resp.) set the value of e_i for data points going through the left branch (right branch resp.). Constraints (65) define variables $s_{j,t}$ as

indicators of whether $a_{j,t}$ is null or not. Constraint (66) ensures that the split keeps the same number of non-zero coefficients. Constraints (65) and (67) define respectively the domain of b_t and $s_{j,t}$. This approach is fast as it only requires to solve for each node $t \in \mathcal{N}$ a MILP with $|\mathcal{J}|$ integer variables.

In the case of axis-aligned splits, the new split function can be obtained more easily. The values of variables $a_{j,t}$ are not modified. It remains to determine the value of b_t that maximizes the distance to the closest data in $\mathcal{I}_R$, and $\mathcal{I}_L$. Let $j^* \in J$ be the feature on which the separation is performed in node t, b_t^m be the highest value of feature j^* in $\mathcal{I}_L$ and b_t^M be the lowest value of feature j^* in $\mathcal{I}_R$ (i.e., $b_t^m = \max_{i \in \mathcal{I}_L} x_{i,j^*}$ and $b_t^M = \min_{i \in \mathcal{I}_R} x_{i,j^*}$). Any value of b_t in $[b_t^m, b_t^M[$ could be selected without altering the path of the data. In order to maximize the minimal distance between the new split function and the data, b_t is set to $\frac{b_t^m + b_t^M}{2}$.

Algorithm 1 corresponds to the post-processing applied to a solution Sol.

Algorithm *Post-processing(Sol)*
- $T \leftarrow T^*$ (where T^* is the tree given by Sol)
- **for** $t \in \mathcal{N}$ **do**
  - **if** t *is applying a split* **then**
    - Compute $\mathcal{I}_L$ ($\mathcal{I}_R$ resp.) subset of data points going through the left branch (right branch resp.) of t
    - Coefficient $(a_{j,t})_{j \in \mathcal{J}}, b_t$ of $T \leftarrow$ values from resolution of (A)
- **return** T

Algorithm 1: Post-processing algorithm

5.2 Fitting parameters

In (Bertsimas & Dunn, 2017), Bertsimas and al. consider Algorithm 2 to select the value of parameters α and δ for a given dataset. In this algorithm, the data are split in three sets: the training set, the test set and the validation set. The algorithm iteratively creates trees which are optimal for the train set, and selects the best one for the validation set. Eventually, the performance of the selected tree is assessed over the test set. Since α is a continuous parameter, the algorithm can not iterate on all its possible values. To overcome this, the authors remove the second objective and add a new constraint that bounds the number of splits allowed in the tree by $C \in \mathbb{Z}^+$:

$$\sum_{j,t \in \mathcal{J} \times \mathcal{N}} a_{j,t} \leq C \tag{69}$$

in the axis-aligned case, and

$$\sum_{j,t \in \mathcal{J} \times \mathcal{N}} s_{j,t} \leq C \tag{70}$$

in the oblique case.

Algorithm 2 $TreeTraining(M, \delta_{MAX}, \beta)$ iterates by varying the values of C and δ within their respective ranges. The input M is the considered formulation, $\overline{\delta}$ is the maximal depth, and β is the minimal number of data points per open leaf. Parameter $\overline{C}_\delta$ is equal to $2^\delta - 1$ in the axis-aligned case, and $(2^\delta - 1)|\mathcal{J}|$ in the oblique case. Note that the value of β is set to $0.05 \times |\mathcal{I}|$ in (Bertsimas & Dunn, 2017), but its value could also be fitted by adding a loop on its value.

Algorithm *TreeTraining(M,$\overline{\delta}$,β)*
- $\mathcal{S} \leftarrow \emptyset$
- **for** $\delta = 1$ *to* $\overline{\delta}$ **do**
  - **for** $C = 1$ *to* $\overline{C}_\delta$ **do**
    - $\hat{T} \leftarrow$ a feasible solution for warm starting (given by CART, or from set $\mathcal{S}$).
    - $T \leftarrow$ an optimal solution of M with $\alpha = 0$ and (69) (axis-aligned), or (70) (oblique) using $\hat{T}$.
    - $\mathcal{S} \leftarrow \mathcal{S} \cup \{T\}$
- Remove within $\mathcal{S}$ all dominated solutions for M.
- $T^* \leftarrow$ the tree in $\mathcal{S}$ that best performs on the validation set.
- **return** T^*

Algorithm 2: Algorithm of (Bertsimas & Dunn, 2017) to train a tree

Algorithm 2 determines relevant values of the parameters, at the cost of a heavy computational time, since each iteration requires the resolution of a MIO. This is especially true for oblique splits as the range of C is proportional to $|\mathcal{J}|$. We design Algorithm 3 which enables to reduce the number of iterations. Since the number of misclassifications is a decreasing piecewise constant function of C, instead of iterating on every possible pair (δ, C), the loop on the value of C may skip some values while leading to the same solutions than Algorithm 1. To do so, we keep the second objective and we set α to a value $\underline{\alpha}$ small enough to prioritise the first objective. Consequently, among all the solutions with an optimal number of misclassifications, we obtain one for which the number of splits is minimal. Thus, at iteration C, the number of splits $\hat{C}$ of the solution may be lower than C, and therefore iterations $\{C-1, C-2, ..., \hat{C}\}$ can be skipped.

Since the loop on C is backwards, unlike Algorithm 2, the tree obtained at the previous iteration can not directly be used as a warm start. To make this tree feasible, we set to 0 its coefficient a that least increases the number of missclassifications.

Algorithm *CompactTreeTraining(M,$\overline{\delta}$,β)*
- $\mathcal{S} \leftarrow \emptyset$
- **for** $\delta = 1$ *to* $\overline{\delta}$ **do**
  - $C \leftarrow \overline{C}_\delta$
  - **while** $C \geq \delta$ **do**
    - $\hat{T} \leftarrow$ a feasible solution for warm starting (given by CART, from set $\mathcal{S}$, or from T obtained at the previous step).
    - $T \leftarrow$ an optimal solution of M with $\alpha = \underline{\alpha}$, (69) (axis-aligned), or (70) (oblique) and using $\hat{T}$.
    - $\hat{C} \leftarrow \sum_{j,t \in \mathcal{J} \times \mathcal{N}} s_{j,t}$ (axis-aligned), or $\sum_{j,t \in \mathcal{J} \times \mathcal{N}} a_{j,t}$ (oblique)
    - Algorithm 1 is applied to T
    - $\mathcal{S} \leftarrow \mathcal{S} \cup \{T\}$
    - $C \leftarrow \hat{C} - 1$
- Remove within $\mathcal{S}$ all dominated solutions for M.
- $T^* \leftarrow$ the tree in $\mathcal{S}$ that best performs on the validation set.
- **return** T^*

Algorithm 3: Training a tree

Note that in Section 6.2.2. we discuss the use of Algorithm 1, therefore we do not always apply the post-proccesing in Algorithm 3.

6 Computational results

In this section, we evaluate numerically all the formulations considered in Sections 2, 3, 4 and 5 summarized in Table 1. We first identify the most efficient ones and then evaluate the learning performances of our new models through Algorithm 3.

Axis-aligned splits		Oblique splits	
Formulation	Type of program	Formulation	Type of program
(T)	PLNE	(T^H)	PLNE
(Q)	QP	(Q^H)	QP
(QF)	PLNE	(QF^H)	PLNE
(QG)	PLNE	(QG^H)	PLNE
(F)	PLNE	(F^H)	PLNE

Table 1: Summary of all formulations

To carry out our tests, datasets were selected from the UCI Repository (Dua & Graff, 2017). Their characteristics are specified in Table 2.

Datasets	$\overline{\mathcal{I}}$	$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{K}\rvert$
Iris	150	4	3
Wine	178	13	3
Dermatology	366	34	6
Breast Cancer	699	9	2
Blood Trans	748	4	2

Table 2: Datasets used for tests

Experimental environment The experiments were performed on a server with 2 Intel Xeon CPUs each with 16 cores and 32 threads of 2.3 GHz and $8 * 16$ GB of RAM running the Linux OS. The experiments were implemented in Julia and C++ and we used the solver Gurobi 9.1.1 (?, ?) for solving the formulations of Table 1.

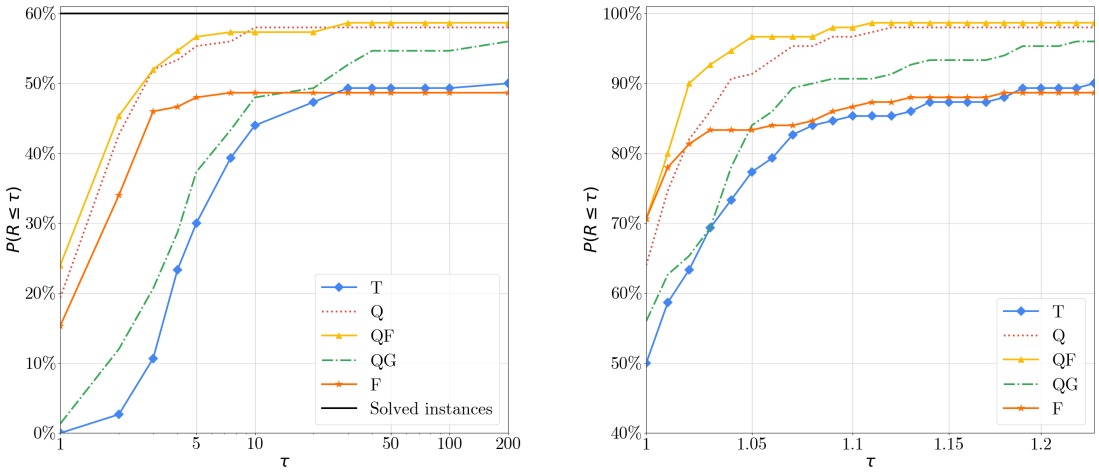
6.1 Comparison of the CPU times and final gaps of the formulations

In order to evaluate the quality of the new formulations from the CPU time point of view, we start by performing tests with fixed parameters. Experiments were carried out on 10 different training sets for each dataset of Table 2. The training sets were a random selection of 80% of the original dataset. We used the following set of paramaters:

- the arbitrary value 0.2 for α,
- depths $\delta \in \{2, 3, 4\}$,
- $\beta = 0$, enable a comparison of flow-based formulations (F) and (F^H) (that do not have a parameter β) with the formulations based on (T),
- for oblique formulations, $\mu = 10^{-4}$.

We set the time limit to one hour.

We present in Figures 6–7 the performance profiles of the considered formulations on 2 criteria: CPU times and final gaps. A performance profile plots one curve for each formulation. Each point of a curve gives, for a given factor τ, the percentage of instances whose criterion was at most τ times greater than the minimal value. In particular, for $\tau = 1$, we have the proportion of instances on which the formulation was the best on the criterion.



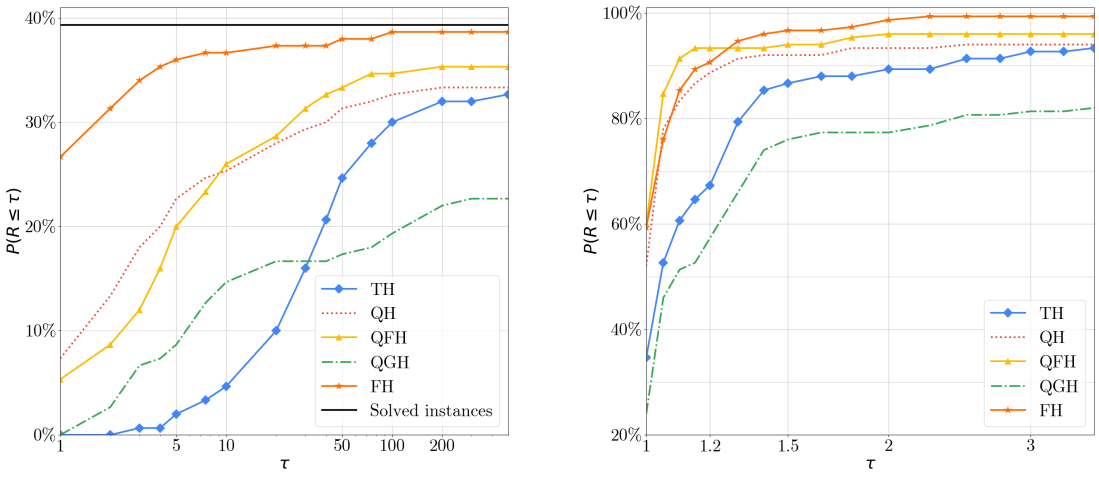
(a) Performance Profile of CPU times.

(b) Performance Profile of final gap.

Figure 6: Performance Profiles in axis-aligned case - All datasets - Time limit : 1 h

Figure 6a presents the performance profile of CPU times for the axis-aligned case. We observe that that (Q) and (QF) have similar CPU times, which indicates that Gurobi probably solves (Q) through a Fortet linearization. We also see that (QG) is significantly slower than (QF). Moreover, (F) is efficient on all datasets except for the blood transfusion one, for which it is not able to solve any instance. Lastly, (T) is clearly slower than (Q), (QF), and (QG).

Figure 6b shows the performance profile of final gaps. Here, by final gap we mean $\frac{UB-LB}{UB} * 100$, where UB (LB resp.) is the best feasible solution (lower bound resp.) found by the formulation within the time limit. For $\tau = 1$ we have the percentage of instances for which the formulation had the smallest gap which includes the percentage of instances solved optimally. The percentage given by the maximum value of τ corresponds to all the instances for which the formulation found a solution within the time limit and the proportion of instances to which no formulation found a solution. This highlights the fact that (T) and (F) were the more likely formulations not to find a solution within the time limit while others did. However we see a difference in the curves of (T) and (F): (T) is, for most values of τ under (F) meaning that when both formulation do not find an optimal solution, (F) is more likely to have a smaller gap.



(a) Performance Profile of CPU times.

(b) Performance Profile of final gap.

Figure 7: Performance Profiles in oblique case - All datasets - Time limit : 1 h

We present in Figure 7a the performance profile of CPU times for the oblique case. The results reveal a similar trend for formulations (Q^H) and (QF^H). However, (F^H) is the fastest and solves much more instances than the other formulations within the time limit. Formulations (QF^H) and (Q^H) are still faster than (T^H), but they solve almost the same number of instances within one hour. Finally, (QG^H) is the slowest formulation and it solves less than half of the considered instances.

To complete the evaluation of the quality of the oblique formulations, we present in Figure 7b the performance profile of the final gaps. We see that (QG^H) is definitely the worst formulation since it is the most likely not to find a solution while others do. We also see that even if (T^H) solves the same amount of solution as other formulations (except (QG^H)), its gaps tends to be higher than the others (except (QG^H)).

We make the following observations:

- Performances of (Q) and (QF) are very similar, it seems the solver, Gurobi, systematically uses the Fortet linearization for our instances;
- (QG) is not a clear improvement over (T) unlike (QF);
- (Q), (QF) and (F) are a clear improvement over (T) in terms of CPU times.

Moving forward, we do not use formulations (Q) and (QG) since (Q) is extremely similar to (QF) and (QG) is the least efficient of all the models.

6.2 Impact of fitting algorithms

We present here results regarding the fitting algorithm i.e. Algorithms 2 and 3. We use Algorithm 2 with formulations (T) and (T^H) and Algorithm 3 with formulations (QF), (QF^H), (F) and (F^H).

As in Bertsimas and al. (Bertsimas & Dunn, 2017), tests were carried out on 5 different partition of the considered datasets. Each dataset was divided in 3 parts: the training set (50% of the original

dataset), the validation set (25%) and the test set (25%). Each formulation had a time limit of 30 minutes for the axis-aligned case and of 5 minutes for the oblique case. The maximum depth is 4 and parameter β is fixed at $5\%|\mathcal{I}|$.

6.2.1 CPU Time

Table 3 presents the CPU times for each dataset and formulations averaged on the 5 different partitions considered. The best formulation is in bold while, for the CPU Time column, the worst is in gray. We observe that (T) is very often the worst and that (QF) is the best in terms of CPU time. The CPU time difference between Algorithm 3 with new formulations and Algorithm 2 with (T) is significant. The reduction even reaches 95% for the blood trans dataset in the oblique case.

Dataset	Splits	CPU Time (in seconds) Algorithm 2 (T)	CPU Time (in seconds) Algorithm 3 (QF)	CPU Time (in seconds) Algorithm 3 (F)	Ratio of the CPU Time to the CPU Time of (T) Algorithm 3 (QF)	Ratio of the CPU Time to the CPU Time of (T) Algorithm 3 (F)
Iris	Axis-aligned	4741	781	**224**	0.8	**0.3**
Iris	Oblique	1000	285	**255**	1.22	**0.76**
Wine	Axis-aligned	80	**46**	197	**0.61**	2.24
Wine	Oblique	1183	**141**	266	**0.29**	0.5
Blood Tr.	Axis-aligned	29414	**4027**	13871	**0.14**	0.47
Blood Tr.	Oblique	64697	**3078**	4192	**0.05**	0.06
Breast C.	Axis-aligned	30309	**6284**	19220	**0.21**	0.63
Breats C.	Oblique	28259	**3083**	5017	**0.11**	0.18

Table 3: Average CPU time of learning algorithms - Maximum depth of tree $\overline{\beta} = 4$

In Table 4, we highlight that both the new algorithm and the new formulations contribute to the improvement of the CPU times. We present two different indicators: the average number of MIPs (i.e. the number of iterations in the loop of Algorithm 3 that were not skipped) and the average number of MIPs that were solved to optimality (i.e. whose resolution time was inferior to 30 minutes for the axis-aligned case and 5 minutes for the oblique case). In parenthesis we represent the percentage of MIPs solved optimally.

Dataset	Splits	Nb. of MIPs computed Algorithm 2 (T)	Nb. of MIPs computed Algorithm 3 (QF)	Nb. of MIPs computed Algorithm 3 (F)	Nb. of MIPs solved optimally Algorithm 2 (T)		Nb. of MIPs solved optimally Algorithm 3 (QF)		Nb. of MIPs solved optimally Algorithm 3 (F)	
Iris	Axis-aligned	20	5	5	18	(89%)	4	(96%)	5	(100%)
Iris	Oblique	98	6	6	96	(98%)	6	(95%)	6	(93%)
Wine	Axis-aligned	20	5	5	20	(100%)	5	(100%)	5	(100%)
Wine	Oblique	332	7	7	331	(100%)	7	(100%)	7	(96%)
Blood Tr.	Axis-aligned	20	5	13	4	(20%)	2	(43%)	3	(20%)
Blood Tr.	Oblique	98	11	17	4	(4%)	1	(9%)	1	(8%)
Breast C.	Axis-aligned	20	4	12	4	(21%)	2	(48%)	5	(41%)
Breats C.	Oblique	228	15	19	14	(6%)	5	(32%)	5	(28%)

Table 4: Number of MIP considered and optimally solved by Algorithms 2 and 3

We can see in Table 4 that a significant amount of iterations were skipped with Algorithm 3 in comparison to Algorithm 2. But we can also see that there is a greater proportion of MIPs solved to

optimality by the new formulations e.g. for the breast cancer dataset in the oblique case, (T^H) solved optimally about 3 times more MIPs but it had to solve 12 to 15 times more MIPs, therefore (QF) and (F) solve a greater proportion of the computed MIPs to optimality. This shows that the new formulations also contribute to the decrease in the time it takes to build a tree.

6.2.2 Impact of the post-processing

In Algorithm 3 the post-processing is applied at each iteration. To assess its impact, we consider two variants. In the first variant the post-processing is never applied, while in the second variant, the post-processing is applied but the post-processed tree is only kept if its performances on the validation set are at least as good as the one of the original tree.

We apply these variants on all datasets, partitions of the data and formulations (QF), (QF^H), (F), and (F^H). In 65% of the cases, the three approaches lead to the same results. Algorithm 3 is the best in 24% of the cases while the first variant is only the best in 11% of the cases. The second variant enables to reach 25% which is a slight improvement. In this variant, the post-processed tree is chosen in 84% of the cases. It improves the results in 22% of the cases and deteriorate them in 6%.

In the following, the results presented for Algorithm 3 corresponds to this second variant which appears to be more efficient.

6.2.3 Performances on test sets

Table 5 shows the average percentage of error on test sets for Algorithm 2 with Formulation (T), Algorithm 3 with Formulation (QF) and Algorithm 3 with Formulation (F). The best performing algorithm is in bold and the worst in gray.

Dataset	Split	Mean error (in %) Algorithm 2 with (T)	Algorithm 3 with (QF)	Algorithm 3 with (F)
Iris	Axis-aligned	**5.79**	**5.79**	6.32
Iris	Oblique	7.37	**5.79**	**5.79**
Wine	Axis-aligned	9.33	**7.100**	8.44
Wine	Oblique	12.89	10.67	**9.78**
Blood Transfusion	Axis-aligned	20.21	19.57	**19.14**
Blood Transfusion	Oblique	**19.57**	19.68	19.79
Breast Cancer	Axis-aligned	**5.38**	5.61	5.73
Breast Cancer	Oblique	5.15	5.38	**4.44**

Table 5: Mean percentage of error on test sets for Algorithms 2 and 3

We see that when Algorithm 2 with Formulation (T) is the best, Algorithm 3 with both formulations is often less than 1% higher. However, when Algorithm 2 with formulation (T) is the worst, Algorithm 3 is often more than 1% lower. That is to say Algorithm 3 with (QF) or (F) has similar or better performances on the test set.

7 Conclusion

We introduce four new formulations, each with two variants depending on whether the splits are axis-aligned or oblique. The first formulation is a quadratic formulation of (T), the model introduced by

Bertsimas et al. (Bertsimas & Dunn, 2017), that we linearize with Fortet and Glover linearizations. The last formulation is an extension to real-valued data of a flow-based formulation (Aghaei et al., 2020). We prove that these new formulations have a better continuous relaxation than (T) in the axis-aligned case and than (T^H) in the oblique case. We also highlight that for some instances the flow-based formulations have better performances.

We present a more efficient version of the algorithm introduced by Bertsimas et al. (Bertsimas & Dunn, 2017) to fit parameters. To enhance the performances on the test sets, we design a post-processing algorithm that shifts the split functions of a tree as far as possible from the data.

Our numerical experiments show that both the Fortet linearization of the quadratic formulation of (T) and the flow-based formulation are faster than (T). Using those formulations together with our parameter fitting algorithm, we significantly reduce the resolution time while maintaining or even improving the performance on the test set.

In future work we will focus on further reducing the resolution time to enable the exact resolution of larger datasets.

References

Aghaei, S., Gomez, A., & Vayanos, P. (2020). Learning optimal classification trees: Strong max-flow formulations. *arXiv preprint arXiv:2002.09142*.

Bertsimas, D., & Dunn, J. (2017). Optimal classification trees. *Machine Learning*, *106*(7), 1039–1082.

Blanquero, R., Carrizosa, E., Molero-Río, C., & Morales, D. R. (2020). Sparsity in optimal randomized classification trees. *European Journal of Operational Research*, *284*(1), 255–272.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Classification and regression trees.

Carvalho, D. V., Pereira, E. M., & Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, *8*(8), 832.

Demirović, E., Lukina, A., Hebrard, E., Chan, J., Bailey, J., Leckie, C., ... Stuckey, P. J. (2022). Murtree: Optimal decision trees via dynamic programming and search. *Journal of Machine Learning Research*, *23*(26), 1–47.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Dua, D., & Graff, C. (2017). *UCI machine learning repository.* Retrieved from `http://archive.ics.uci.edu/ml`

Fortet, R. (1959). L'algèbre de Boole et ses applications en recherche opérationnelle. *Cahiers du Centre d'Études de Recherche Opérationnelle*, *4*, 5–36.

Glover, F. (1975). Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, *22*, 455-460.

Goodman, B., & Flaxman, S. (2017, Oct.). European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine*, *38*(3), 50-57. Retrieved from `https://ojs.aaai.org/index.php/aimagazine/article/view/2741` doi: 10.1609/aimag.v38i3.2741

Jost, L. (2006). Entropy and diversity. *Oikos*, *113*(2), 363–375.

Laurent, H., & Rivest, R. L. (1976). Constructing optimal binary decision trees is np-complete. *Information processing letters*, *5*(1), 15–17.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 1135–1144).

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, *1*(5), 206–215.

Vapnik, V. (1963). Pattern recognition using generalized portrait method. *Automation and remote control*, *24*, 774–780.

Verwer, S., & Zhang, Y. (2017). 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* (pp. 94–103).

Wachter, S., Mittelstadt, B., & Floridi, L. (2017). Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *International Data Privacy Law*, *7*(2), 76–99.

Wachter, S., Mittelstadt, B., & Russell, C. (2017). Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, *31*, 841.

Zhou, J., Gandomi, A. H., Chen, F., & Holzinger, A. (2021). Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, *10*(5), 593.

Zhou, W., Zhang, L., & Jiao, L. (2002). Linear programming support vector machines. *Pattern recognition*, *35*(12), 2927–2936.