Advances in Data Analysis and Classification
https://doi.org/10.1007/s11634-021-00467-2

REGULAR ARTICLE

Robust optimal classification trees under noisy labels

Victor Blanco[1] · Alberto Japón[2] · Justo Puerto[2]

Received: 15 December 2020 / Revised: 15 July 2021 / Accepted: 16 September 2021

Abstract
In this paper we propose a novel methodology to construct Optimal Classification Trees that takes into account that noisy labels may occur in the training sample. The motivation of this new methodology is based on the superaditive effect of combining together margin based classifiers and outlier detection techniques. Our approach rests on two main elements: (1) the splitting rules for the classification trees are designed to maximize the separation margin between classes applying the paradigm of SVM; and (2) some of the labels of the training sample are allowed to be changed during the construction of the tree trying to detect the label noise. Both features are considered and integrated together to design the resulting *Optimal* Classification Tree. We present a Mixed Integer Non Linear Programming formulation for the problem, suitable to be solved using any of the available off-the-shelf solvers. The model is analyzed and tested on a battery of standard datasets taken from UCI Machine Learning repository, showing the effectiveness of our approach. Our computational results show that in most cases the new methodology outperforms both in accuracy and AUC the results of the benchmarks provided by OCT and OCT-H.

Keywords Multiclass classification · Optimal classification trees · Support vector machines · Mixed integer non linear programming · Classification · Hyperplanes

Mathematics Subject Classification 62H30 · 90C11 · 68T05 · 32S22

✉ Victor Blanco
vblanco@ugr.es

Alberto Japón
ajapon1@us.es

Justo Puerto
puerto@us.es

1 Institute of Mathematics (IMAG), Universidad de Granada, Granada, Spain

2 Institute of Mathematics (IMUS), Universidad de Sevilla, Sevilla, Spain

1 Introduction

Discrete Optimization has experienced a tremendous growth in the last decades, both in its theoretical and practical sides, partially provoked by the emergence of new computational resources as well as real-world applications that have boosted this growth. This impulse has also motivated the use of Discrete Optimization models to deal with problems involving a large number of variables and constraints, that years before would have not been possible to be dealt with. One of the fields in which Discrete Optimization has caused a larger impact is in Machine Learning. The incorporation of binary decisions to the classical approaches as Support Vector Machine Cortes and Vapnik (1995), Classification Trees Breiman et al. (1984), Linear Regression and Clustering, amongst other, has considerably enhanced their performance in terms of accuracy and interpretability (see e.g. Benati et al. (2017, 2021); Bertsimas and Dunn (2017); Blanco et al. (2020a, c, 2018); Blanquero et al. (2020a, b); Drucker et al. (1997); Gaudioso et al. (2017)). In particular, one of the most interesting applications of Machine Learning is that related with Supervised Classification.

Supervised classification aims at finding hidden patterns from a training sample of labeled data in order to predict the labels of out-of-sample data. Several methods have been proposed in order to construct highly predictive classification tools. Some of the most widely used methodologies are based on Deep Learning mechanisms Agarwal et al. (2018), *k*-Nearest Neighborhoods Cover and Hart (1967); Tang and Xu (2016), Naïve Bayes Lewis (1998), Classification Trees Breiman et al. (1984); Friedman et al. (2001) and Support Vector Machines Cortes and Vapnik (1995). The massive use of these tools has induced, in many situations, that malicious adversaries adaptively manipulate their data to mislead the outcome of an automatic analysis, and new classification rules must be designed to handle the possibility of this noise in the training labels. A natural example are the spam filters for emails, where malicious emails are becoming more difficult to automatically be detected since they have started to incorporate patterns that typically appear in legitimate emails (see e.g., Guzella and Caminhas (2009); Yu and Xu (2008)). As a consequence, the development of robust methods against these kind of problems has attracted the attention of researchers (see e.g., Bertsimas et al. (2019); Blanco et al. (2020b)).

In the context of binary classification problems, Support Vector Machines (SVM), introduced by Cortes and Vapnik Cortes and Vapnik (1995), builds the decision rule by means of a separating hyperplane with large margin between classes. This hyperplane can be obtained by solving a convex quadratic optimization problem, in which the goal is to separate data by their two differentiated classes, maximizing the margin between them and minimizing the misclassification errors. Duality properties of this optimization problem allow one to extend the methodology to find nonlinear separators by means of kernels. In Classification and Regression Trees (CART), firstly introduced by Breiman et. al Breiman et al. (1984), one constructs the decision rule based on a hierarchical relation among a set of nodes which is used to define paths that lead observations from the root node (highest node in the hierarchical relation), to some of the leaves in which a class is assigned to the data. These paths are obtained according to different optimization criteria over the predictor variables of the training sample. The decision rule comes up naturally, the classes predicted for new observations are the ones assigned to the terminal nodes in which observations fall in. Historically, CART is obtained heuristically through a greedy approach, in which each level of the tree is sequentially constructed: starting at the root node and using the whole training sample, the method minimizes an impurity measure function obtaining as a result a split that divides the sample into two disjoint sets which determine the two descendant nodes. This process is repeated until a given termination criterion is reached (minimum number of observations belonging to a leaf, maximum depth of the tree, or minimum percentage of observations of the same class on a leaf, among others). In this approach, the tree grows following a top-down greedy approach, an idea that is also shared in other popular decision tree methods like C4.5 Quinlan (1993) or ID3 Quinlan (1996). The advantage of these methods is that the decision rule can be obtained rather quickly even for large training samples, since the whole process relies on solving manageable problems at each node. Furthermore, these rules are interpretable since the splits only take into account information about lower or upper bounds on a single feature. Nevertheless, there are some remarkable disadvantages in these heuristic methodologies. The first one is that they may not obtain the *optimal* classification tree, since they look for the best split locally at each node, not taking into account the splits that will come afterwards. Thus, these local branches may not capture the proper structure of the data, leading to misclassification errors in out-of-sample observations. The second one is that, specially under some termination criteria, the solutions provided by these methods can result into very deep (complex) trees, resulting in overfitting and, at times, loosing interpretability of the classification rule. This difficulty is usually overcome by pruning the tree as it is being constructed by comparing the gain on the impurity measure reduction with respect to the complexity cost of the tree.

Mixing together the powerful features of standard classification methods and Discrete Optimization has motivated the study of supervised classification methods under a new paradigm (see Bertsimas and Dunn (2019)). In particular, recently, Bertsimas and Dunn Bertsimas and Dunn (2017) introduced the notion of *Optimal Classification Trees* (OCT) by approaching CART under optimization lens, providing a Mixed Integer Linear Programming formulation to optimally construct Classification Trees. In this formulation, binary variables are introduced to model the different decisions to be taken in the construction of the trees: deciding whether a split is applied and if an observation belongs to a terminal node. Moreover, the authors proved that this model can be solved for reasonable size datasets, and equally important, that for many different real datasets, significant improvements in accuracy with respect to CART can be obtained. In contrast to the standard CART approach, OCT builds the tree by solving a single optimization problem taking into account (in the objective function) the complexity of the tree, avoiding post pruning processes. Moreover, every split is directly applied in order to minimize the misclassification errors on the terminal nodes, and hence, OCTs are more likely to capture the essence of the data. Furthermore, OCTs can be easily adapted in the so-called OCT-H model to decide on splits based on hyperplanes (oblique) instead of on single variables. Another remarkable advantage of using optimization tools in supervised classification methods is that features such as sparsity or robustness, can be incorporated to the models by means of binary variables and constraints Günlük et al. (2018). The interested reader is refereed to the recent survey Carrizosa et al. (2020). We would like to finish this discussion pointing out one of the main differences between SVM and Classification Trees: SVM accounts for misclassification errors based on distances (to the separating hyperplane), i.e., the closer to the correct side of the separating hyperplane, the better, whereas in Classification Trees all misclassified observations are equally penalized.

Recently, Blanco et. al Blanco et al. (2020b) proposed different SVM-based methods that provide robust classifiers under the hypothesis of label noises. The main idea supporting those methods is that labels are not reliable, and in the process of building classification rules it may be beneficial to *flip* some of the labels of the training sample to obtain more accurate classifiers. With this paradigm, one of the proposed methods, RE-SVM, is based on constructing a SVM separating hyperplane, but simultaneously allowing observations to be relabeled during the training process. The results obtained by this method, in datasets in which noise was added to the training labels, showed that this strategy outperforms, in terms of accuracy, classical SVM and other SVM-based robust methodologies. See Bertsimas et al. (2019) for alternative robust classifiers under label noise.

In this paper we propose a novel binary supervised classification method, called Optimal Classification Tree with Support Vector Machines (OCTSVM), that profits both from the ideas of SVM and OCT to build classification rules. Specifically, our method uses the hierarchical structure of OCTs, which leads to easily interpretable rules, but splits are based on SVM hyperplanes, maximizing the margin between the two classes at each node of the tree. The fact that the combination of SVM and classification tree tools provides enhanced classifiers is not new. A similar approach can be found in Bennett and Blue (1998). Nevertheless, in that paper the authors analyze the greedy CART strategy by incorporating, sequentially the maximization of the margin, over known assignments of observations to the leaves of the tree. Opposite to that, OCTSVM does not assume those assumptions and it performs an exact optimization approach. Moreover, this new method also incorporates decisions on relabeling observations in the training dataset, making it specially suitable for datasets where adversary attacks are suspected. The results of our experiments show that OCTSVM outperforms other existing methods under similar testing conditions. In contrast to the robust classifiers under label noise provided in Bertsimas et al. (2019), our method is not based on the worst-case paradigm commonly used in the field of robust optimization, but in the convenience of finding good classifiers under the presence of unknown noisy labels.

The rest of the paper is organized as follows. In Sect. 2 we recall the main ingredients of our approach, in particular, SVM, RE-SVM and OCTs, as well as the notation used through the paper. Section 3 is devoted to introduce our methodology, and presents a valid Mixed Integer Non Linear Programming (MINLP) formulation. In Sect. 4 we report the results obtained in our computational experiments, in particular, the comparison of our method with OCT, OCT-H and the greedy CART. Finally, some conclusions and further research on the topic are drawn in Sect. 5.

2 Preliminaries

In this section we recall the main ingredients in the approach that will be presented in Sect. 3 which allows us to construct robust classifiers under label noises, namely, Support Vector Machines with Relabeling (RE-SVM) and Optimal Classification Trees with oblique splits (OCT-H).

All through this paper, we consider that we are given a training sample $\mathcal{X} = \{(x_1, y_1), \dots (x_n, y_n), \} \subseteq \mathbb{R}^p \times \{-1, +1\}$, in which p features have been measured for a set of n individuals $(x_1, \dots, x_n)$ as well as a ± 1 label is also known for each of them $(y_1, \dots, y_n)$. The goal of supervised classification is, to derive, from $\mathcal{X}$, a decision rule $D_{\mathcal{X}} : \mathbb{R}^p \rightarrow \{-1, 1\}$ capable to accurately predict the right label of out-sample observations given only the values of the features. We assume, without loss of generality that the features are normalized, i.e., $x_1, \dots, x_n \in [0, 1]^p$.

2.1 Support vector machines

One of the most used optimization-based method to construct classifiers for binary labels is SVM Cortes and Vapnik (1995). This classifier is constructed by means of a separating hyperplane in the feature space, $\mathcal{H} = \left\{z \in \mathbb{R}^p : \omega' z + \omega_0 = 0\right\}$, such that the decision rule becomes:

$$D_{\mathcal{X}}^{SVM}(z) = \begin{cases} -1 & \text{if } \omega' z + \omega_0 < 0, \\ 1 & \text{if } \omega' z + \omega_0 \geq 0. \end{cases}$$

To construct such a hyperplane, SVM chooses the one that simultaneously maximizes the separation between classes and minimizes the errors of misclassified observations. These errors are measured proportional to the distances (in the feature space) from the observations to their label half-space. SVM can be formulated as a convex non linear programming (NLP) problem. This approach allows one for the use of a kernel function as a way of embedding the original data in a higher dimension space where the separation may be easier without increasing the difficulty of solving the problem (the so-called kernel trick).

2.2 Relabeling

On the other hand, SVM has also been studied in the context of robust classification. In Blanco et al. (2020b) three new models derived from SVM are developed to be applicable to datasets in which the observations may have wrong labels in the training sample. This characteristic is incorporated into the models by allowing some of the labels to be swapped (relabelled) at the same time that the separating hyperplane is built. Two of these models combine SVM with cluster techniques in a single optimization problem while the third method, the so-called RE-SVM, relabels observations based on misclassification errors without using clustering techniques, what makes it easier to train. The RE-SVM problem can be formulated as:

$$\min \frac{1}{2}\|\omega\|_2^2 + c_1 \sum_{i=1}^{n} e_i + c_2 \sum_{i=1}^{n} \xi_i \tag{RE-SVM}$$

$$\begin{aligned} \text{s.t. } & (1-2\xi_i)y_i(\omega' x_i + \omega_0) \geq 1 - e_i, && \forall i = 1, \ldots, n, && (1)\\ & \omega \in \mathbb{R}^p,\ \omega_0 \in \mathbb{R}, && \\ & e_i \in \mathbb{R}^+,\ \xi_i \in \{0, 1\}, && \forall i = 1, \ldots, n, \end{aligned}$$

where ξ_i takes value 1 if the ith observation of the training sample is relabelled, and 0 otherwise and e_i is the misclassifying error defined as the hinge loss:

$$e_i = \begin{cases} \max\{0, 1 - y_i(\omega' x_i + \omega_0)\} \text{ if observation } i \text{is not relabelled} \\ \max\{0, 1 + y_i(\omega' x_i + \omega_0)\} \text{ if observation} i \text{ is relabelled} \end{cases},$$

for $i = 1, \ldots, n$. The costs parameters c_1 and c_2 (unit cost per misclassifying error and per relabelled observation) allow one to find a trade-off between large separation between classes: c_1 and c_2 are parameters modelling the unit cost of misclassified errors and relabelling, respectively. ($\|\cdot\|_2$ stands for the Euclidean norm in $\mathbb{R}^p$.) Constraints (1) assures the correct definition of the hinge loss and relabelling variables.

The problem above can be reformulated as a Mixed Integer Second Order Cone Optimization (MISOCO) problem, for which off-the-shelf optimization solvers such as CPLEX or Gurobi are able to solve medium size instances in reasonable CPU time.

2.3 Optimal classification trees

Classification Trees (CT) are a family of classification methods based on a hierarchical relation among a set of nodes. The decision rule for CT methods is built by recursively partitioning the feature space by means of hyperplanes. At the first stage, a root node for the tree is considered where all the observations belongs to. Branches are sequentially created by splits on the feature space, creating intermediate nodes until a leaf node is reached. Then, the predicted label for an observation is given by the majority class of the leaf node where it belongs to.

Specifically, at each node, t, of the tree a hyperplane $\mathcal{H}_t = \{z \in \mathbb{R}^p : \omega_t' z + \omega_{t0} = 0\}$ is constructed and the splits are defined as $\omega_t' z + \omega_{t0} < 0$ (left branch) and $\omega_t' z + \omega_{t0} \geq 0$ (right branch). In Fig. 1 we show a simple classification tree with depth two, for a small dataset with 6 observations, that are correctly classified on the leaves.

The most popular method to construct Classification Trees from a training dataset is CART, introduced by Brieman et. al Breiman et al. (1984). CART is a greedy heuristic approach, which myopically constructs the tree without further foreseen to deeper nodes. Starting at the root node, it decides the splits by means of hyperplanes minimizing an impurity function in each node. Each split results in two new nodes, and this procedure is repeated until a stopping criterion is reached (maximal depth,

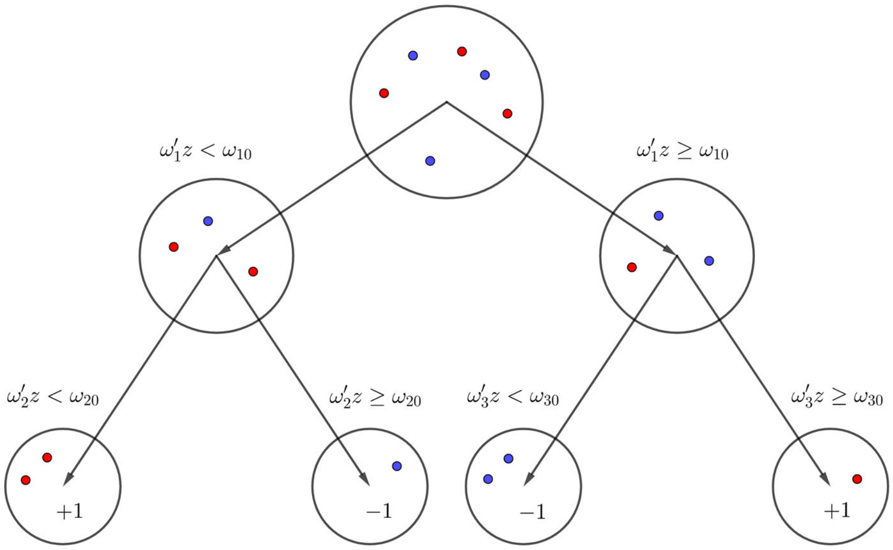
$\omega_1'z < \omega_{10}$
$\omega_1'z \geq \omega_{10}$
$\omega_2'z < \omega_{20}$
$\omega_2'z \geq \omega_{20}$
$\omega_3'z < \omega_{30}$
$\omega_3'z \geq \omega_{30}$
$+1$
-1
-1
$+1$

Fig. 1 Decision tree of depth two

minimum number of observations in the same node, etc). Deep CART trees may lead to overfitting in out-of-sample observations, and therefore trees are normally subject to a prune process based on the trade-off between the impurity function reduction and a cost-complexity parameter. The main advantage of CART is that it is easy to implement and fast to train.

On the other hand, Bertsimas and Dunn Bertsimas and Dunn (2017) have recently proposed an optimal approach to build CTs by solving a mathematical programming problem which builds the decision tree in a compact model considering its whole structure and at the same time making decisions on pruning or not pruning the branches.

Given a maximum depth, D, for the Classification Tree it can have at most $T = 2^{D+1} - 1$ nodes. These nodes are differentiated in two types:

- Branch nodes: $\tau_B = \{1, \ldots, \lfloor T/2 \rfloor\}$ are the nodes where the splits are applied.
- Leaf nodes: $\tau_L = \{\lceil T/2 \rceil, \ldots, T\}$ are the nodes where predictions for observations are performed.

We use the following notation concerning the hierarchical structure of a tree:

- $p(t)$: parent of node t, for $t = 1, \ldots, T$.
- τ_{bl}: set of nodes that follow the left branch on the path from their parent nodes. Analogously, we define τ_{br} as the set of nodes whose right branch has been followed on the path from their parent nodes.
- u: set of nodes that have the same depth inside the tree. We represent by U the whole set of levels. The root node is the zero-level, u_0, hence, for a given depth D we have $D + 1$ levels, being u_D the set of leaf nodes.

OCTs are constructed by minimizing the following objective function:

$$\sum_{t \in \tau_L} L_t + \alpha \sum_{t \in \tau_B} d_t,$$

where L_t stands for the misclassification errors at the leaf t (measured as the number of wrongly classified observations in the leaf), and d_t is a binary variable that indicates if a split is produced at t. Therefore, the constant α is used to regulate the trade-off between the complexity (depth) and the accuracy (misclassifying errors of the training sample) of the tree. In its simplest version, motivated by what it is done in CART, the splits are defined by means of a single variable, i.e., in the form $x_j \leq \omega_{j0}$. Nevertheless, OCT can be extended to a more complex version where the splits are hyperplanes defined by their normal vector, $a \in \mathbb{R}^p$ which is known as OCT-H. Moreover, a robust version of OCT has also been studied under the noise label scenario Bertsimas et al. (2019).

3 Optimal classification trees with SVM splits and relabeling (OCTSVM)

This section is devoted to introduce our new classification methodology, namely OCTSVM. The rationale of this approach is to combine the advantage of hierarchical classification methods such as Classification Trees, with the benefits from using distance-based classification errors, by means of hyperplanes maximizing the margin between them (SVM paradigm). Therefore, this new model rests on the idea of constructing an optimal classification tree in which the splits of the nodes are performed by following the underlying ideas of model (RE – SVM): (1) the splits are induced by hyperplanes in which the positive and the negative classes are separated maximizing the margin between classes, (2) minimizing the classification errors, and (3) allowing observations to be relabeled along the training process. In contrast to what it is done in other Classification Tree methods, OCTSVM does not make a distinction (beyond the hierarchical one) between branch and leaf nodes, in the sense that RE-SVM based splits are sequentially applied in each node, and the final classification for any observation comes from the hyperplanes resulting at the last level of the tree, in case there are no pruned branches, or at the last node where a split was made in case of a pruned branch.

As it has been pointed out before, OCT-H is a classification tree that allows the use of general (oblique) hyperplane splits, which is built by solving a single optimization problem that takes into account the whole structure of the tree. Nevertheless, despite the good results this method has proven to obtain in real classification problems, a further improvement is worth to be considered. In Fig. 2 we see a set of points in the plane differentiated by geometrical elements (triangles and circles) in two classes. Looking at the left picture, one can see one of the optimal solutions of OCT-H for depth equal to two, where the red hyperplane is the split applied at the root node and the black ones are applied on the left and right descendants, which define the four leaves. This solution is optimal, for a certain value of the cost-complexity parameters, since it does not make any mistakes on the final classification. Nevertheless, since

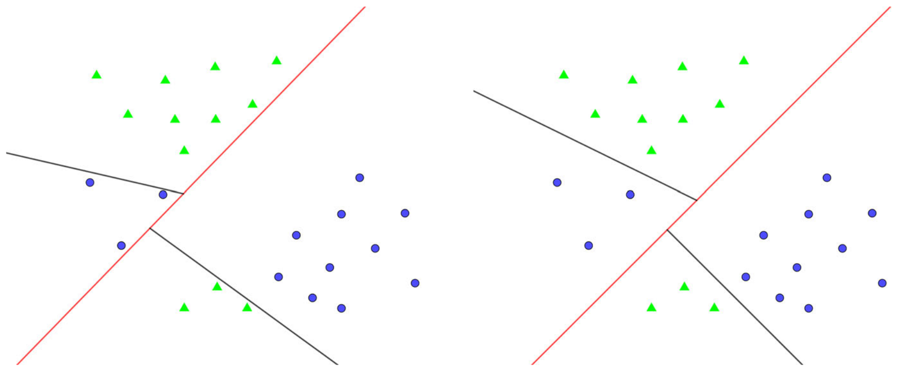

Fig. 2 Optimal solutions for OCT-H with $D = 2$ (left) and OCTSVM with $D = 1$ (right)

this method does not have any kind of control on the distances from points to the hyperplanes, one can observe that the blue class has very tiny margins at the leaves, and hence, for this class, misclassification errors are more likely to occur in out-of-sample observations. On the other hand, on the right side of Fig. 2 one sees another possible optimal solution for the OCTSVM model with depth equal to one (note that unlike OCT-H, OCTSVM constructs a final SVM-based classifier at each of the leaf nodes, which may be identified with an extra depth). Again, the red hyperplane is the split applied at the root node and the black ones are the classification splits applied at the two leaves. Despite these two methods are obtaining a perfect classification on the training sample, Fig. 2 shows that OCTSVM provides a more balanced solution than OCT-H since it has wider margins between both classes, what could be translated into a higher accuracy for out-of-sample observations.

In order to formulate the OCTSVM as a MINLP, we will start describing the rationale of its objective function that must account for the margins induced by the splits at all nodes, the classification errors, the penalty paid for relabelling observations and the cost-complexity of the final classification tree. To formulate the above concepts we need different sets of variables. First of all, we consider continuous variables: $\omega_t \in \mathbb{R}^p,\ \omega_{t_0} \in \mathbb{R},\ t = 1, \dots, T$, which represent the coefficients and the intercept of the hyperplane performing the split at node t. Taking into account that the margin of the hyperplane $\mathcal{H}_t = \{z : \omega_t z + \omega_{t0} = 0\}$ is given by $\frac{2}{||\omega_t||}$, maximizing the minimum margin between classes induced by the splits can be done introducing an auxiliary variable $\delta \in \mathbb{R}$ (that will be minimized in the objective function) which is enforced by the following constraints:

$$\frac{1}{2}||\omega_t||_2 \leq \delta \qquad \forall t = 1, \dots, T. \tag{2}$$

Once the maximization of the margin is set, we have to model the minimization of the errors at the nodes, whereas at the same time we minimize the number of relabelled observations. These two tasks are accomplished by the variables $e_{it} \in \mathbb{R},\ i = 1, \dots, n,\ t = 1, \dots, T$, that account for the misclassification error of observation i at node t, and $\xi_{it} \in \{0, 1\}$ binary variables modelling whether observation i is relabeled or not at node t. If c_1 and c_2 are the unit costs of misclassification and relabelling, respectively, our goal is achieved adding to the objective function the following two terms:

$$c_1 \sum_{i=1}^{n} \sum_{t=1}^{T} e_{it} + c_2 \sum_{i=1}^{n} \sum_{t=1}^{T} \xi_{it}.$$

The correct meaning of these two sets of variables must be enforced by some families of constraints that we describe next. Nevertheless, for the sake of readability before describing those constraints modeling these e_{it} and ξ_{it}, we must introduce another family of variables the $\beta_{it} \in \mathbb{R}^p$, $\beta_{i0} \in \mathbb{R}$, $i = 1, \ldots, n$, $t = 1, \ldots, T$, which are continuous variables equal to the coefficients of the separating hyperplane at node t when observation i is relabelled, and equal to zero otherwise. In addition, we consider binary variables $z_{it} \in \{0, 1\}$ needed to control whether observation i belongs to node t of the tree. Now, putting all these elements together, as it is done in RE-SVM, we can properly define the splits and their errors at each node of the tree using the following constraints:

$$y_i(\omega_t' x_i + \omega_{t0}) - 2y_i(\beta_t' x_i + \beta_{it0}) \geq 1 - e_{it} - M(1 - z_{it}), \quad \begin{cases} \forall i = 1, \ldots, n, \\ t = 1, \ldots, T, \end{cases} \tag{3}$$

$$\beta_{itj} = \xi_{it}\omega_{tj}, \quad \forall i = 1, \ldots, n, t = 0, \ldots, T, \; j = 0, \ldots, p. \tag{4}$$

Constraints (4), which can be easily linearized, are used to define the β_{it} variables: they equal the ω_t variables when the observation i is relabelled ($\xi_i = 1$), and are equal to zero otherwise ($\xi_i = 0$). On the other hand, constraints (3) control the relabelling at each node of the tree. If an observation i is in node t ($z_{it} = 1$), and $\xi_i = 0$, we obtain the standard SVM constraints for the separating hyperplane. Nevertheless, if $\xi_i = 1$, then the separating constraints are applied for the observation i as if its class were the opposite to its actual one, i.e., as if observation i is relabeled. Moreover, since M is a big enough constant, these constraints do not have any kind of impact in the error variables of node t if observation i does not belong to this node ($z_{it} = 0$).

On the other hand, there are still some details left that must be imposed to make the model to work as required. In the decision tree, observations start at the root node and they advance descending through the levels of the tree until they reach a leaf or a pruned node. Hence, we have to guarantee that observations must belong to one, and only one, node per level. By means of the z_{it} variables, this can be easily done by the usual assignment constraints applied in each level, $u \in U$, of the tree:

$$\sum_{t \in u} z_{it} = 1, \qquad \forall i = 1, \ldots, n, \; u \in U. \tag{5}$$

Moreover, for consistency in the relation between a node and its ancestor, it is clear that if observation i is in node t ($z_{it} = 1$), then, observation i must be also in the parent of node t ($z_{ip(t)} = 1$), with the only exception of the root node. Besides, if observation i is not in node t ($z_{it} = 0$), then i can not be in its successors, and this is modeled by adding the following constraints to the problem:

$$z_{it} \leq z_{ip(t)}, \qquad \forall i = 1, \ldots, n, t = 2, \ldots, T. \tag{6}$$

So far, the OCTSVM model has everything it needs to properly perform the splits by following the RE-SVM rationale described in (RE – SVM), taking into consideration the tree complexity, and maintaining the hierarchical relationship among nodes. The last element that we need to take care of, to assure the correct performance of the whole model, is to define how observations follow their paths inside the tree. We get from constraints (6) that observations move from parent to children (nodes), but every non terminal node has a left and a right child node, and we need to establish how observations take the left or the right branch. Since the splits are made by the separating hyperplane, we force observations that lie on the positive half space of a hyperplane to follow the right branch of the parent node, and observations that lie on the negative one to take the left branch. This behavior is modeled with the binary variables $\theta_{it} \in \{0, 1\}$, that are used to identify whether observation i lies in the positive half space of the separating hyperplane at node t, $\theta_{it} = 1$, or if observation i lies on the negative half space, $\theta_{it} = 0$. By considering M a big enough constant, the correct behavior of the path followed by the observations is enforced by the following constraints:

$$\omega_t' x_i + \omega_{t0} \geq -M(1 - \theta_{it}), \qquad \forall i = 1, \ldots, n, t = 1, \ldots, T, \tag{7}$$

$$\omega_t' x_i + \omega_{t0} \leq M\theta_{it}, \qquad \forall i = 1, \ldots, n, t = 1, \ldots, T. \tag{8}$$

Hence, by making use of these θ_{it} variables, and distinguishing between nodes that come from left splits, τ_{bl} (nodes indexed by even numbers), and right splits, τ_{br} (nodes indexed by odd numbers), we control that the observations follow the paths through the branches in the way we described above throughout the following constraints:

$$z_{ip(t)} - z_{it} \leq \theta_{ip(t)}, \qquad \forall i = 1, \ldots, n, t \in \tau_{bl} \tag{9}$$

$$z_{ip(t)} - z_{it} \leq 1 - \theta_{ip(t)}, \qquad \forall i = 1, \ldots, n, t \in \tau_{br} \tag{10}$$

According to constraints (9), if an observation i is on the parent node of an even node t ($z_{ip(t)} = 1$), and i lies on the negative half space of the hyperplane defining the split on $p(t)$ ($\theta_{ip(t)} = 0$), then z_{it} is forced to be 1. Hence, $\theta_{ip(t)} = 0$ implies that observation i takes the left branch to the child node $t \in \tau_{bl}$. Moreover, we can see that this constraint is consistent since if $z_{ip(t)} = 1$, but observation i is not in the left child node, $z_{it} = 0$, $t \in \tau_{bl}$, then $\theta_{ip(t)}$ equals 1, what means that observation i lies on the positive half space of the hyperplane of $p(t)$. On the other hand, constraints (10) are similar but for the right child nodes, τ_{br}. If an observation i is in the parent node of an odd node $t \in \tau_{br}$, and i lies on the positive half space of the hyperplane of $p(t)$ ($\theta_{ip(t)} = 1$), then, $z_{it} = 1$ what means that observation i has to be on node t.

The final term to be included in the objective function of the problem is the complexity of the resulting tree. Following the approach in OCT and OCT-H, we consider binary variables $d_t \in \{0, 1\}$, $t = 1, \ldots, T$, that control whether a separating split is applied at node t. Thus, to control the tree complexity resulting of the process, we minimize the sum of these variables multiplied by a cost penalty c_3. Gathering all the components together, the objective function to be minimized in our problem results in

$$\delta + c_1 \sum_{i=1}^{n} \sum_{t=1}^{T} e_{it} + c_2 \sum_{i=1}^{n} \sum_{t=1}^{T} \xi_{it} + c_3 \sum_{t=1}^{T} d_t.$$

According to this, it is important to make a distinction between the methods that use, or the ones that do not use, SVM based splits. When a model does not use SVM based splits, taking into account that the binary variable $d_t = 1$ implies that a hyperplane is being used to split the points in node t, complexity can be easily regulated by just imposing $\|\omega_t\|_2 \leq M d_t$ in all the branch nodes. Nevertheless, when using a SVM based method such as the one we are presenting here, the previous constraint would be in conflict with constraints (3). This is due to the fact that $d_t = 0$ would imply $\|\omega_t\|_2 = 0$, and under this scenario, observations in this node would have to pay for the margin violation error $e_{it} = 1$, even though these errors are not justified since points are not being separated at this node. To overcome this issue, we need to add some other constraints to the model, but before getting into the mathematical formulation, we would like to emphasize the different aspects that explain the difficulty of the problem. The objective function accounts for the complexity cost of node t ($d_t = 1$) when a hyperplane is actually built so as to split the points in node t. In case we do not pay the complexity cost in node t ($d_t = 0$), it does not necessarily mean that a hyperplane is not built at this node, it could simply turn out to be a hyperplane that leaves all the observations at one of the half spaces it creates, i.e., a hyperplane that is not splitting the points. These non splitting hyperplanes do not affect the training set at all, and more importantly, they would not affect out of sample observations since predictions will occur in leaf nodes or in the first branch node in which $d_t = 0$. Going back to the formulation, the first step now is to introduce some binary variables, $h_{it} \in \{0, 1\}$, $i \in N$, $t \in \tau_b$, that will be relaxed afterwards, defined by $h_{it} = z_{it}\theta_{it}$. These variables tell us which observations belong to node t and at the same time lie on the positive half space of the split created in t. It is important to define these variables because they are used to measure whether the hyperplane is splitting the points in node t or not. This is going to be done by means of some new binary variables $v_t \in \{0, 1\}$, $t \in \tau_b$, that will be equal to zero in case all the points in node t belong to the positive half space of the hyperplane built in this node, $\mathcal{H}_t$, and one otherwise. For the definition of the d_t variables to make sense, the following constraints must simultaneously be considered:

$$h_{it} = z_{it}\theta_{it} \qquad \forall i = 1, \ldots, n, t = 1, \ldots, T, \qquad (11)$$

$$\sum_{i=1}^{n} (1 - h_{it}) \leq M v_t \qquad \forall t = 1, \ldots, T, \qquad (12)$$

$$\sum_{i=1}^{n} h_{it} - M(1 - v_t) \leq M d_t \qquad \forall t = 1, \ldots, T, \tag{13}$$

where M is a big enough constant. The reader should observe that (11) is the definition of the h_{it} variables as the product of z_{it} by θ_{it}. These constraints can be easily linearized by the usual tricks. On the other hand, taking into account that v_t variables would tend to be zero (by the effects on constraints (13)), constraints (12) stand for the definition of the v_t variables, since these variables could be equal to zero just in case all the h_{it} are equal to one, what means that all the observations belong to the positive half space of $\mathcal{H}_t$. Finally, we obtain through constraints (13) the definition of the d_t variables. Since these variables are being minimized in the objective function, they would try to be equal to zero. Nevertheless, this can just happen if $v_t = 0$, what means that all the observations belong to the positive half space of $\mathcal{H}_t$, or if $\sum\limits_{i \in N} h_{it} = 0$ in case $v_t = 1$, what means that all the observations belong to the negative half space of $\mathcal{H}_t$. Hence, d_t is equal to one if and only if the points in node t are being actually separated by $\mathcal{H}_t$.

Finally, another point that it is important to remark about the d_t variables is that once a non-leaf node does not split (that is, the corresponding branch is pruned at this node), the successors of node t can not make splits either to maintain the correct hierarchical structure of the tree. Recalling that $p(t)$ is the parent node of node t, we can guarantee this relationship throughout the following constraints

$$d_t \leq d_{p(t)}, \qquad \forall t = 2, \ldots, T. \tag{14}$$

Some of the constraints discussed above require big-M constants. The value of these constants can be adjusted once we know the dimension (n, p) of the dataset involved in the optimization problem. Recall that $x_i \in [0, 1]^p$, therefore it is well known that the maximum distance between two points in such a domain is $\sqrt{p}$. Hence, the maximum distance from a point to a hyperplane is also $\sqrt{p}$. According to this, the big-M constant in constraints (3), (6) and (7) is actually $\sqrt{p}$.

On the other hand, the h_{it} variables are upper bounded by one, and therefore $\sum\limits_{i=1}^{n} h_{it}$ is upper bounded by n. As a result, the big-M constant involved in constraints (11) and (12) is n as well.

Gathering all the constraints together, and substituting the generic big-M constants by those estimated in our discussion above, the OCTSVM is obtained by solving the following MINLP:

$$\min \delta + c_1 \sum_{i=1}^{n} \sum_{t=1}^{T} e_{it} + c_2 \sum_{i=1}^{n} \sum_{t=1}^{T} \xi_{it} + c_3 \sum_{t=1}^{T} d_t \tag{OCTSVM}$$

$$
\begin{aligned}
\text{s.t. } & \frac{1}{2}||\omega_t||_2 \leq \delta, \quad \forall t = 1, \ldots, T, \\
& y_i(\omega_t' x_i + \omega_{t0}) - 2y_i(\beta_t' x_i + \beta_{it0}) \geq 1 - e_{it} - \sqrt{p}(1 - z_{it}), \quad \forall i = 1, \ldots, n, t = 1, \ldots, T, \\
& \beta_{itj} = \xi_{it}\omega_{tj}, \quad \forall i = 1, \ldots, n, t = 0, \ldots, T, \; j = 0, \ldots, p, & (15) \\
& \sum_{t \in u} z_{it} = 1, \quad \forall i = 1, \ldots, n, \; u \in U, \\
& z_{it} \leq z_{ip(t)}, \quad \forall i = 1, \ldots, n, t = 2, \ldots, T, \\
& \omega_t' x_i + \omega_{t0} \geq -\sqrt{p}(1 - \theta_{it}), \quad \forall i = 1, \ldots, n, t = 1, \ldots, T, \\
& \omega_t' x_i + \omega_{t0} \leq \sqrt{p}\theta_{it}, \quad \forall i = 1, \ldots, n, t = 1, \ldots, T, \\
& z_{ip(t)} - z_{it} \leq \theta_{ip(t)}, \quad \forall i = 1, \ldots, n, t \in \tau_{bl}, \\
& z_{ip(t)} - z_{it} \leq 1 - \theta_{ip(t)}, \quad \forall i = 1, \ldots, n, t \in \tau_{br}, \\
& h_{it} = z_{it}\theta_{it} \quad \forall i = 1, \ldots, n, t = 1, \ldots, T, & (16) \\
& \sum_{i=1}^{n}(1 - h_{it}) \leq n v_t \quad \forall t = 1, \ldots, T, \\
& \sum_{i=1}^{n} h_{it} - n(1 - v_t) \leq n d_t \quad \forall t = 1, \ldots, T, \\
& d_t \leq d_{p(t)} \quad \forall t = 2, \ldots, T, \\
& e_{it} \in \mathbb{R}^+, \beta_{it} \in \mathbb{R}^p, \beta_{it0} \in \mathbb{R}, \xi_{it}, z_{it}, \theta_{it}, h_{it} \in \{0, 1\}, \forall i = 1, \ldots, n, t = 1, \ldots, T, \\
& \omega_t \in \mathbb{R}^p, \omega_{t0} \in \mathbb{R}, d_t, v_t \in \{0, 1\}, \forall t = 1, \ldots, T.
\end{aligned}
$$

The reader may note that the above formulation has two families of bilinear constraints, namely (15) and (16), that can be easily linearized giving rise to a mixed integer second order cone formulation.

In Table 1 we summarize the variables, index sets and parameters used in our model.

4 Experiments

In this section we present the results of our computational experiments. Five different classification tree-based methods are compared, CART, OCT, OCT-H, OCT+SVM and OCTSVM, on nine popular real-life datasets from UCI Machine Learning Repository Dua and Graff (2017). Notice that OCT+SVM is a modification of our OCTSVM in which the ξ_{it} variables are fixed to zero, i.e., no relabeling is allowed in the model. This method is included so as to assess the isolated effect of using margin-based splits, without relabeling, within the classification trees. The considered datasets together with their names and dimensions (n: number of observations, p: number of features) are reported in the three tables of this section.

Our computational experiments focus on the analysis of the accuracy and AUC (area under the ROC curve) of the different classification tree-based methods. This analysis is based in four different experiments for each one of the nine considered dataset. Our goal is to analyze the usefulness of the different methods for classifying data affected by label noise. Therefore, in our experiments we use, apart from the original datasets, three different modifications where in each one of them a percentage

Table 1 Summary of the notation, variables and parameters used in our model

Variables	
$\omega_t \in \mathbb{R}^p$	Coefficients of the separating hyperplane of node t.
$\omega_{t_0} \in \mathbb{R}$	Intercepts of the separating hyperplane of node t.
$\delta = \max\limits_{t=1,\ldots,T} \left\{\frac{1}{2}\|\|\omega_t\|\|_2\right\}$	Inverse of the minimum margin between splitting hyperplanes.
$e_{it} \in \mathbb{R}_+$	misclassification error of observation i at node t.
$\xi_{it} \in \{0, 1\}$	If observation i is relabeled or not at node t.
$\beta_{it} \in \mathbb{R}^p$	Coefficients of the separating hyperplane at node t when observation i is relabelled, and equal to zero otherwise.
$\beta_{i0} \in \mathbb{R}$	Intercepts of the separating hyperplane at node t when observation i is relabelled, and equal to zero otherwise.
$z_{it} \in \{0, 1\}$	If observation i belongs or not to node t.
$d_t \in \{0, 1\}$	If a split is applied or not at node t.
$\theta_{it} \in \{0, 1\}$	If observation i lies on the positive half space.
$h_{it} \in \{0, 1\}$	If observation i is in node t and lies on the positive half space.
$v_t \in \{0, 1\}$	If not all observations in node t lie on the positive half space of the hyperplane in node t.
Sets and indices	
$i = 1, \ldots, n$	training observations.
$t = 1, \ldots, T$	nodes of the tree.
$p(t)$	parent of node t.
$\tau_B = \{1, \ldots, \lfloor T/2 \rfloor\}$	Branch nodes.
$\tau_L = \{\lfloor T/2 \rfloor, \ldots, T\}$	Leaf nodes.
τ_{bl}	nodes that follow the left branch on the path from their parent nodes.
τ_{br}	nodes whose right branch has been followed on the path from their parent nodes.
u	set of nodes that have the same depth inside the tree.
Parameters	
c_1	unit misclassification cost.
c_2	unit relabelling cost.
c_3	unit splitting cost.

of the labels in the sample are randomly flipped. The percentages of flipped labels range in {0%, 20%, 30%, 40%}, where 0% indicates that the original training data set is used to construct the tree.

We perform a 4-fold cross-validation scheme, i.e., datasets are split into four random train-test partitions. One of the folds is used for training the model while the rest are used for testing. When testing the classification rules, we compute the accuracy, in percentage, on out of sample data:

$$ACC = \frac{\#\textit{Well Classified Test Observations}}{\#\textit{Test Observations}} \cdot 100.$$

as well as the AUC, which provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example.

The CART method was coded in `R` using the `rpart` library. OCT, OCT-H, OCT+SVM and OCTSVM mathematical programming models were coded in `Python` and solved using `Gurobi` 8.1.1 on a PC Intel Core i7-7700 processor at 2.81 GHz and 16GB of RAM. A time limit of 30 seconds was set for training. Not all the problems were solved to optimality, nevertheless, this time limit was enough in order to obtain good classifiers. The average gap of the experiments is reported in table 4.

The calibration of the parameters of the different optimization-based models compared in these experiments was set as follows:

- For OCTSVM we used a grid on $\left\{10^i : i = -5, \dots, 5\right\}$ for the constants c_1 and c_2, and a smaller one $\left\{10^i : i = -2, \dots, 2\right\}$ for c_3. For OCT+SVM c_2 is not used since all the ξ_{it} variables are set to zero.
- For OCT, in order to check every possible optimal subtree of a maximal tree for a given depth D, we did the predictions in two steps. We first set up a grid on the parameter $c_1 = \left\{1, \dots, 2^D - 1\right\}$ and solved a slightly different OCT model in which the objective function did not take into account the complexity term, since complexity was already considered in the model by adding the following constraint $\sum\limits_{t \in \tau_B} d(t) \leq c_1$. The resulting solutions that were optimal for a certain c_1 of the original OCT problem were afterwards computed. This methodology was not used in OCT-H since the grid of the modified problem should have been extended to $c_1 = \left\{1, \dots, p(2^D - 1)\right\}$, therefore we used the same grid that we used for c_1 in OCTSVM directly into OCT-H formulation. On the other hand, the minimum number of observations per leaf was set to a 5% of the training sample size.
- CART trees were post-processed to satisfy any depth constraint as it was done with OCT. The minimum number of observations per leaf was the same 5% of the training sample size that we used for OCT and OCT-H.

Last to mention, the depth, D, considered in these experiments was set equal to three for CART, OCT and OCT-H, whereas for OCTSVM and OCT+SVM we fixed depth equal to two, creating consequently trees with 3 levels, to set a fair comparison among the different methods.

For each dataset, we replicate each experiment four times. In Table 2 we show the average (± standard deviation) accuracies for each one of the methods and datasets. In addition, Table 3 shows the average (± standard deviation) AUC results. The best results are highlighted (boldfaced). The first column stands for the percentage of flipped labels (FL) of the training sets. On the other hand, the last column shows the mean difference (Diff) between OCTSVM and the best result among OCT and OCT-H. Finally, in Table 4 we report the average MINLP gaps obtained by the models at the end of the training time limit.

Observe that when 0% of the training labels are flipped (i.e., the original dataset), one realizes of a general trend in accuracy and AUC of the different models: **CART** < **OCT** < **OCT-H** < **OCTSVM**, with the only exceptions of `Wholesale` and `Banknote` in which OCT-H obtains a non-meaningful slightly larger average accuracy. We would like to emphasize that these results are not surprising. Indeed, on the one hand, OCT is an optimal classification version of the CART algorithm, and hence better results should be expected from this model, as already shown in the computational experience in Bertsimas and Dunn (2017), and also evidenced in our experiments. Moreover, OCT is just a restricted version of OCT-H, in that in the latter, the set of admissible hyperplanes is larger than in the former. Also, as already pointed out in Fig. 2, OCTSVM goes one step further and, apart from allowing oblique hyperplanes based trees, has a more solid structure due to the margin maximization, the distance based errors and the possibility of relabeling points. All together results in higher accuracy and AUC results as shown in our experiments. In some datasets the comparison of the different methods gives rather similar results, however in some others OCTSVM is above the other methods more than 5% percent both in accuracy and AUC. The simplified methodology based on OCTSVM in which no relabeling is allowed, named OCT+SVM, shows a similar performance in both accuracy and AUC than OCT-H.

Turning to the results on datasets with flipped labels on the training dataset, OCTSVM clearly outperforms the rest of methods and consistently keeps its advantage in terms of accuracy and AUC with respect to the other methods. OCT, OCT-H and OCT+SVM, which are both above CART, alternate higher-lower results among the different datasets. Our method, namely OCTSVM, clearly is capable to capture the wrongly labeled observations and constructs classification rules able to reach higher accuracies and AUC, even when 40% of the labels were flipped, while other methods, give rise to classifiers that significantly worsen their performance in terms of accuracy and AUC in the test sample.

Concerning the MINLP Gaps obtained with the optimization-based methods (Table 4) one can observe that within the 30 seconds of time limit, most of the instances, for all the methods, are not optimally solved, but still good quality classifiers are constructed in all cases. In particular, OCT, OCT-H and OCTSVM finish the CPU time for training with large gaps, which indicates the computational complexity of the problems. In contrast, the gaps obtained with OCT+SVM are considerably smaller with respect to OCT-H and OCTSVM, but of the same size than OCT, which shows that the decision on relabeling observations makes the problem harder to solve.

Table 2 Average accuracies (± standard deviations) obtained in our computational experiments

	% FL	CART	OCT	OCT-H	OCT+SVM	OCTSVM	Diff
	0	84.91 ± 1.31	85.48 ± 0.99	85.16 ± 1.28	85.62 ± 1.08	**86.16 ±0.68**	0.68 ± 0.76
Australian	20	83.35 ± 5.30	**85.31 ±1.12**	80.05 ± 4.57	81.93 ± 7.83	85.25 ± 1.34	-0.06 ± 1.03
(690,14)	30	70.50 ± 11.89	79.27 ± 7.55	72.01 ± 4.25	75.24 ± 10.46	**82.57 ±7.46**	3.29 ± 11.96
	40	54.34 ± 7.94	70.36 ± 9.56	64.63 ± 4.25	64.38 ± 7.93	**77.92 ±8.19**	7.56 ± 11.71
	0	91.64 ± 2.09	87.94 ± 1.54	**98.60 ±0.47**	92.68 ± 4.86	96.06 ± 3.19	-2.54 ± 3.32
Banknote	20	88.00 ± 2.63	85.99 ± 1.91	70.01 ± 9.18	66.46 ± 10.01	**89.97 ±4.71**	3.97 ± 4.63
(1372,5)	30	85.79 ± 3.38	**85.96 ±2.27**	63.18 ± 9.91	67.36 ± 12.85	84.03 ± 12.46	-1.93 ± 12.72
	40	69.13 ± 10.33	77.73 ± 9.25	59.15 ± 8.53	58.45 ± 12.66	**87.16 ±12.46**	9.42 ± 8.67
	0	91.81 ± 1.51	93.71 ± 0.94	94.21 ± 1.06	**96.15 ±0.94**	95.09 ± 5.27	2.44 ± 1.07
BreastCancer	20	90.38 ± 2.63	**92.88 ±1.72**	89.45 ± 4.24	82.96 ± 5.29	91.85 ± 3.07	-1.03 ± 3.08
(683,9)	30	87.52 ± 3.38	**92.15 ±1.84**	84.41 ± 5.41	79.16 ± 4.94	89.90 ± 2.75	-2.24 ± 3.85
	40	73.95 ± 12.53	**90.44 ±2.68**	75.36 ± 6.81	74.58 ± 5.15	87.28 ± 4.71	-3.15 ± 5.26
	0	72.34 ± 3.57	75.02 ± 1.82	78.95 ± 2.29	82.31 ± 2.05	**83.36 ±1.41**	4.41 ± 1.88
Heart	20	70.53 ± 6.16	74.10 ± 3.57	74.66 ± 4.24	73.95 ± 5.98	**80.13 ±4.39**	5.46 ± 4.92
(270,13)	30	71.87 ± 3.72	71.87 ± 6.23	71.19 ± 5.61	73.7 ± 4.69	**77.25 ±6.31**	5.37 ± 7.24
	40	71.72 ± 3.63	64.18 ± 5.32	65.18 ± 5.04	66.82 ± 5.15	**74.26 ±3.39**	9.07 ± 4.94
	0	81.09 ± 4.73	85.34 ± 2.36	84.77 ± 1.97	83.15 ± 4.46	**85.55 ±2.65**	0.20 ± 2.70
Ionosphere	20	74.46 ± 6.77	**78.74 ±3.14**	74.88 ± 5.78	77.06 ± 3.63	78.05 ± 4.65	-0.68 ± 5.43
(351,34)	30	65.80 ± 6.76	74.30 ± 5.70	73.61 ± 4.47	74.84 ± 4.46	**76.83 ±5.21**	2.53 ± 7.27
	40	60.48 ± 7.39	70.89 ± 5.30	66.06 ± 6.69	71.73 ± 4.64	**75.68 ±2.62**	4.79 ± 5.77
	0	59.19 ± 3.04	60.30 ± 2.90	61.24 ± 2.40	60.48 ± 2.59	**62.85 ±1.72**	1.61 ± 2.90

Table 2 continued

	% FL	CART	OCT	OCT-H	OCT+SVM	OCTSVM	Diff
MONK's	20	55.40 ± 7.01	59.59 ± 3.42	58.96 ± 4.93	59.49 ± 1.77	**62.40 ±2.15**	2.80 ± 3.79
(415,17)	30	55.10 ± 4.88	57.90 ± 5.00	59.21 ± 2.91	58.73 ± 0.97	**60.57 ±4.67**	1.36 ± 4.93
	40	50.30 ± 5.99	56.86 ± 3.04	56.86 ± 3.61	58.84 ± 2.11	**61.91 ±2.30**	5.05 ± 4.41
	0	69.19 ± 8.95	76.09 ± 4.82	74.82 ± 3.18	80.66 ± 2.26	**81.67 ±4.26**	5.57 ± 4.26
Parkinson	20	61.95 ± 12.39	73.95 ± 4.34	66.40 ± 5.20	74.96 ± 4.69	**77.29 ±3.88**	3.33 ± 5.27
(240,40)	30	57.82 ± 10.61	71.05 ± 7.03	62.63 ± 6.46	70.27 ± 5.42	**75.57 ±3.70**	4.52 ± 6.56
	40	51.76 ± 8.18	67.95 ± 7.44	59.39 ± 3.98	62.38 ± 5.51	**69.59 ±5.10**	1.64 ± 9.46
	0	62.11 ± 10.86	69.17 ± 4.82	71.53 ± 4.69	74.15 ± 3.70	**76.86 ±3.86**	5.32 ± 6.16
Sonar	20	58.06 ± 9.03	64.41 ± 5.65	67.41 ± 4.07	75.21 ± 5.59	**71.62 ±4.29**	4.20 ± 4.84
(208,60)	30	53.35 ± 6.40	60.62 ± 4.78	65.11 ± 5.76	67.36 ± 7.27	**70.07 ±4.36**	4.96 ± 5.38
	40	50.04 ± 6.76	58.79 ± 5.18	60.24 ± 4.15	62.44 ± 4.13	**64.95 ±11.88**	4.71 ± 10.68
	0	89.61 ± 2.22	90.35 ± 1.28	**90.52** ± 1.27	87.54 ± 3.80	88.59 ± 1.82	-1.93 ±2.63
Wholesale	20	81.22 ± 10.12	**87.80 ±3.56**	85.14 ± 2.95	75.21 ± 5.59	81.63 ± 4.67	-6.17 ± 5.67
(440,7)	30	77.40 ± 12.03	**83.99 ±5.15**	78.01 ± 7.01	74.89 ± 6.35	78.95 ± 5.06	-5.03 ± 8.30
	40	61.46 ± 14.41	73.14 ± 16.11	72.31 ± 5.70	72.01 ± 10.52	**76.06 ±4.94**	3.76 ± 16.90

Table 3 Average AUC (± standard deviations) obtained in our computational experiments

	% FL	CART	OCT	OCT-H	OCT+SVM	OCTSVM	Diff
	0	85.31 ± 1.72	86.10 ± 0.93	85.18 ± 1.52	85.92 ± 1.16	**86.30 ±0.78**	0.20 ± 0.70
Australian	20	83.78 ± 5.81	**85.88 ±1.23**	80.33 ± 5.19	81.87 ± 9.42	85.52 ± 8.77	-0.35 ± 1.15
(690,14)	30	70.26 ± 12.49	79.44 ± 7.70	71.87 ± 4.32	74.89 ±11.96	**82.70 ±8.77**	3.26 ± 13.08
	40	53.98 ± 6.12	70.04 ± 10.26	64.91 ± 5.58	63.07 ± 9.80	**77.39 ±9.39**	7.35 ± 12.53
	0	91.37 ± 2.15	87.52 ± 1.47	**98.56 ±0.43**	90.84 ± 7.11	96.01 ± 3.23	-2.55 ± 3.34
Banknote	20	87.80 ± 2.69	85.87 ± 1.76	66.91 ± 10.95	63.38 ± 12.15	**89.47 ± 5.26**	3.60 ± 5.17
(1372,5)	30	85.33 ± 3.90	**85.54 ±2.40**	60.13 ± 11.04	64.97 ± 13.83	84.17 ± 11.63	-1.37 ± 11.84
	40	67.54 ± 12.06	76.64 ± 10.95	56.05 ± 8.14	55.68 ± 12.33	**85.85 ±10.02**	9.20 ± 9.98
	0	91.36 ± 1.92	93.10 ± 1.33	93.60 ± 1.54	**95.73 ±1.27**	94.20 ± 7.98	2.62 ± 1.41
BreastCancer	20	89.62 ± 2.86	**91.95 ±2.46**	88.52 ± 5.76	79.58 ± 8.69	90.32 ± 3.40	-1.09 ± 4.14
(683,9)	30	85.28 ± 8.18	**90.82 ±2.46**	82.96 ± 8.50	74.00 ± 7.94	87.06 ± 4.37	-3.75 ± 6.23
	40	67.61 ± 15.04	**88.90 ±3.96**	69.71 ± 10.28	67.46 ± 8.33	85.48 ± 7.57	-3.42 ± 9.10
	0	71.91 ± 2.89	74.49 ± 1.97	78.62 ± 2.19	82.17 ± 1.53	**82.87 ±1.21**	4.24 ± 1.96
Heart	20	69.88 ± 6.09	73.31 ± 3.42	74.24 ± 3.91	73.04 ± 6.08	**79.40 ±4.29**	5.15 ± 4.43
(270,13)	30	71.47 ± 3.04	71.16 ± 6.40	70.14 ± 5.82	72.75 ± 5.15	**73.97 ± 6.87**	4.73 ± 7.89
	40	71.45 ± 3.01	63.91 ± 5.34	64.07 ± 6.14	66.29 ± 5.85	**73.97 ±3.08**	9.89 ± 6.29
	0	78.29 ± 6.21	81.90 ±**3.84**	81.46 ± 2.04	77.49 ± 5.66	**82.38 ±3.71**	0.48 ± 4.26

Table 3 continued

	% FL	CART	OCT	OCT-H	OCT+SVM	OCTSVM	Diff
Ionosphere	20	69.22 ± 9.11	**74.39 ±3.87**	70.69 ± 6.64	73.18 ± 4.46	72.64 ± 7.39	-1.70 ±7.48
(351,34)	30	60.11 ± 8.73	70.20 ± 6.56	69.09 ± 4.62	70.62 ± 6.42	**71.37 ±5.07**	1.16 ± 7.39
	40	56.09 ± 7.09	68.23 ± 5.56	60.00 ± 7.84	64.32 ± 6.35	**69.44 ±3.51**	1.21 ± 6.80
	0	58.49 ± 4.06	59.83 ± 3.21	59.84 ± 3.85	59.90 ± 2.78	**60.92 ±2.46**	1.08 ± 3.16
MONK's	20	54.93 ± 6.45	59.06 ± 3.96	58.29 ± 3.85	57.51± 2.83	**60.68 ±2.36**	1.62 ± 4.50
(415,17)	30	53.56 ± 5.26	57.25 ± 5.98	57.85 ± 2.95	55.86 ± 2.91	**59.58 ±3.09**	1.73 ± 3.39
	40	49.51 ± 4.56	56.08 ± 3.41	55.67 ± 4.11	57.33 ± 2.57	**59.79 ±2.96**	4.12 ± 3.81
	0	69.58 ± 8.13	76.15 ± 1.77	74.95 ± 3.05	80.77 ± 2.19	**81.81 ±4.07**	5.66 ± 4.14
Parkinson	20	62.69 ± 11.63	74.08 ± 4.34	66.48 ± 5.27	75.13 ± 4.72	**77.37 ±3.47**	3.29 ± 5.34
(240,40)	30	58.15 ± 10.43	70.89 ± 7.57	62.57 ± 6.71	70.18 ± 5.54	**75.87 ±3.47**	4.97 ± 7.28
	40	51.94 ± 8.09	67.89 ± 7.61	59.36 ± 3.90	62.27 ± 5.98	**69.32 ±5.30**	1.42 ± 9.52
	0	62.11 ± 10.86	68.68 ± 4.65	71.19 ± 4.77	73.64 ± 4.25	**76.55 ±3.68**	5.35 ± 6.34
Sonar	20	58.06 ± 9.03	63.75 ± 6.20	67.46 ± 5.67	68.81 ± 4.86	**71.61 ±4.41**	4.15 ± 4.85
(208,60)	30	53.35 ± 6.40	59.88 ± 5.54	65.20 ± 5.67	66.65 ± 7.78	**69.64 ±4.42**	4.43 ± 5.10
	40	50.04 ± 6.76	58.00 ± 5.25	59.82 ± 4.98	62.05 ± 4.40	**64.99 ±11.18**	5.17 ± 10.17
	0	89.61 ± 2.22	88.82 ± 2.47	**88.94 ±3.02**	86.25 ± 3.58	87.36 ± 2.98	-1.57 ± 4.25
Wholesale	20	81.22 ± 10.12	**85.80 ± 5.37**	82.41 ± 5.53	67.10 ± 12.64	77.12 ± 7.94	-8.67 ± 7.50
(440,7)	30	77.40 ± 12.03	**81.80 ±8.12**	74.57 ± 10.13	67.99 ± 12.29	72.40 ± 9.12	-9.39 ± 13.03
	40	61.46 ± 14.41	**72.30 ±15.57**	64.72 ± 12.85	59.88 ± 10.52	69.66 ± 10.35	-2.64 ± 9.89

Table 4 Average MINLP Gaps of the optimization-based models to construct classification trees, within the time limit

	% FL	OCT	OCT-H	OCT+SVM	OCTSVM
	0	12.50	85.36	27.33	91.59
Australian	20	18.75	97.50	30.01	86.80
(690,14)	30	68.75	99.23	45.31	82.92
	40	62.50	99.66	35.84	85.53
	0	100.00	78.00	76.64	82.53
Banknote	20	68.75	99.99	75.16	93.65
(1372,5)	30	56.25	99.99	76.69	91.75
	40	43.75	99.70	73.77	85.20
	0	56.25	80.62	19.19	79.40
BreastCancer	20	56.04	87.50	54.07	78.17
(683,9)	30	81.09	94.81	37.54	81.35
	40	82.89	98.26	23.57	86.62
	0	66.93	88.86	29.55	68.34
Heart	20	54.68	93.78	45.09	81.01
(270,13)	30	54.97	93.22	36.63	90.98
	40	99.06	96.82	32.54	85.05
	0	68.33	75.00	47.23	84.15
Ionosphere	20	61.42	85.94	52.38	100.00
(351,34)	30	80.80	89.63	74.39	84.16
	40	75.00	92.42	46.26	78.52
	0	12.25	97.09	24.99	78.19
MONK's	20	37.22	97.03	18.68	68.73
(415,17)	30	24.77	99.75	12.09	90.95
	40	62.01	96.41	21.62	83.78
	0	34.52	76.15	36.61	89.07
Parkinson	20	55.92	79.91	40.15	85.52
(240,40)	30	43.75	87.07	40.79	97.06
	40	36.97	88.49	41.49	84.54
	0	50.00	75.63	19.88	77.73
Sonar	20	68.75	81.94	39.45	79.17
(208,60)	30	68.75	83.00	30.48	88.75
	40	75.00	78.62	38.71	91.91
	0	31.25	38.67	28.71	32.34
Wholesale	20	30.88	96.20	14.22	94.26
(440,7)	30	31.25	95.07	7.39	91.25
	40	75.00	88.35	6.34	97.52

5 Conclusions and further research

Supervised classification is a fruitful field that has attracted the attention of researchers for many years. One of the methods that has experienced a more in depth transformation in the last years is classification trees. Since the pioneer contribution by Breiman et al. Breiman et al. (1984), where CART was proposed, this technique has included tools from mathematical optimization giving rise to the so called OCT Bertsimas and Dunn (2017); Bertsimas et al. (2019), methods that are *optimal* in some sense. In spite of that, there is still some extra room for further improvements, in particular making classifiers more robust against perturbed datasets. Our contribution is this paper goes in that direction and it augments the catalogue of classification tree methods able to handle noise in the labels of the dataset.

We have proposed a new optimal classification tree approach able to handle label noise in the data. Two main elements support our approach: the splitting rules for the classification trees are designed to maximize the separation margin between classes and wrong labels of the training sample are detected and changed at the same time that the classifier is built. The method is encoded on a Mixed Integer Second Order Cone Optimization problem so that one can solve medium size instances by any of the nowadays available off-the-shelf solvers. We report intensive computational experiments on a battery of datasets taken from UCI Machine Learning repository showing the effectiveness and usefulness of our approach.

Future research lines on this topic include the analysis of nonlinear splits when branching in OCTSVM, both using kernel tools derived from SVM classifiers or specific families of nonlinear separators. This approach will result into more flexible classifiers able to capture the nonlinear trends of many real-life datasets. Additionally, we also plan to address the design of math-heuristic algorithms which keep the essence of OCTs but capable to train larger datasets and the analysis of a multiclass version of our approach.

Acknowledgements This research has been partially supported by Spanish Ministerio de Ciencia e Innovación, Agencia Estatal de Investigación/FEDER grant number PID2020-114594GBC21, Junta de Andalucía projects P18-FR-1422, P18-FR-2369 and projects FEDERUS-1256951, BFQM-322-UGR20, CEI-3-FQM331 and NetmeetData-Ayudas Fundación BBVA a equipos de investigación científica 2019. The first author was also partially supported by the IMAG-Maria de Maeztu grant CEX2020-001105-M /AEI /10.13039/501100011033.

Funding Funding for open access charge: Universidad de Granada / CBUA

References

Agarwal N, Balasubramanian VN, Jawahar C (2018) Improving multiclass classification by deep networks using dagsvm and triplet loss. Pattern Recogn Lett 112:184–190

Benati S, Puerto J, Rodríguez-Chía AM (2017) Clustering data that are graph connected. Eur J Oper Res 261(1):43–53

Benati S, Ponce D, Puerto J, Rodríguez-Chía AM (2021) A Branch-and-price procedure for clustering data that are graph connected. Eur J Oper Res. https://doi.org/10.1016/j.ejor.2021.05.043

Bennett, K. P., and Blue, J. A support vector machine approach to decision trees. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)* (1998), vol. 3, IEEE, pp. 2396–2401

Bertsimas D, Dunn J (2017) Optimal classification trees. Mach Learn 106(7):1039–1082

Bertsimas D, Dunn J, Pawlowski C, Zhuo YD (2019) Robust classification. INFORMS Journal on. Optimization 1(1):2–34

Bertsimas, D., and Dunn, J. W. *Machine learning under a modern optimization lens*. Dynamic Ideas LLC, 2019

Blanco V, Japón A, Ponce D, Puerto J (2020) On the multisource hyperplanes location problem to fitting set of points. Computers & Operations Research 105124

Blanco, V., Japón, A., and Puerto, J. A mathematical programming approach to binary supervised classification with label noise. arXiv preprint arXiv:2004.10170 (2020)

Blanco V, Japón A, Puerto J (2020c) Optimal arrangements of hyperplanes for svm-based multiclass classification. Adv Data Anal Classif 14(1):175–199

Blanco V, Puerto J, Salmerón R (2018) Locating hyperplanes to fitting set of points: A general framework. Computers Operat Res 95:172–193

Blanquero R, Carrizosa E, Jiménez-Cordero A, Martín-Barragán B (2020a) Selection of time instants and intervals with support vector regression for multivariate functional data. Comput Operat Res 123:105050

Blanquero R, Carrizosa E, Ramírez-Cobo P, Sillero-Denamiel MR (2020b) A cost-sensitive constrained lasso. Adv Data Analys Classification 1–38

Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. CRC Press, United States

Carrizosa E, Molero-Río C, Morales DR (2021) Mathematical optimization in classification and regression trees. TOP 29(1):5–33

Cortes C, Vapnik V (1995) Support-vector networks. Machine learn 20(3):273–297

Cover T, Hart P (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13(1):21–27

Drucker H, Burges CJ, Kaufman L, Smola AJ, Vapnik V (1997) Support vector regression machines. In Advances in neural information processing systems 155–161

Dua, D., and Graff, C. UCI machine learning repository, 2017

Frénay B, Verleysen M (2013) Classification in the presence of label noise: a survey. IEEE Trans Neural Netw Learn Syst 25(5):845–869

Friedman, J., Hastie, T., and Tibshirani, R. *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001

Gaudioso M, Gorgone E, Labbé M, Rodríguez-Chía AM (2017) Lagrangian relaxation for svm feature selection. Computers Operat Res 87:137–145

Günlük O, Kalagnanam J, Li M, Menickelly M, Scheinberg K (2021) Optimal decision trees for categorical data via integer programming. J Glob Optim 81:233–260

Guzella TS, Caminhas WM (2009) A review of machine learning approaches to spam filtering. Expert Syst Appl 36:10206–10222

Lewis, D. D. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning* (1998), Springer, pp. 4–15

Quinlan J (1996) Machine learning and id3. Morgan Kauffman, Los Altos

Quinlan, R. C4. 5. *Programs for machine learning* (1993)

Tang X, Xu A (2016) Multi-class classification using kernel density estimation on k-nearest neighbours. Electron Lett 52(8):600–602

Weerasinghe S, Erfani SM, Alpcan T, Leckie C (2019) Support vector machines resilient against training data integrity attacks. Pattern Recognition 96:106985

Yu B, Xu ZB (2008) A comparative study for content-based dynamic spam classification using four machine learning algorithms. Knowl-Based Syst 21:355–362