Beyond the ROC Curve: Classification Trees Using Cost-Optimal Curves, with Application to Imbalanced Datasets

Magzhan Gabidolla [1] **Arman Zharmagambetov** [2] **Miguel Á. Carreira-Perpiñán** [1]

Abstract

Important applications such as fraud or spam detection or churn prediction involve binary classification problems where the datasets are imbalanced and the cost of false positives greatly differs from the cost of false negatives. We focus on classification trees, in particular oblique trees, which subsume both the traditional axis-aligned trees and logistic regression, but are more accurate than both while providing interpretable models. Rather than using ROC curves, we advocate a loss based on minimizing the false negatives subject to a maximum false positive rate, which we prove to be equivalent to minimizing a weighted 0/1 loss. This yields a curve of classifiers that provably dominates the ROC curve, but is hard to optimize due to the 0/1 loss. We give the first algorithm that can iteratively update the tree parameters globally so that the weighted 0/1 loss decreases monotonically. Experiments on various datasets with class imbalance or class costs show this indeed dominates ROC-based classifiers and significantly improves over previous approaches to learn trees based on weighted purity criteria or over- or undersampling.

1. Introduction

Many important practical applications involve a binary classification problem with imbalanced classes (e.g. few positives and many negatives) or asymmetric costs (e.g. a false positive is much more costly than a false negative), where the positives and negatives are suitably defined in each case. Examples are fraud or spam detection or churn prediction. Although many types of classifiers may be used, we focus on classification trees, which are widely recognized as among the most interpretable models. We consider the traditional axis-aligned trees (where each decision node uses a single feature) and also sparse oblique trees (where each decision node uses a linear combination of a small subset of features). The latter are far more powerful and subsume as particular cases axis-aligned trees (such as CART or C5.0) and linear classifiers (such as logistic regression or linear SVMs). In both cases each leaf node outputs a constant label.

In these types of problems, optimizing the raw accuracy does not work well because it can largely ignore the low-cost or infrequent class. It is desirable to have control on the number of false positives or true positives. A popular way to achieve this is through the ROC curve, but this results in suboptimal classifiers because it does not explicitly optimize for accuracy or true positives while controlling the false positive rate.

Our paper has two contributions. Firstly, in section 3, we formally propose the concept of *cost-optimal curve (COC)*. This defines a set of optimal-accuracy classifiers as a function of the false positive level. We prove this is equivalent to a penalized formulation which has the form of a weighted 0/1 loss and (with our new algorithm) is more amenable to optimization, although still NP-hard in general. Although the COC idea is straightforward, its properties and optimization appear not to have been explored before. Note this is different from using a weighted surrogate loss such as the cross-entropy; while this (being differentiable) can be easily optimized, its optimum can be quite far from the true (0/1 loss) one. Second, in section 4, we propose the first algorithm that directly tries to optimize this problem for classification trees, with the guarantee that the weighted 0/1 loss decreases monotonically at each iteration. In section 5, our experiments confirm that the algorithm provides a good approximation to the ideal COC curve and is generally much better than using the ROC curve or approaches based on over- or undersampling, cost-sensitive surrogates or weighted purity criteria for constructing the tree.

[1]Dept. of Computer Science and Engineering, University of California, Merced, USA. [2]Meta AI (FAIR). Correspondence to: Miguel Á. Carreira-Perpiñán <mcarreira-perpinan@ucmerced.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

2. Related Work

Imbalanced learning has been studied extensively (Branco et al., 2016; Fernández et al., 2018; He & Ma, 2013; Japkowicz & Shah, 2011; Sun et al., 2009). As the distribution of classes is significantly skewed, it is challenging for standard classifiers to learn and predict the minority class accurately. Several techniques have been proposed to address this. One commonly used approach is resampling (Sun et al., 2009; Fernández et al., 2018), which aims to rebalance the class distribution by either oversampling the minority class or undersampling the majority class. More advanced sampling techniques exist, such as SMOTE (Chawla et al., 2002) and ADASYN (He et al., 2008), that are capable of generating new synthetic samples to increase the representation of the minority class. Apart from sampling, evaluation metrics play a crucial role in assessing the performance of imbalanced learning algorithms. Accuracy alone may not be an appropriate metric, as it can be misleading due to the class imbalance. Instead, metrics like precision, recall, F1–score, and AUC–ROC are commonly used to evaluate the performance of classifiers in imbalanced learning scenarios (Japkowicz & Shah, 2011). Another set of techniques focuses on modifying the classification algorithms to handle imbalanced datasets more effectively. Cost-sensitive learning assigns different misclassification costs to different classes, encouraging the classifier to prioritize correct classification of the minority class (Branco et al., 2016; Fernández et al., 2018; He & Ma, 2013). However, this does not use the 0/1 loss as in eq. (2), but a surrogate loss (e.g. the cross-entropy), which means one does not optimize the true positives given a false positive rate.

In the context of decision trees, resampling techniques are still applicable, as they are model agnostic. Another standard approach is to obtain an ROC curve and perform model selection (Cook & Goldman, 1984). Some methods propose to minimize directly a desired objective such as AUC–ROC, F1–score, etc. However, this is challenging since the optimization problem with such metrics is difficult even with linear models. Therefore, previous works either rely on greedy recursive partitioning (Ferri et al., 2002; Gajowniczek & Ząbkowski, 2021) (similar to CART (Breiman et al., 1984)) or are not scalable, as they solve the problem by brute-force search with branch-and-bound (Lin et al., 2020). Cost-sensitive learning for decision trees has also been studied in various settings (Elkan, 2001; Höppner et al., 2022) but limited to traditional axis-aligned trees trained via greedy recursive partitioning. The works most closely related to our cost-optimal curve (COC) are Drummond & Holte (2000); Raubertas et al. (1994), who propose to construct a true-positive versus false-positive curve by learning a new decision tree for each class-specific cost matrix. This is a form of cost-sensitive learning but, instead of using a surrogate (not 0/1) loss, each tree itself is grown using a CART-style greedy procedure. This uses a local purity criterion (the Gini index) to split nodes but does not optimize any global loss function over the tree parameters and is highly suboptimal (Hastie et al., 2009). Thus, unlike us, each tree's problem is not defined by optimizing the true positives given a false positive rate, or a weighted 0/1 loss of both.

A fundamental innovation in our paper, on which the entire concept of the COC curve relies, is the ability to optimize a well-defined objective function (a weighted 0/1 loss), specifically over decision trees. This is not possible with the traditional framework based on greedy recursive partitioning (Breiman et al., 1984; Quinlan, 1993), which does not even define an overall loss function over a space of trees. Also, we want to be able to learn more complex and powerful types of trees, beyond the traditional axis-aligned ones. To be able to do this, we rely on the framework of the Tree Alternating Optimization (TAO) algorithm (Carreira-Perpiñán & Tavallali, 2018) (see section 4). TAO and its variations are able to reduce monotonically over iterations an objective function of quite general form until convergence over the parameters of a fixed-structure tree. This results in smaller, more accurate trees (Zharmagambetov et al., 2021), and has been used successfully to train trees for a number of tasks based on precisely defined loss functions, such as clustering (Gabidolla & Carreira-Perpiñán, 2022b), dimensionality reduction (Zharmagambetov & Carreira-Perpiñán, 2022a) or semi-supervised learning (Zharmagambetov & Carreira-Perpiñán, 2022b). These benefits also carry over to decision forests, whether using bagging, (gradient) boosting or direct optimization over all the trees (Carreira-Perpiñán & Zharmagambetov, 2020; Zharmagambetov & Carreira-Perpiñán, 2020; Gabidolla et al., 2022; Gabidolla & Carreira-Perpiñán, 2022a; Carreira-Perpiñán et al., 2023). Here, we extend TAO to handling COC curves for a single classification tree.

3. The ROC Curve and the Cost-Optimal Curve (COC)

3.1. The ROC Curve

Throughout the paper, we will write TP and FP to mean true and false positive rate, respectively, and likewise TN and FN for true and false negative rate; and define the ROC (and COC) space as points (FP,TP). Assume we have trained a base classifier which outputs the probability that an input instance $\mathbf{x}$ is in the positive class, $p(y = +1|\mathbf{x}) \in [0, 1]$. (For classification trees with constant-label leaves this is usually achieved by labeling each leaf with the proportion of instances that are positive from among the instances that

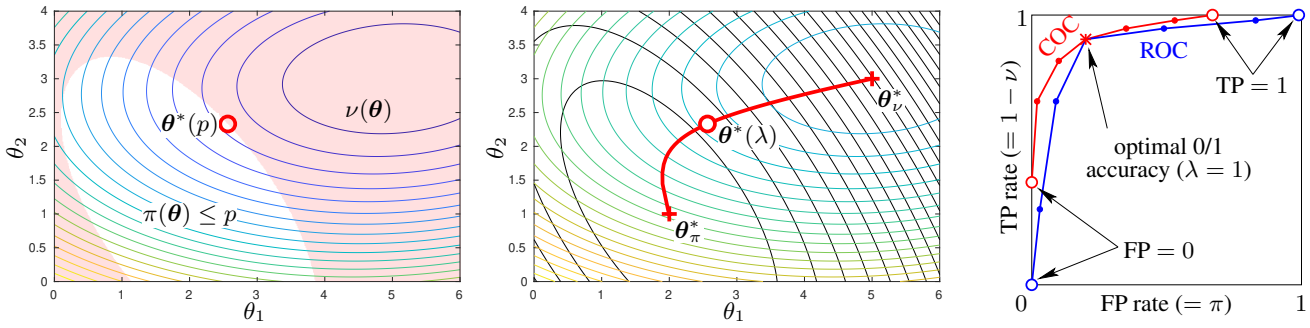


Figure 1. Illustration of the COC curve for a classifier with parameters $\boldsymbol{\theta} \in \mathbb{R}^2$, FP rate $\pi(\cdot)$ and FN rate $\nu(\cdot)$. *Left*: $\boldsymbol{\theta}^*(p)$ is an optimal classifier (minimizing ν) with an FP rate of at most p. The infeasible set is in pink. *Middle*: the optimal classifier path $\boldsymbol{\theta}^*(\lambda)$ over the cost λ, i.e., minimizing $\nu + \lambda\,\pi$, from $\boldsymbol{\theta}^*_\nu = \boldsymbol{\theta}^*(0)$ to $\boldsymbol{\theta}^*_\pi = \boldsymbol{\theta}^*(\infty)$. The contours of ν and π are in color and black, respectively. Note that, for ease of visualization, we plot the contours and path as continuous; in reality, π and ν are discrete (0/1 loss). *Right*: the COC curve corresponding to the optimal classifier path and the ROC curve (assuming as base classifier that for $\lambda = 1$).

reach the leaf.) Typically, but not necessarily, this classifier was trained to optimize the accuracy on the training set. As is well known, the ROC curve is obtained by postprocessing this classifier through a threshold $t \in [0, 1]$ (considered as a free parameter) so that we predict the positive class if $p(y = +1|\mathbf{x}) > t$. Over a training set with N points this defines a set of at most $N + 1$ classifiers, each corresponding to an ROC point (FP,TP). This includes two constant classifiers: one (for $t = 0$) predicts always the positive class regardless of $\mathbf{x}$ and is at ROC point (1,1); the other (for $t = 1$) predicts always the negative class and is at (0,0). These two classifiers and likely others for t near 0 or 1 are obviously useless, as they ignore one class. This happens even if the base classifier achieves a perfect accuracy of 1; the ROC curve deteriorates it as we vary t away from $\frac{1}{2}$.

The ROC curve is a fast, simple way to achieve a classifier with lower overall accuracy but a more desirable FP rate, particularly in cases with imbalanced classes or asymmetric class costs. However, it is clear that (except possibly for the base classifier) *this does not produce a classifier that, having the desired FP rate, is optimal within its model class*. To obtain this we need to solve a different optimization problem as a function of the FP rate, as we describe next. This will then produce a different curve that we call *cost-optimal curve (COC)*.

3.2. The Cost-Optimal Curve (COC)

Assume a training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ where $\mathbf{x}_n \in \mathcal{X}$ and $y_n \in \{-1, +1\}$ for $n = 1, \dots, N$, and a classifier $T(\cdot; \boldsymbol{\theta})$: $\mathcal{X} \to \{-1, +1\}$ with parameters $\boldsymbol{\theta}$. We define as our desired classifier one that minimizes the FN rate[1] given that the FP rate is at most $p \in [0, N]$:

$$\min_{\boldsymbol{\theta}} \ \nu(\boldsymbol{\theta}) \quad \text{s.t.} \quad \pi(\boldsymbol{\theta}) \le p \qquad \text{with} \tag{1}$$

$$\nu(\boldsymbol{\theta}) = \sum_{n:\ y_n = +1}^{N} L(y_n, T(\mathbf{x}_n; \boldsymbol{\theta})),\ \pi(\boldsymbol{\theta}) = \sum_{n:\ y_n = -1}^{N} L(y_n, T(\mathbf{x}_n; \boldsymbol{\theta}))$$

where $L(y, y') = 0$ if $y = y'$ and 1 if $y \neq y'$ is the 0/1 loss, and ν and π are the FN and FP rate, respectively. Note $\nu(\boldsymbol{\theta}) + \pi(\boldsymbol{\theta}) = \sum_{n=1}^N L(y_n, T(\mathbf{x}_n; \boldsymbol{\theta}))$ is the 0/1 loss on the whole training set. Also, usually the problem includes a regularization term on $\boldsymbol{\theta}$, but we ignore this to keep the notation simple.

This is a constrained optimization problem whose feasible set are those classifiers having a FP rate of at most p (fig. 1 (left)). By solving (1) for all $p \in [0, N]$ we obtain a finite collection of at most $N + 1$ classifiers, each for a different FP and FN (hence TP) rate, which defines the COC curve. The following properties arise immediately:

- The COC curve dominates the ROC curve (or indeed any other curve using the same classifier family). That is, for any point (FP,TP) on the ROC curve there exists another point (FP',TP') on the COC curve with FP' $\le$ FP and TP' $\ge$ TP. This follows directly from (1). See fig. 1 (right).
- The extreme points of the COC curve are at $(\alpha, 1)$ and $(0, \beta)$ where $\alpha < 1$ and $\beta > 0$, unlike for the ROC curve (where $\alpha = 1$ and $\beta = 0$). This follows by noting that we can always classify one class perfectly while also classifying some of the other class' points correctly, if we optimize as in (1). Thus, we expect the advantage of the COC curve to be larger in regions of high TP or low FP (which are often the most important regimes in practice).

Another advantage of the COC curve over the ROC one is that it does not require the classifier to output probabilities; all the classifier needs to do is predict a class.

[1] Although we use the term "rate", in the equations, to keep them simpler, we use the absolute count. The normalizing factors (number of positives and negatives in the training set) can be absorbed into p in eq. (1) or λ in eq. (2).

3.3. Cost-Sensitive (Weighted) 0/1 Loss

Consider now the following unconstrained optimization problem[2] where $\lambda \geq 0$:

$$\min_{\boldsymbol{\theta}} \ \nu(\boldsymbol{\theta}) + \lambda\, \pi(\boldsymbol{\theta}). \tag{2}$$

This objective function is a *weighted 0/1 loss*: it gives a misclassification cost of 1 to a FN and of λ to a FP. The usual misclassification error (unweighted 0/1 loss) results for $\lambda = 1$. In the Appendix, we prove that problems (1) and (2) have the same set of solutions, i.e., solving (1) for all $p \in [0, N]$ produces the same set of classifiers as solving (2) for all $\lambda \geq 0$ and hence the same COC curve. However, computationally it is easier to solve (2) than (1) in our case (see section 4). Note that, if the base classifier used to construct the ROC curve was trained to minimize the 0/1 loss (i.e., eq. (2) with $\lambda = 1$), then the ROC curve would touch the COC curve for $\lambda = 1$, but the COC curve would otherwise dominate. See fig. 1 (middle and right).

It is important to differentiate this from traditional work on cost-sensitive learning (e.g. Höppner et al., 2022; Vanderschueren et al., 2022 using logistic regression). The latter also uses a loss function where one class has a different weight, as in (2). But *the loss function is the cross-entropy or other surrogate loss, not the 0/1 loss* (and what the ROC space reports are 0/1 loss TP and FP), hence the result does not yield a COC curve.

Computationally, the ROC approach is fast because it trains a single classifier, while the COC one trains one classifier per value of λ. However, the training time can be significantly reduced by using warm-start, i.e., solving (2) for a sequence of λ values and initializing each from the previous one's solution.

Just as with the ROC curve, we can define the area under the curve (AUC) as a summary statistic for the COC curve. But note that both the ROC and COC curves are really finite sets of (at most $N + 1$) points, Thus, their "area" is not well defined, particularly if they consist of few points, as happens with trees. Indeed, for a classification tree with L leaves, the ROC curve cannot have more than $L + 1$ points. The COC curve for trees does not have such a limit because each COC point corresponds to a different tree.

3.4. Ideal vs Approximate COC Curve

In practice, optimizing (1) or (2) exactly is generally impractical because the 0/1 loss typically defines NP-hard problems. This is true for trees and linear classifiers, for example. One can still use an approximate optimization such as a surrogate loss (e.g. the cross-entropy or hinge loss for linear classifiers), or—better—our algorithm in section 4. This will result in an approximate COC curve, which will be dominated by the ideal COC. Likewise, the base ROC classifier will be approximate. The approximate COC curve should dominate the ROC curve because the former still tries to minimize the FN for any FP rate.

[2] We could also define (1) as $\min_{\boldsymbol{\theta}} \nu(\boldsymbol{\theta}) + \pi(\boldsymbol{\theta})$ s.t. $\pi(\boldsymbol{\theta}) \leq p$, i.e., maximize the accuracy rather than the TP rate. But turning this into a penalized problem $\min_{\boldsymbol{\theta}} \nu(\boldsymbol{\theta}) + \pi(\boldsymbol{\theta}) + \lambda\, \pi(\boldsymbol{\theta})$ shows it to be equivalent to (2) but using a penalty parameter $1 + \lambda$.

3.5. The COC Curve on a Test Set

Throughout the previous discussion we have assumed a training set on which problems (1) and (2) are defined. Will the COC curve dominate the ROC curve on a test set and hence lead to better generalization? Under the usual assumption in machine learning that the training and test sets come from the same distribution, one would expect so, and this is confirmed in our experiments.

4. Optimizing a Cost-Sensitive 0/1 Loss over a Classification Tree

To realize the advantages of the COC curve one needs to optimize (2)*, which uses a weighted 0/1 loss, as best as possible*. Traditionally, as in CART and C5.0, decision trees have been trained in a heuristic way by greedy recursive partitioning. This sets a decision node's split by optimizing a local "purity" criterion (such as the Gini index or information gain), but the overall procedure does not optimize any loss over the entire tree. Likewise, while the local criterion can include class weights, this does not optimize an overall class-weighted loss (0/1 or otherwise). Recently, an algorithm has been proposed (*Tree Alternating Optimization (TAO)*) (Carreira-Perpiñán & Tavallali, 2018) which does optimize a global loss over a parametric tree (axis-aligned or oblique). Here, we extend TAO to handle a weighted 0/1 loss objective such as that in (2).

The core idea of TAO is to take a parametric tree of fixed structure and perform optimization steps in turn over the parameters of a single node while keeping the rest of the parameters fixed. This succeeds because of the three theorems that we describe below. It works quite similar to how one would optimize a neural network, but instead of gradients (which do not apply) TAO uses alternating optimization on a fixed tree structure. This results in iteratively updating all the parameters in the tree (decision node hyperplanes and leaf class labels), with a monotonic decrease of the objective function at each iteration over all nodes and convergence to a local optimum.

More formally, let $T(\cdot; \boldsymbol{\Theta})$: $\mathcal{X} \rightarrow \{-1, +1\}$ be a decision tree of a given, predetermined structure of depth Δ with learnable parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{i \in \mathcal{D}} \cup \{\theta_i \in \{-1, 1\}\}_{i \in \mathcal{L}}$, where $\mathcal{D}$ are the set of decision nodes and $\mathcal{L}$ are the set of leaf nodes. The predictive function of a tree $T(\mathbf{x}; \boldsymbol{\theta})$

works by routing $\mathbf{x}$ from the root to exactly one leaf and outputting the leaf's class label. The routing function at a decision node i works by applying the decision function $g_i(\mathbf{x};\boldsymbol{\theta}_i)$: $\mathcal{X} \rightarrow \{\texttt{left}_i, \texttt{right}_i\} \subset \mathcal{D} \cup \mathcal{L}$, which for oblique splits $\boldsymbol{\theta}_i = \{\mathbf{w}_i, w_{i0}\}$ mean "go to the left child if $\mathbf{w}_i^T\mathbf{x} + w_{i0} < 0$, else go to the right child". Axis-aligned trees are a special case of oblique trees, for which $\mathbf{w}_i$ is all zeros but one at the selected feature index and $-w_{i0}$ serves as a threshold value. The objective function for TAO in the context of cost-sensitive learning is to minimize a weighted 0/1 loss:

$$E(\boldsymbol{\Theta}) = \sum_{n:\ y_n=+1}^{N} L(y_n, T(\mathbf{x}_n;\boldsymbol{\Theta})) + \lambda \sum_{n:\ y_n=-1}^{N} L(y_n, T(\mathbf{x}_n;\boldsymbol{\Theta})) + \alpha \sum_{i\in\mathcal{D}} \phi_i(\boldsymbol{\theta}_i). \quad (3)$$

We add a regularization term $\phi(\cdot)$ to decision node weights, specifically ℓ_1 norm to promote sparse oblique splits. The hyperparameter $\alpha \geq 0$ then controls the strength of the regularization.

We now show how we extend the TAO algorithm to handle this cost-sensitive 0/1 loss problem. The algorithm is based on 3 theorems. The first theorem of the original TAO states the **separability condition**: the objective function (3) separates over any set of non-descendant nodes (e.g. all nodes at the same depth), and thus those can be optimized independently and in parallel. This follows from the fact the tree makes hard decisions and that the loss is separable over individual points. Our cost-sensitive problem is no different in this respect, and we directly use this theorem without any modification.

For presenting the next two theorems, the following definitions are helpful:

- *Reduced set* $\mathcal{R}_i \subset \{1,\dots,N\}$ of node i (decision node or leaf) is the training instances that reach node i given the current tree parameters.
- *Care points* $\mathcal{C}_i \subset \mathcal{R}_i$ of a decision node i are the training instances in the reduced set $\mathbf{x}_n \in \mathcal{R}_i$ for which $T_{\texttt{left}_i}(\mathbf{x}_n;\boldsymbol{\Theta}_{\texttt{left}_i}) \neq T_{\texttt{right}_i}(\mathbf{x}_n;\boldsymbol{\Theta}_{\texttt{right}_i})$ where $T_c(\cdot;\boldsymbol{\Theta}_c)$ is the predictive function for the subtree rooted at node c given the current tree parameters. In other words, care points for a given decision node consists of training instances in its reduced set for which one of its children misclassifies the instance while the other child classifies it correctly.

Theorem 4.1 (Reduced problem over a decision node)**.** *Consider the objective function $E(\boldsymbol{\Theta})$ of eq. (3) and a decision node i. Assume the parameter values $\boldsymbol{\Theta} \setminus \{\boldsymbol{\theta}_i\}$ of all the nodes except i are fixed. Then, as a function of $\boldsymbol{\theta}_i$, eq. (3) can be reduced to the following cost-sensitive binary classification problem:*

$$E_i(\boldsymbol{\theta}_i) = \sum_{n\in\mathcal{C}_i:\ y_n=+1} L(\overline{y}_{in}, g_i(\mathbf{x}_n;\boldsymbol{\theta}_i)) + \lambda \sum_{n\in\mathcal{C}_i:\ y_n=-1} L(\overline{y}_{in}, g_i(\mathbf{x}_n;\boldsymbol{\theta}_i)) + \alpha\,\phi_i(\boldsymbol{\theta}_i) \quad (4)$$

where $\overline{y}_{in} = \arg\min_{c\in\{\texttt{left}_i,\texttt{right}_i\}} L(y_n, T_c(\mathbf{x}_n;\boldsymbol{\Theta}_c))$ is the child of i which gives the correct prediction for the instance n.

This follows from the fact that all a decision node can do with an instance is to send it down its left or right child, and the ideal choice is the one that gives the correct prediction. It is interesting to observe that although this problem looks similar in notation to the high level objective function (3), it is not a class-based cost sensitive, but more of an instance-based cost sensitive. This is because both a negative and a positive label point might select the same child as "best" which already gives two different costs for one class, thus making the problem not a class-based cost sensitive.

For oblique nodes this problem is NP-hard but can be well approximated with a convex surrogate; we use ℓ_1-regularized logistic regression where each instance incurs some cost and we solve it using LIBLINEAR (Fan et al., 2008). We can guarantee a monotonic decrease in the objective by only accepting this update if it improves over the previous step.

Theorem 4.2 (Reduced problem over a leaf)**.** *Consider the objective function $E(\boldsymbol{\Theta})$ of eq. (3) and a leaf node i. Assume the parameter values $\boldsymbol{\Theta} \setminus \{\theta_i\}$ of all the nodes except i are fixed. Then, as a function of θ_i, eq. (3) can be reduced to the following simple problem:*

$$E_i(\theta_i) = \sum_{n\in\mathcal{R}_i:\ y_n=+1} L(y_n,\theta_i) + \lambda \sum_{n\in\mathcal{R}_i:\ y_n=-1} L(y_n,\theta_i) \quad (5)$$

where $\mathcal{R}_i$ is the reduced set of the leaf i. The optimal solution $\theta_i^* = 1$ if $|\mathcal{R}_i^+| \geq \lambda\,|\mathcal{R}_i^-|$ else -1, where $|\mathcal{R}_i^+|$ ($|\mathcal{R}_i^-|$) is the number of points in the positive (negative) class in the reduced set $\mathcal{R}_i$ of the leaf i.

Since our leaves are just constant label predictors, the solution to this problem is simply a weighted majority vote, i.e. a class label with largest cost in the reduced set. We provide the proofs of these theorems in Appendix C along with the pseudocode of the overall TAO algorithm with COC in figs. 6 and 7 of Appendix.

To obtain COC curves we train several decision trees with TAO for differing cost of false positives λ. To obtain a more stable and smoother path of trees we follow this procedure: first we train a base model by setting λ such that the total

cost of FPs and FNs are equal. Then from this base model we follow two COC paths: one by increasing the λ and the other by decreasing it using the following schedules: $\lambda \leftarrow \lambda\,\beta$ and $\lambda \leftarrow \lambda/\beta$, respectively, where $\beta > 1$ controls how fast we move. In each step of the path we warm-start from the previous tree which accelerates the training and helps to produce smoother COC curves. The increasing λ path stops when the FP-rate $= 0$, and similarly for the decreasing λ path when the FN-rate $= 0$.

4.1. Multiclass Problems

Although we focus on binary classification problems, the algorithm can be straightforwardly extended to the multiclass case. Instead of a single cost of FPs λ, the objective function in eq. (3) would contain K different types of costs $\{\lambda_k\}_{k=1}^{K}$ for each type of error.

4.2. Computational Complexity

This is dominated by the decision node optimization, which we solve using logistic regression. Assuming this is linear on the sample size, training all the decision nodes at the same depth is approximately constant and equal to training one logistic regression on the whole training set. Thus, the total sequential cost of one iteration is roughly equal to that of Δ logistic regressions on the whole dataset. As noted before, all the nodes at the same depth can be trained in parallel.

5. Experiments

5.1. Illustration of the Optimality of the COC Curve

We first illustrate the optimality of exact COC curves over its approximations and over traditional ROC curves. Because obtaining these exact optimal curves amounts to solving a weighed 0/1 loss problem and since this is generally NP-hard for all nontrivial ML models, for demonstration purposes we choose two simple cases: an oblique tree of depth 1 on a toy problem and an axis-aligned decision stump on a real dataset.

For binary classification problems, a depth 1 oblique tree can be equivalently considered as a linear classifier. Optimizing a 0/1 loss with a linear model is still an NP-hard problem, but for very simple toy datasets we can formulate it as a mixed integer programming and rely on existing MIO solvers. We create a non-linearly separable toy 2D balanced binary classification dataset of 100 points sampled from two Gaussian distributions, and use an MIO solver to obtain a range of linear models by varying the cost of FPs λ. In the left part of fig. 2 we plot their (FP,TP) rate result which by definition generates the COC curve. We also plot the models resulting from optimizing a weighed cross-entropy loss, i.e. logistic regression, which is a widely used surrogate. We can clearly see how this surrogate is consistently inferior to the optimal one for all ranges of FP-rate. In the figure we can also observe how the traditional ROC curve obtained by varying the threshold in logistic regression is also below the COC, but approximately matches the weighted cross-entropy. This emphasizes how optimizing a weighted 0/1 loss is important to obtain models with better TP/FP rate than the convex surrogates. Note also how the optimal COC curve reaches the TP-rate of 1 or FP-rate of 0 faster than the ROC curve, i.e. it touches the left and top axes well before the corner points.

A decision stump, i.e. an axis-aligned tree of depth 1 is perhaps one of the simplest models in ML, yet for many simple datasets it achieves decent performance (Holte, 1993). Here we use them to illustrate the importance of COC curves, because unlike other models in ML, for this simple case an exact solution can be found by simple enumeration. For a relatively imbalanced email spam dataset of about 5000 points we train a series of decision stumps for differing cost of FPs λ and plot the corresponding exact COC curve in the middle part of fig. 2. For the base model we also plot the ROC curve which consists of only 3 points: one is the corresponding COC, and two others are the corner points in the ROC space. Instead of using a weighed 0/1 loss as our splitting criterion, we can also employ a weighted Gini index used in the traditional CART algorithm to obtain an approximate COC plot. Although CART uses this splitting criterion for other reasons, in the case of a decision stump we use it to again illustrate that other surrogates for the weighted 0/1 loss produces inferior results in terms of TP/FP rate. Even for this very simple model we can clearly observe the gap between COC and its approximations.

We now illustrate the performance of COC for oblique decision trees where each model is obtained using the TAO algorithm. We choose MNIST dataset and turn it into a binary imbalanced classification in a special way, the details of it we provide in Appendix E. For this imbalanced problem we first train the base oblique tree of depth $\Delta = 3$ which is the starting point in the COC path. From this point we then follow paths in both directions by changing the cost of FPs λ, and for each step we use the TAO algorithm to optimize the corresponding weighted 0/1 loss. We plot the performance of these COC curves in the right part of fig. 2 along with the traditional ROC curve for the base tree. Clearly, the COC significantly dominates the ROC, and is much closer to the top left corner, and reaches the TP-rate of 1 or FP-rate of 0 much early. The visualizations of some of these trees can be found in fig. 8 in Appendix.

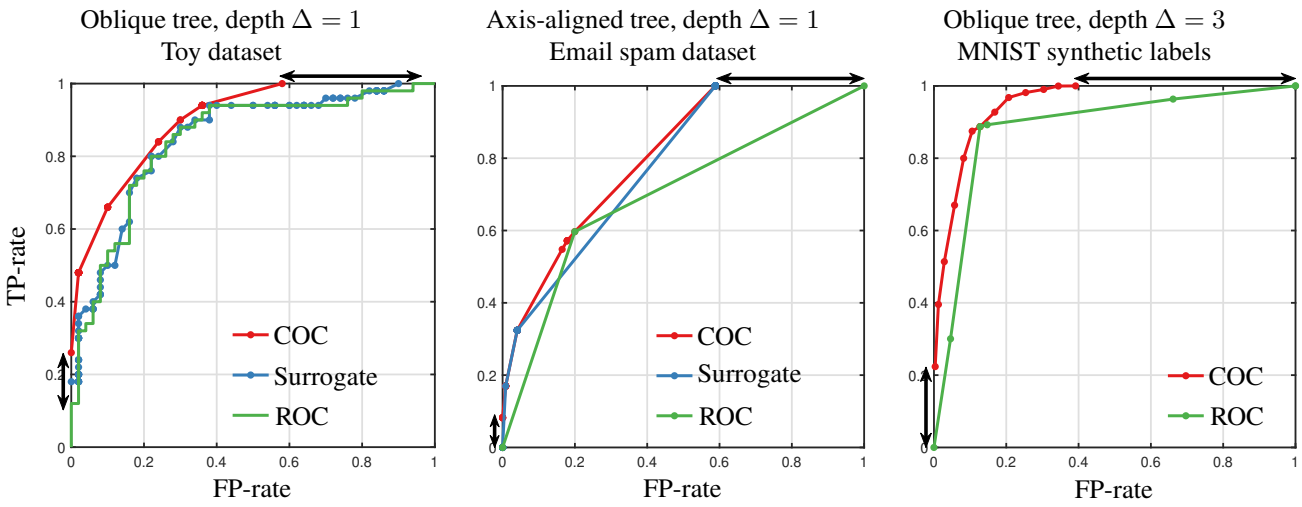


Figure 2. Illustration of the optimality of COC. Each point represents one model, and they are connected for better visualization.

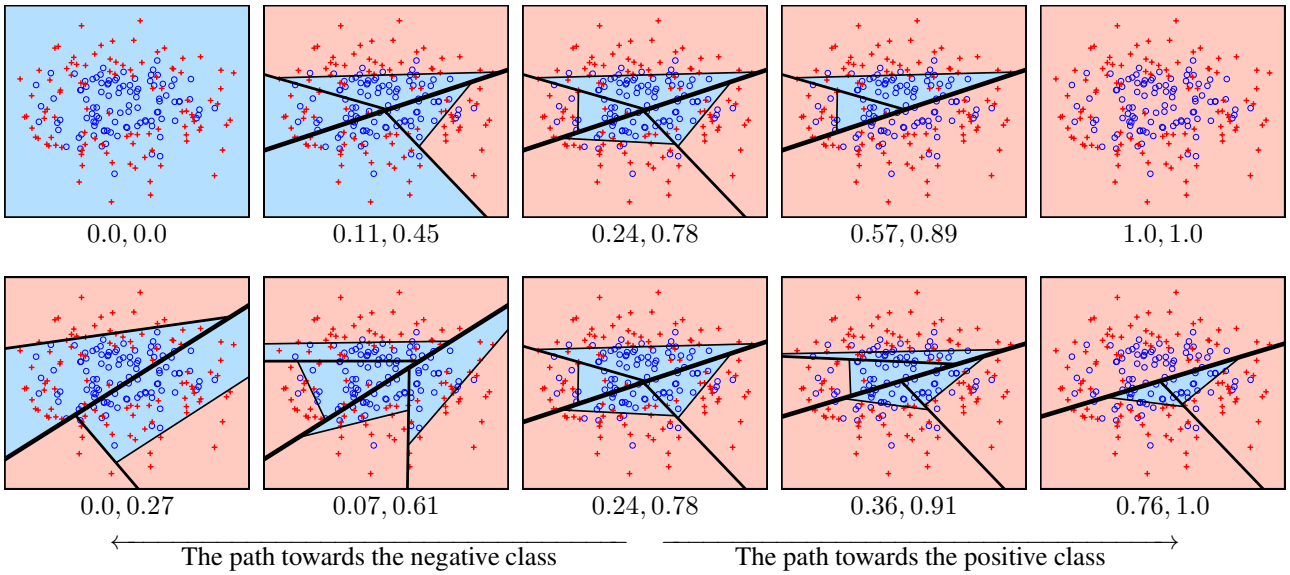


Figure 3. Top: trees corresponding to the ROC curve, bottom: trees corresponding to the COC curve. Below the plot we specify the FP-rate and TP-rate.

5.2. Toy 2D Illustration of COC Trees

To understand better how the decision trees behave in the COC framework of cost sensitive learning we perform experiments on a toy 2D problem and visualize the decision boundaries of the produced oblique trees. The toy dataset consists of $N = 200$ training points, and it is a balanced binary classification problem shown in fig. 3. We first train an oblique tree of depth $\Delta = 3$ with equal misclassification costs for both classes and obtain a tree produced in the middle of fig. 3. We then follow COC paths in both directions by either increasing or decreasing the cost of FPs, and each time warm starting from the previous tree. Some of these trees are visualized in the bottom row of fig. 3. The changing colors of decision regions clearly show how the more important class dominates and covers more area of the input space. The leftmost and rightmost trees correspond to the endpoints of the path which are the intersection points with the left and top axis of the ROC space. We can also see how the trees keep some stability and have some resemblance with each other owing to the warm-starting in COC.

The top row of fig. 3 shows how the tree changes in the traditional ROC curve. Because the only thing that changes is the threshold probability to be in the positive class, this can affect only the class label in the leaves. By noting that the parent node of two leaves with the same class label can be pruned without changing the predictive function of the

tree, one can observe how the tree is pruned and turned into a constant classifier as the threshold probability reaches 0 or 1 corresponding to the leftmost and rightmost plots. This clearly demonstrates the limited ability of an ROC curve with decision trees as the only thing it can do is to change the leaf label and prune the tree.

5.3. Imbalanced Classification Benchmarks

The results of our experiments on real-world imbalanced classification benchmarks validate the benefit of the proposed COC approach. Decision trees optimized by TAO for a cost-sensitive 0/1 loss produce much better TP and FP-rate performance than other established baselines such as cost supportive CART or C5.0 and other oblique decision trees such as OC1. While we motivate COC curves to be superior to ROCs both theoretically and experimentally, and also illustrate the limited ability of ROCs with trees, in the main experiments of this section *we use stronger comparison baselines: CART and C5.0 learned using different values of class costs each time to obtain models/points of different TP/FP-rate, and similarly for OC1, but with different sampling ratio* (OC1 implementation does not support costs). Comparing with ROC of a single tree would have been unfair, and we further justify the approach of using different costs vs a single ROC in fig. 9 of Appendix for CART. The advantage of using different costs is clearly visible, and that is what we compare against.

5.3.1. Setup

We perform experiments on multiple different real-world imbalanced datasets of various sizes and of different types of features such as credit card fraud detection, churn prediction and email spam detection. We provide the details regarding the dataset description and preprocessing in Appendix H.3. For the baselines, we compare against standard established decision tree methods: CART and C5.0. Both of these methods have support for costs per class, and so we use them to train models for different costs of FPs to obtain a range of models in the ROC space. There are not many implementations of oblique trees, and so we only include OC1 (Murthy et al., 1994), which is a variation of the greedy top-down induction technique for oblique splits. We also attempted to run one particular implementation of optimal trees which has support for class costs (Lin et al., 2020), but it did not finish within the runtime limit of 2 hours even for our smallest dataset of size $N_{\text{train}} = 4.9\text{k}$, $D = 45$. In Appendix H we provide all the details regarding the experiment setup such as hyperparameter tuning and implementation details.

	Tree	E_{test}	#leaves	Δ	time (s)
Covertype (500k,54,7)	OC1	28.45±2.71	709	22	1385
	CART	12.01±0.50	8327	45	10
	C5.0	9.68±0.26	9300	63	47
	TAO	7.84±0.11	4850	16	1020
RCV1 (15k,47k,52)	OC1	fails to produce results			
	C5.0	69.14±0.71	1531	257	8027
	CART	62.91±0.55	708	148	131
	TAO	57.15±0.31	867	13	648

Table 1. Results on imbalanced multiclass problems. E_{test} is the normalized (0-to-100) cost-based 0/1 error.

5.3.2. Results

We depict the performance of the resulting models in terms of TP-vs-FP-rate for the test set in fig. 4. Each point corresponds to one particular model, but we connect these points to create a "curve" for better visualization. Though these "curves" are commonly created with ROC, we would like to emphasize that this is not truly a curve but just a set of points (classifiers) connected with lines. COC refers to oblique decision trees trained with TAO for differing costs of FPs λ. Although these are not truly optimal models in the sense of the weighted 0-1 loss, we still refer to them as COC, because TAO directly and effectively optimizes this NP-hard problem with a guarantee of a monotonic decrease of an objective function.

In general we can observe how the COC dominates over the other methods throughout the whole range of TP and FP rates, and in several cases with a large margin, particularly for MNIST and Epsilon datasets. One important observation is that COC is much smoother and more stable than others showing the effectiveness of TAO in optimizing a given weighted 0-1 loss at each step of the path. Greedy top-down induction techniques provide quite unstable results since each time they need regrow the tree from scratch and cannot optimize/warm-start from the previous tree. In Appendix F we also provide additional comparison experiments with sampling-based techniques.

5.3.3. Training time

In general the training time of COC is slower than CART or C5.0, but still has manageable runtime. For the MNIST dataset training 14 models sequentially in COC takes about 150 seconds, while training the same number of CART models takes 80 seconds. OC1 is the slowest one taking more than 10 minutes in this setting. Warm starting ability of TAO was essential for the speedup in training of COC.

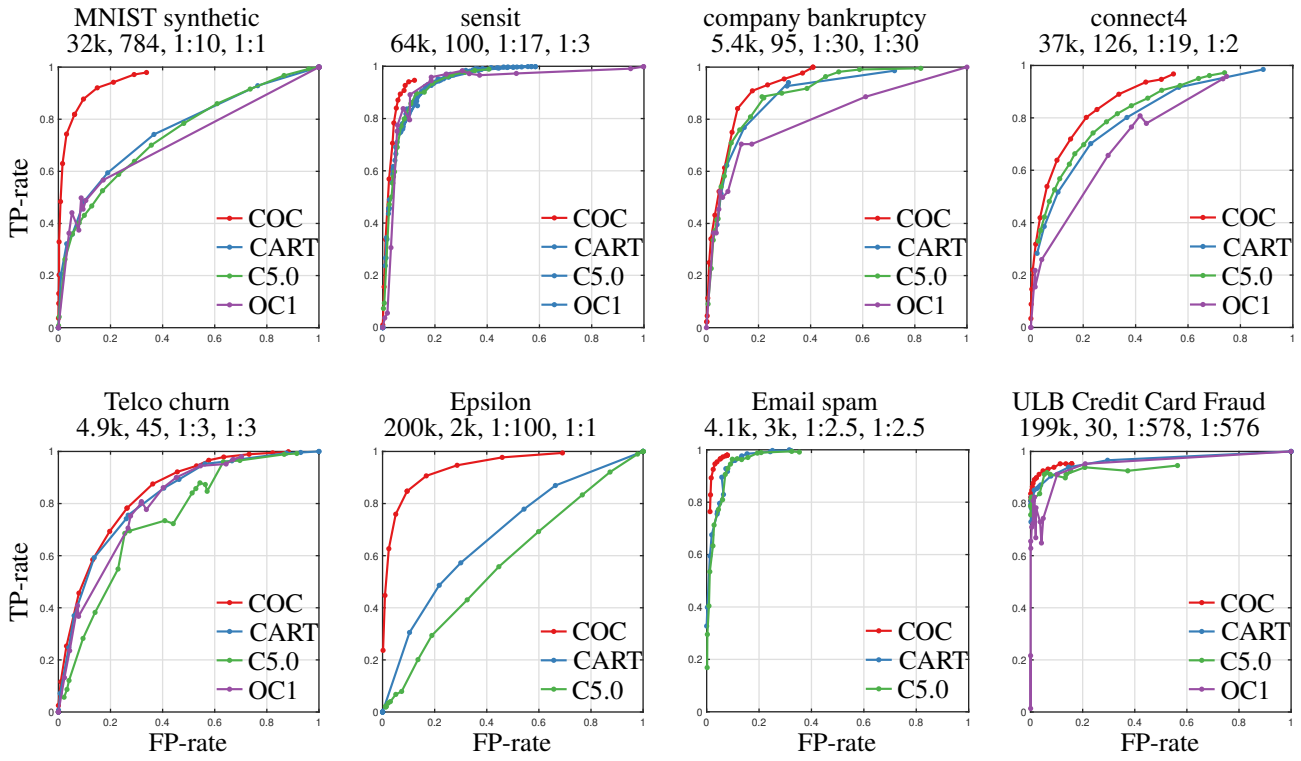


Figure 4. Comparison of different decision tree models in the test set. Under the dataset name we specify the training set size N_{train}, feature dimension D, class imbalance ratio for training and test set. Each point represents one model, and they are connected for better visualization. COC corresponds to oblique trees trained with TAO. CART and C5.0 correspond to training those algorithms with different class costs. OC1 is obtained by using either undersampling or oversampling. All results except for the Epsilon dataset are averaged over 5 runs. The result of OC1 is missing on Epsilon and Email spam datasets because it fails to produce any results on them.

5.3.4. Multiclass problems

Though we focus on the more prevalent imbalanced binary-class problems, our method can be extended to the multi-class case. By defining a weight for every type of error in the 0/1 loss, we can directly adapt the TAO algorithm to handle this objective. We perform experiments on two naturally imbalanced multiclass datasets, where we set the class costs to be such that the total cost of each class in the training set are equally distributed. Table 1 shows the results: TAO achieves lower cost-based 0/1 test error while using fewer leaves and being shallower.

6. Conclusion

Previous approaches to learn classification trees with asymmetric class costs or imbalanced classes involve over- or undersampling or cost-sensitive surrogate losses, postprocessed to construct an ROC curve so that one can achieve a desired false positive / true positive point. Here, we advocate for the ideal way to achieve this: the cost-optimal curve (COC), where each classifier is accuracy or true-positive optimal for a given false positive rate. This involves a weighted 0/1 loss, which is generally NP-hard to optimize exactly. Rather than replacing it with a surrogate loss, we have given an algorithm that tries to learn directly the COC curve for classification trees. For each false positive rate or cost, it monotonically decreases the weighted 0/1 loss at each iteration. While more computationally costly, experimentally this produces much better classification trees than previous approaches in reasonable time. We are working on extending this to tree ensembles.

Acknowledgments

Work partially supported by NSF award IIS–2007147.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning at a basic research level. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Branco, P., Torgo, L., and Ribeiro, R. P. A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys*, 49(2):31:1–31:50, August 2016.

Breiman, L. J., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Wadsworth, Belmont, Calif., 1984.

Carreira-Perpiñán, M. Á. and Tavallali, P. Alternating optimization of decision trees, with application to learning sparse oblique trees. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems (NEURIPS)*, volume 31, pp. 1211–1221. MIT Press, Cambridge, MA, 2018.

Carreira-Perpiñán, M. Á. and Zharmagambetov, A. Ensembles of bagged TAO trees consistently improve over random forests, AdaBoost and gradient boosting. In *Proc. of the 2020 ACM-IMS Foundations of Data Science Conference (FODS 2020)*, pp. 35–46, Seattle, WA, October 19–20 2020.

Carreira-Perpiñán, M. Á., Gabidolla, M., and Zharmagambetov, A. Towards better decision forests: Forest Alternating Optimization. In *Proc. of the 2023 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'23)*, pp. 7589–7598, Vancouver, Canada, June 18–22 2023.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. SMOTE: Synthetic minority over-sampling technique. *J. Artificial Intelligence Research*, 16:321–357, 2002.

Cook, E. F. and Goldman, L. Empiric comparison of multivariate analytic techniques: Advantages and disadvantages of recursive partitioning analysis. *J. Chronic Diseases*, 37(9–10):721–731, 1984.

Drummond, C. and Holte, R. C. Exploiting the cost (in)sensitivity of decision tree splitting criteria. In Langley, P. (ed.), *Proc. of the 17th Int. Conf. Machine Learning (ICML'00)*, pp. 239–246, Stanford, CA, June 29 – July 2 2000.

Duarte, M. F. and Hu, Y. H. Vehicle classification in distributed sensor networks. *J. Parallel and Distributed Computing*, 64(7):826–838, July 2004.

Elkan, C. The foundations of cost-sensitive learning. In *Proc. of the 17th Int. Joint Conf. Artificial Intelligence (IJCAI'01)*, pp. 973–978, Seattle, Washington, USA, August 4–10 2001.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. LIBLINEAR: A library for large linear classification. *J. Machine Learning Research*, 9:1871–1874, August 2008.

Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., and Herrera, F. *Learning from Imbalanced Data Sets*. Springer-Verlag, 2018.

Ferri, C., Flach, P. A., and Hernández-Orallo, J. Learning decision trees using the area under the ROC curve. In *Proc. of the 19th Int. Conf. Machine Learning (ICML'02)*, pp. 139–146, Sydney, Australia, June 8–12 2002.

Gabidolla, M. and Carreira-Perpiñán, M. Á. Pushing the envelope of gradient boosting forests via globally-optimized oblique trees. In *Proc. of the 2022 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'22)*, pp. 285–294, New Orleans, LA, June 19–24 2022a.

Gabidolla, M. and Carreira-Perpiñán, M. Á. Optimal interpretable clustering using oblique decision trees. In *Proc. of the 28th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (SIGKDD 2022)*, pp. 400–410, Washington, DC, August 14–18 2022b.

Gabidolla, M., Zharmagambetov, A., and Carreira-Perpiñán, M. Á. Improved multiclass AdaBoost using sparse oblique decision trees. In *Int. J. Conf. Neural Networks (IJCNN'22)*, Padua, Italy, July 18–22 2022.

Gajowniczek, K. and Ząbkowski, T. ImbTreeAUC: An R package for building classification trees using the area under the ROC curve (AUC) on imbalanced datasets. *SoftwareX*, 15(100755), July 2021.

Hastie, T. J., Tibshirani, R. J., and Friedman, J. H. *The Elements of Statistical Learning—Data Mining, Inference and Prediction*. Springer Series in Statistics. Springer-Verlag, second edition, 2009.

He, H. and Ma, Y. (eds.). *Imbalanced Learning: Foundations, Algorithms, and Applications*. John Wiley & Sons, 2013.

He, H., Bai, Y., Garcia, E. A., and Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Int. J. Conf. Neural Networks (IJCNN'08)*, pp. 1322–1328, Hong Kong, China, June 1–8 2008.

Holte, R. C. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11 (1):63–90, April 1993.

Höppner, S., Baesens, B., Verbeke, W., and Verdonck, T. Instance-dependent cost-sensitive learning for detecting

transfer fraud. *Eur. J. Operational Research*, 297(1): 291–300, February 16 2022.

Japkowicz, N. and Shah, M. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, November 1998.

Lewis, D. D., Yang, Y., Rose, T. G., and Li, F. RCV1: A new benchmark collection for text categorization research. *J. Machine Learning Research*, 5:361–397, April 2004.

Liang, D., Lu, C.-C., Tsai, C.-F., and Shih, G.-A. Financial ratios and corporate governance indicators in bankruptcy prediction: A comprehensive study. *Eur. J. Operational Research*, 252(2):561–572, July 16 2016.

Lin, J., Zhong, C., Hu, D., Rudin, C., and Seltzer, M. Generalized and scalable optimal sparse decision trees. In Daumé III, H. and Singh, A. (eds.), *Proc. of the 37th Int. Conf. Machine Learning (ICML 2020)*, pp. 6150–6160, Online, July 13–18 2020.

Murthy, S. K., Kasif, S., and Salzberg, S. A system for induction of oblique decision trees. *J. Artificial Intelligence Research*, 2:1–32, 1994.

Quinlan, J. R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

Raubertas, R. F., Rodewald, L. E., Humiston, S. G., and Szilagyi, P. G. ROC curves for classification trees. *Medical Decision Making*, 14(2):169–174, April 1994.

Sun, Y., Wong, A. K. C., and Kamel, M. S. Classification of imbalanced data: A review. *Int. J. Pattern Recognition and Artificial Intelligence*, 23(4):687–719, 2009.

Vanderschueren, T., Verdonck, T., Baesens, B., and Verbeke, W. Predict-then-optimize or predict-and-optimize? an empirical evaluation of cost-sensitive learning strategies. *Information Sciences*, 594:400–415, May 2022.

Zharmagambetov, A. and Carreira-Perpiñán, M. Á. Smaller, more accurate regression forests using tree alternating optimization. In Daumé III, H. and Singh, A. (eds.), *Proc. of the 37th Int. Conf. Machine Learning (ICML 2020)*, pp. 11398–11408, Online, July 13–18 2020.

Zharmagambetov, A. and Carreira-Perpiñán, M. Á. Learning interpretable, tree-based projection mappings for nonlinear embeddings. In *Proc. of the 25th Int. Conf. Artificial Intelligence and Statistics (AISTATS 2022)*, pp. 9550–9570, Online, March 28–30 2022a.

Zharmagambetov, A. and Carreira-Perpiñán, M. Á. Semi-supervised learning with decision trees: Graph Laplacian tree alternating optimization. In *Advances in Neural Information Processing Systems (NEURIPS)*, volume 35, pp. 2392–2405. MIT Press, Cambridge, MA, 2022b.

Zharmagambetov, A., Hada, S. S., Gabidolla, M., and Carreira-Perpiñán, M. Á. Non-greedy algorithms for decision tree optimization: An experimental comparison. In *Int. J. Conf. Neural Networks (IJCNN'21)*, Virtual event, July 18–22 2021.

A. Appendix Abstract

We provide the proof of the equivalence between the constraint and penalty forms of the COC problem in section B, the proofs of the cost sensitive TAO theorems in section C, the pseudocode of our method in section D, the description of the MNIST dataset with synthetic labels and the visualization of the trees for this problem in section E, the additional experimental comparison with sampling techniques in section F, experiments on additional datasets and exploration of hyperparameters in section G, and finally, the description of the experiments' setup for reproducibility: datasets, comparison methods, and hyperparameters in section H.

B. Equivalence between the Constraint and Penalty Forms of the Problem

We consider the problem in the main paper of constructing a cost-optimal curve (COC). Assume a training set $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$ where $\mathbf{x}_n \in \mathcal{X}$ and $y_n \in \{-1, +1\}$ for $n = 1, \dots, N$, and a classifier $T(\cdot; \boldsymbol{\theta})$: $\mathcal{X} \rightarrow \{-1, +1\}$ with parameters $\boldsymbol{\theta}$ taking values in some set. We gave two versions of the learning problem. One is the constraint version:

$$\min_{\boldsymbol{\theta}} \ \nu(\boldsymbol{\theta}) \quad \text{s.t.} \quad \pi(\boldsymbol{\theta}) \leq p \qquad \text{with } \nu(\boldsymbol{\theta}) = \sum_{n:\ y_n=+1}^{N} L(y_n, T(\mathbf{x}_n; \boldsymbol{\theta})),\ \pi(\boldsymbol{\theta}) = \sum_{n:\ y_n=-1}^{N} L(y_n, T(\mathbf{x}_n; \boldsymbol{\theta})) \tag{6}$$

where $p \in [0, N]$, $L(y, y') = 0$ if $y = y'$ and 1 if $y \neq y'$ is the 0/1 loss, and ν and π are the FN and FP rate, respectively. The other is the penalty version:

$$\min_{\boldsymbol{\theta}} \ \nu(\boldsymbol{\theta}) + \lambda\, \pi(\boldsymbol{\theta}) \tag{7}$$

where $\lambda \geq 0$. This objective function has the form of a weighted 0/1 loss. We will also define the following function:

$$C(\lambda) = \min_{\boldsymbol{\theta}} \ \nu(\boldsymbol{\theta}) + \lambda\, \pi(\boldsymbol{\theta}). \tag{8}$$

Problems (6) and (7) have the same set of solutions, i.e., solving (6) for all $p \in [0, N]$ produces the same set of classifiers as solving (7) for all $\lambda \geq 0$ and hence the same COC curve, as we will prove next. Hence, instead of (6), we solve (7), which is computationally easier with trees. Before reaching that result, we will prove several properties of the problems. Throughout, we assume that π and ν are lower bounded (w.l.o.g. $\nu(\boldsymbol{\theta}) \geq 0$ and $\pi(\boldsymbol{\theta}) \geq 0\ \forall \boldsymbol{\theta}$), and define "$\boldsymbol{\theta} \in \arg\min_{\boldsymbol{\theta}} \nu(\boldsymbol{\theta})$ s.t. $\pi(\boldsymbol{\theta}) \leq p$" or "$\boldsymbol{\theta} \in \arg\min_{\boldsymbol{\theta}} (\nu(\boldsymbol{\theta}) + \lambda\, \pi(\boldsymbol{\theta}))$" to mean any *global* minimizer of the constraint or penalty problem, respectively. The minimizer in $\boldsymbol{\theta}$ need not be unique, but the minimum objective function value is unique. Fig. 5 illustrates the constraint and penalty problems for a 1D dataset, where the solution can be found exactly.

Theorem B.1 (monotonicity of penalty solution)**.** *Assume π and ν are nonnegative functions of $\boldsymbol{\theta}$ in problems (6) and (7). Let $\lambda_2 > \lambda_1 > 0$ and $\boldsymbol{\theta}_i \in \arg\min_{\boldsymbol{\theta}} (\nu(\boldsymbol{\theta}) + \lambda_i\, \pi(\boldsymbol{\theta}))$ for $i \in \{1, 2\}$. Then $\pi(\boldsymbol{\theta}_2) \leq \pi(\boldsymbol{\theta}_1)$ and $\nu(\boldsymbol{\theta}_2) \geq \nu(\boldsymbol{\theta}_1)$, and $\nu(\boldsymbol{\theta}_1) + \lambda_1\, \pi(\boldsymbol{\theta}_1) \leq \nu(\boldsymbol{\theta}_2) + \lambda_2\, \pi(\boldsymbol{\theta}_2)$.*

Proof. Let us prove first that $\pi(\boldsymbol{\theta}_2) \leq \pi(\boldsymbol{\theta}_1)$ by contradiction. Assume $\pi(\boldsymbol{\theta}_2) > \pi(\boldsymbol{\theta}_1)$. Since $\boldsymbol{\theta}_1$ is a global minimizer for λ_1, we have that $\nu(\boldsymbol{\theta}_1) + \lambda_1\, \pi(\boldsymbol{\theta}_1) \leq \nu(\boldsymbol{\theta}) + \lambda_1\, \pi(\boldsymbol{\theta})\ \forall \boldsymbol{\theta}$, in particular for $\boldsymbol{\theta} = \boldsymbol{\theta}_2$. Hence:

$$\nu(\boldsymbol{\theta}_1) + \lambda_1\, \pi(\boldsymbol{\theta}_1) \leq \nu(\boldsymbol{\theta}_2) + \lambda_1\, \pi(\boldsymbol{\theta}_2) \iff \tag{9}$$

$$\nu(\boldsymbol{\theta}_1) - \nu(\boldsymbol{\theta}_2) \leq \lambda_1(\pi(\boldsymbol{\theta}_2) - \pi(\boldsymbol{\theta}_1)) \overset{(*)}{<} \tag{10}$$

$$\lambda_2(\pi(\boldsymbol{\theta}_2) - \pi(\boldsymbol{\theta}_1)) \iff \tag{11}$$

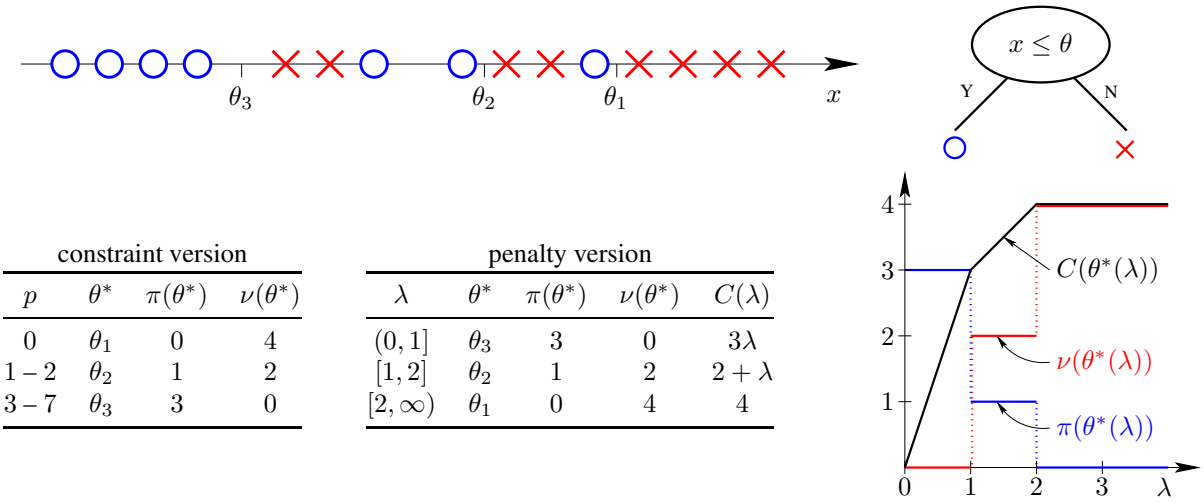
$$\nu(\boldsymbol{\theta}_1) + \lambda_2\, \pi(\boldsymbol{\theta}_1) < \nu(\boldsymbol{\theta}_2) + \lambda_2\, \pi(\boldsymbol{\theta}_2) \tag{12}$$

where in $(*)$ we used the fact that $\lambda_1 < \lambda_2$ and $\pi(\boldsymbol{\theta}_2) > \pi(\boldsymbol{\theta}_1)$. Likewise, since $\boldsymbol{\theta}_2$ is a global minimizer for λ_2, we have that $\nu(\boldsymbol{\theta}_2) + \lambda_2\, \pi(\boldsymbol{\theta}_2) \leq \nu(\boldsymbol{\theta}) + \lambda_2\, \pi(\boldsymbol{\theta})\ \forall \boldsymbol{\theta}$, in particular for $\boldsymbol{\theta} = \boldsymbol{\theta}_1$. Hence:

$$\nu(\boldsymbol{\theta}_2) + \lambda_2\, \pi(\boldsymbol{\theta}_2) \leq \nu(\boldsymbol{\theta}_1) + \lambda_2\, \pi(\boldsymbol{\theta}_1)$$

which contradicts eq. (12).

Now, to prove that $\nu(\boldsymbol{\theta}_2) \geq \nu(\boldsymbol{\theta}_1)$. This follows directly from eq. (10) and the fact we just proved that $\pi(\boldsymbol{\theta}_2) \leq \pi(\boldsymbol{\theta}_1)$.



constraint version

p	θ^*	$\pi(\theta^*)$	$\nu(\theta^*)$
0	θ_1	0	4
$1-2$	θ_2	1	2
$3-7$	θ_3	3	0

penalty version

λ	θ^*	$\pi(\theta^*)$	$\nu(\theta^*)$	$C(\lambda)$
$(0,1]$	θ_3	3	0	3λ
$[1,2]$	θ_2	1	2	$2+\lambda$
$[2,\infty)$	θ_1	0	4	4

Figure 5. Illustration of the constraint and penalty forms of the COC problem for a 1D training set having two classes: positive (○) and negative (×). The classifier is a depth-1 decision tree (1-rule) which classifies x as positive if $x \le \theta$ and negative otherwise. The two tables show, for the constraint and penalty problems, the solution as a function of p or λ, respectively. The graph corresponds to the penalty problem. Note the optimum parameter values $\theta^*(p)$ and $\theta^*(\lambda)$ need not be unique (e.g. moving θ_1 left or right slightly in the figure is still optimum), but the objective function value at θ^* is unique. It can be seen that the constraint and penalty problems have the same set of solutions over p or λ, with $\pi(\theta^*(\lambda))$ (or $\nu(\theta^*(\lambda))$) being monotonically decreasing (increasing) and piecewise constant over λ, and $C(\lambda)$ being continuous, piecewise linear and strictly monotonically increasing except in the last interval, where it is constant.

Finally, from eq. (9) it follows that, either

$$\nu(\boldsymbol{\theta}_1) + \lambda_1\,\pi(\boldsymbol{\theta}_1) < \nu(\boldsymbol{\theta}_2) + \lambda_2\,\pi(\boldsymbol{\theta}_2)$$

if $\pi(\boldsymbol{\theta}_2) > 0$ (strictly decreasing), or

$$\nu(\boldsymbol{\theta}_1) + \lambda_1\,\pi(\boldsymbol{\theta}_1) = \nu(\boldsymbol{\theta}_2) + \lambda_2\,\pi(\boldsymbol{\theta}_2)$$

if $\pi(\boldsymbol{\theta}_2) = 0$ (constant). The latter can only happen in the last piece, because π is nonnegative and, as a function of λ, $\pi(\boldsymbol{\theta}^*(\lambda))$ is decreasing. □

Theorem B.2 (penalty objective is continuous piecewise linear)**.** *Assume π and ν are functions of $\boldsymbol{\theta}$ and take values in $\{0,1,\dots,N\}$ (this holds if they are 0/1 loss functions in problems (6) and (7)). Then $C(\lambda)$ for $\lambda > 0$ is continuous and piecewise linear, where all the pieces are strictly increasing except possibly the last piece, which may be constant.*

Proof. From theorem B.1, ν and π are monotonic over $\boldsymbol{\theta}^*(\lambda)$. Since they take values in $\{0,1,\dots,N\}$, then both ν and π must be piecewise constant with at most $N+1$ pieces. Hence, there exist M real values $0 < \lambda_1 < \lambda_2 < \cdots < \lambda_M$ such that ν and π are constant in each interval $(\lambda_i, \lambda_{i+1})$. Hence $C(\lambda)$ is linear in each interval. Further, let us show that C must be continuous. Within each interval this follows because C is linear there. At the boundary λ_i between two intervals $(\lambda_{i-1}, \lambda_i)$ and $(\lambda_i, \lambda_{i+1})$ it follows because C is increasing (from theorem B.1). Indeed, write $C(\lambda)$ as $\nu_i + \lambda\,\pi_i$ in $(\lambda_{i-1}, \lambda_i)$ and $\nu_{i+1} + \lambda\,\pi_{i+1}$ in $(\lambda_i, \lambda_{i+1})$. If C was discontinuous at λ_i then we would have $\nu_i + \lambda_i\,\pi_i \neq \nu_{i+1} + \lambda_i\,\pi_{i+1}$ and since C is increasing this becomes $\nu_i + \lambda_i\,\pi_i < \nu_{i+1} + \lambda_i\,\pi_{i+1}$. But this would imply that (ν_i, π_i) has a lower value than (ν_{i+1}, π_{i+1}) on the right interval, contradicting the fact that (ν_{i+1}, π_{i+1}) were global minimizers. □

Theorem B.3 (equivalence of the constraint and penalty problems)**.** *Assume π and ν are nonnegative functions of $\boldsymbol{\theta}$ in problems (6) and (7). Then the two sets of solutions, of problem (6) for $p \ge 0$ and of problem (7) for $\lambda \ge 0$, are equal.*

Proof. First we prove that any solution of problem (7) (for a value $\lambda \ge 0$) is a solution of problem (6) for some value of p. Let $\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} (\nu(\boldsymbol{\theta}) + \lambda\,\pi(\boldsymbol{\theta}))$. This means $\nu(\boldsymbol{\theta}^*) + \lambda\,\pi(\boldsymbol{\theta}^*) \le \nu(\boldsymbol{\theta}) + \lambda\,\pi(\boldsymbol{\theta})\ \forall \boldsymbol{\theta}$. Call $p^* = \pi(\boldsymbol{\theta}^*)$. Then we will show that $\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \nu(\boldsymbol{\theta})$ s.t. $\pi(\boldsymbol{\theta}) \le p$ for $p = p^*$, by contradiction. Assume $\exists \boldsymbol{\theta} \neq \boldsymbol{\theta}^*$ with $\pi(\boldsymbol{\theta}) \le p^*$ and $\nu(\boldsymbol{\theta}) < \nu(\boldsymbol{\theta}^*)$ (i.e., a better solution of problem (6) for p^*). Then $\nu(\boldsymbol{\theta}) + \lambda\,\pi(\boldsymbol{\theta}) < \nu(\boldsymbol{\theta}^*) + \lambda\,\mathbf{p}^* = \nu(\boldsymbol{\theta}^*) + \lambda\,\pi(\boldsymbol{\theta}^*)$, which is a contradiction. Hence, $\forall \lambda \ge 0$, $\boldsymbol{\theta}^*(\lambda)$ gives a solution $\boldsymbol{\theta}^*(p^*)$ of problem (6).

Now we prove that any solution of problem (6) (for a value $p \geq 0$) is a solution of problem (7) for some value of λ. Let $\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} \nu(\boldsymbol{\theta})$ s.t. $\pi(\boldsymbol{\theta}) \leq p$. Assume $\pi(\boldsymbol{\theta}^*) = p$. (If $\pi(\boldsymbol{\theta}^*) = p^* < p$ then $\boldsymbol{\theta}^*$ is optimum for any $p \geq p^*$ and the constraint is inactive (unnecessary) $\forall p \geq p^*$.) Assume $\exists \lambda \geq 0$ such that $\boldsymbol{\theta}' \in \arg\min_{\boldsymbol{\theta}} (\nu(\boldsymbol{\theta}) + \lambda \pi(\boldsymbol{\theta}))$ satisfies $\pi(\boldsymbol{\theta}') = p^*$ (this follows because, from theorems B.1 and B.2, C is invertible in its domain). Then, it follows that $\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}} (\nu(\boldsymbol{\theta}) + \lambda \pi(\boldsymbol{\theta}))$ (although it is not necessary that $\boldsymbol{\theta}' = \boldsymbol{\theta}^*$). Since $\boldsymbol{\theta}'$ is a minimizer we have that $\nu(\boldsymbol{\theta}') + \lambda \pi(\boldsymbol{\theta}') = \nu(\boldsymbol{\theta}') + \lambda \pi(\boldsymbol{\theta}^*)$ is a minimizer. This can be proven by contradiction. Assume $\nu(\boldsymbol{\theta}') < \nu(\boldsymbol{\theta}^*)$. Then $\boldsymbol{\theta}'$ is feasible in problem (6) (since $\pi(\boldsymbol{\theta}') = \pi(\boldsymbol{\theta}^*) = p$) and it has a lower objective function value than $\boldsymbol{\theta}^*$, which contradicts that $\boldsymbol{\theta}^*$ was a minimizer. □

Theorem B.1 shows that the optimum values of ν and π as a function of λ are monotonically increasing and decreasing, respectively, and the optimum objective $C(\lambda)$ is also increasing with λ (strictly increasing except in the last, infinite interval). Theorem B.2 shows that $C(\lambda)$ is continuous and piecewise linear, hence invertible in its domain. Theorem B.3 shows that problems (6) and (7) are equivalent: the set of respective solutions over p and λ are one-to-one. That is, the solution of the penalty form for a value of λ equals that of the constraint form for some value p; specifically, $p = \pi(\boldsymbol{\theta}^*(\lambda))$. And, conversely, the solution of the constraint form for a value p equals that of the penalty form for some value λ, specifically, a value λ such that $\pi(\boldsymbol{\theta}^*(\lambda)) = p$ (this value is not explicit but must be found by searching over $\lambda \geq 0$).

Finally, we give a different, simple proof for theorem B.3 but assuming the functions are smooth. This does not apply to the 0/1 loss we focus on, but it would apply to the cross-entropy, for example.

Theorem B.4 (equivalence of the constraint and penalty problems)**.** *Assume $\nu(\boldsymbol{\theta})$ and $\pi(\boldsymbol{\theta})$ are continuously differentiable and nonnegative in problems (6) and (7). Then both problems have the same stationary points.*

Proof. Let us apply the KKT conditions to both problems. For problem (7), we have that $\nabla \nu(\boldsymbol{\theta}) + \lambda \nabla \pi(\boldsymbol{\theta}) = \mathbf{0}$ (and $\lambda \geq 0$). For problem (7), the Lagrangian is $\mathcal{L}(\boldsymbol{\theta}, \lambda) = \nu(\boldsymbol{\theta}) - \lambda(p - \pi(\boldsymbol{\theta}))$, where λ is the Lagrange multiplier, and we have that $\nabla \nu(\boldsymbol{\theta}) + \lambda \nabla \pi(\boldsymbol{\theta}) = \mathbf{0}$ and that either $\pi(\boldsymbol{\theta}) < p$ and $\lambda = 0$, or $\pi(\boldsymbol{\theta}) = p$ and $\lambda \geq 0$. The second case (where the constraint becomes an equality constraint) shows the stationary points of both problems are one-to-one. The first case (where the constraint is inactive) means we obtain the same solution by setting $p = \pi(\boldsymbol{\theta})$ as an equality constraint. □

C. Theorem Statements and Proofs for the Cost-Sensitive TAO Algorithm

Formally let $T(\cdot; \boldsymbol{\Theta})$: $\mathcal{X} \rightarrow \{-1, +1\}$ be a decision tree of a given, predetermined structure of depth Δ with learnable parameters $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{i \in \mathcal{D}} \cup \{\theta_i \in \{-1, 1\}\}_{i \in \mathcal{L}}$, where $\mathcal{D}$ are the set of decision nodes and $\mathcal{L}$ are the set of leaf nodes. The predictive function of a tree $T(\mathbf{x}; \boldsymbol{\theta})$ works by routing $\mathbf{x}$ from the root to exactly one leaf and outputting the leaf's class label. The routing function at a decision node i works by applying the decision function $g_i(\mathbf{x}; \boldsymbol{\theta}_i)$: $\mathcal{X} \rightarrow \{\texttt{left}_i, \texttt{right}_i\} \subset \mathcal{D} \cup \mathcal{L}$, which for oblique splits $\boldsymbol{\theta}_i = \{\mathbf{w}_i, w_{i0}\}$ mean "go to the left child if $\mathbf{w}_i^T \mathbf{x} + w_{i0} < 0$, else go to the right child". Axis-aligned trees are a special case of oblique trees, for which $\mathbf{w}_i$ is all zeros but one at the selected feature index and $-w_{i0}$ serves as a threshold value. The objective function for TAO in the context of cost-sensitive learning is to minimize a weighted 0/1 loss:

$$E(\boldsymbol{\Theta}) = \sum_{n:\ y_n = +1}^{N} L(y_n, T(\mathbf{x}_n; \boldsymbol{\Theta})) + \lambda \sum_{n:\ y_n = -1}^{N} L(y_n, T(\mathbf{x}_n; \boldsymbol{\Theta})) + \alpha \sum_{i \in \mathcal{D}} \phi_i(\boldsymbol{\theta}_i) \tag{13}$$

We add a regularization term $\phi(\cdot)$ to decision node weights, specifically ℓ_1 norm to promote sparse oblique splits. The hyperparameter $\alpha \geq 0$ then controls the strength of the regularization.

We now show how we extend the TAO algorithm to handle this cost-sensitive 0/1 loss problem. The algorithm is based on 3 theorems. The first theorem of the original TAO states the **separability condition**: the objective function (13) separates over any set of non-descendant nodes (e.g. all nodes at the same depth), and thus those can be optimized independently and in parallel. This follows from the fact the tree makes hard decisions and that the loss is separable over individual points. Our cost-sensitive problem is no different in this respect, and we directly use this theorem without any modification.

For presenting the next 2 theorems the following definitions are helpful: *Reduced set* $\mathcal{R}_i \subset \{1, \dots, N\}$ of node i (decision node or leaf) is the training instances that reach node i given the current tree parameters. *Care points* $\mathcal{C}_i \subset \mathcal{R}_i$ of a decision node i are the training instances in the reduced set $\mathbf{x}_n \in \mathcal{R}_i$ for which $T_{\texttt{left}_i}(\mathbf{x}_n; \boldsymbol{\Theta}_{\texttt{left}_i}) \neq T_{\texttt{right}_i}(\mathbf{x}_n; \boldsymbol{\Theta}_{\texttt{right}_i})$ where $T_c(\cdot; \boldsymbol{\Theta}_c)$ is the predictive function for the subtree rooted at node c given the current tree parameters. In other words, care

points for a given decision node consists of training instances in its reduced set for which one of its children misclassifies the instance while the other child classifies it correctly.

Theorem C.1 (Reduced problem over a decision node)**.** *Consider the objective function $E(\boldsymbol{\Theta})$ of eq. (13) and a decision node i. Assume the parameter values $\boldsymbol{\Theta} \setminus \{\boldsymbol{\theta}_i\}$ of all the nodes except i are fixed. Then, as a function of $\boldsymbol{\theta}_i$, eq. (13) can be reduced to the following cost-sensitive binary classification problem:*

$$E_i(\boldsymbol{\theta}_i) = \sum_{n \in \mathcal{C}_i:\ y_n=+1} L(\overline{y}_{in}, g_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \lambda \sum_{n \in \mathcal{C}_i:\ y_n=-1} L(\overline{y}_{in}, g_i(\mathbf{x}_n; \boldsymbol{\theta}_i)) + \alpha\, \phi_i(\boldsymbol{\theta}_i) \tag{14}$$

where $\overline{y}_{in} = \arg\min_{c \in \{\texttt{left}_i, \texttt{right}_i\}} L(y_n, T_c(\mathbf{x}_n; \boldsymbol{\Theta}_c))$ is the child of i which gives the correct prediction for the instance n.

Proof. Using the assumption that the parameter values of $\boldsymbol{\Theta} \setminus \{\boldsymbol{\theta}_i\}$ of all the nodes except i are fixed it is easy to see that the optimization problem in eq. (13) will only depend on the parameters of the node i. This follows directly from the separability condition discussed above. Furthermore, it will only depend on the training instances in the reduced set $\mathcal{R}_i$ of the node i as changes to this decision function $g_i(\cdot; \boldsymbol{\theta}_i)$ will have no effect on the other training instances. Then, let us analyze what can happen to a given training point $n \in \mathcal{R}_i$ by changes to this decision node i. The only thing a decision function $g_i(\mathbf{x}_n; \boldsymbol{\theta}_i)$ can do with a given point n is to send it either to the left child or to the right child. If both of its children gives the correct prediction or both of them misclassifies this point n, then it would not matter where to send it, and so this point n can be dropped from the objective function, thus allowing us to define the reduced problem of eq. (14) only over the set of care points $\mathcal{C}_i$. Then for the given care point $n \in \mathcal{C}_i$ one of the children gives the correct prediction while the other child misclassifies it, and we denote the child with the correct prediction as $\overline{y}_{in}$. Now, most importantly in the context of cost-sensitive learning, if this point $y_n = +1$ belongs to the positive class then the cost of sending it to the wrong child is 1, while if this point $y_n = -1$ is a negative class then the cost of sending it to the wrong child is λ. Applying this observation to each trainning point $n \in \mathcal{C}_i$ and noting the regularization term $\phi_i(\boldsymbol{\theta}_i)$, one can see how the high-level optimization problem of eq. (13) will reduce to this weighted 0/1 loss problem over the parameters of this node $\boldsymbol{\theta}_i$ of eq. 14. □

Theorem C.2 (Reduced problem over a leaf)**.** *Consider the objective function $E(\boldsymbol{\Theta})$ of eq. (13) and a leaf node i. Assume the parameter values $\boldsymbol{\Theta} \setminus \{\theta_i\}$ of all the nodes except i are fixed. Then, as a function of θ_i, eq. (13) can be reduced to the following simple problem:*

$$E_i(\theta_i) = \sum_{n \in \mathcal{R}_i:\ y_n=+1} L(y_n, \theta_i) + \lambda \sum_{n \in \mathcal{R}_i:\ y_n=-1} L(y_n, \theta_i) \tag{15}$$

where $\mathcal{R}_i$ is the reduced set of the leaf i. The optimal solution $\theta_i^* = +1$ if $|\mathcal{R}_i^+| \geq \lambda\, |\mathcal{R}_i^-|$ else -1, where $|\mathcal{R}_i^+|$ ($|\mathcal{R}_i^-|$) is the number of points in the positive (negative) class in the reduced set $\mathcal{R}_i$ of the leaf i.

Proof. By direct application of the separability condition the objective function of eq. (13) will only depend on the label θ_i of the leaf node i and only on the leaf's reduced set $\mathcal{R}_i$. There are only two choices for the label $\theta_i \in \{-1, +1\}$. By simple calculation, the optimal label θ_i^* then corresponds to the weighted majority class in the reduced set $\mathcal{R}_i$. □

C.1. Multiclass problems

Although we focus on cost-sensitive *binary* classification problems, we should note how the TAO algorithm can be straightforwardly extended to the *multiclass* case. Instead of a single cost of FPs λ, the objective function in eq. (13) would contain K different types of costs $\{\lambda_k\}_{k=1}^K$ for each type of error (the more general case would involve a $K \times K$ matrix of different types of costs, but to handle imbalanced multiclass problems we simply define a single misclassification cost λ_k for a class k so that to balance the total cost across all the classes. The TAO algorithm can also be straightforwardly adapted to this more general case). Then the reduced problem over a decision node would still be a weighted 0/1 loss binary classification problem, but the weights coming from K different types of costs $\{\lambda_k\}_{k=1}^K$. The solution to the reduced problem over a leaf node would still be a weighed majority class.

D. Pseudocode

We provide a pseudocode of TAO for training oblique decision trees for a 0/1 loss with costs in fig. 6 and a pseudocode for obtaining a set of models in the COC framework in fig. 7.

input initial tree $T(\cdot, \boldsymbol{\Theta})$ of depth Δ,
training set $\{\mathbf{x}_n, \mathbf{y}_n \in \{-1, 1\}\}_{n=1}^N$, cost of false positives λ,
regularization parameter $\alpha \geq 0$
repeat
  for $d = \Delta$ **to** 0 **do**
    for all nodes $node$ at depth d
      if $node$ is a leaf
        set the label of the $node$ to the most
        costly class in the reduced set
      else
        fit a weighted 0/1 binary classifier
        where weights come from the costs
      end if
    end for
  end for
until stopping criterion
return tree $T(\cdot; \boldsymbol{\Theta})$

Figure 6. Pseudocode of TAO (reverse BFS order) for cost-sensitive learning.

input: training set $\{\mathbf{x}_n, \mathbf{y}_n \in \{-1, 1\}\}_{n=1}^N$,
depth of the tree Δ, regularization parameter $\alpha \geq 0$,
schedule parameter $\beta > 1$.
Set the base cost $\lambda_0 = \frac{N^+}{N^-}$
$T_0(\cdot; \boldsymbol{\Theta}) =$ TAO on complete random tree of depth Δ with cost $\lambda = \lambda_0$
$T_0^-(\cdot; \boldsymbol{\Theta}) = T_0(\cdot; \boldsymbol{\Theta})$, $\lambda = \lambda_0$, $i = 0$
repeat
  $\lambda \leftarrow \beta\, \lambda$
  $T_{i+1}^-(\cdot; \boldsymbol{\Theta}) =$ TAO on $T_i^-(\cdot; \boldsymbol{\Theta})$ as initial tree with cost λ
  $i \leftarrow i + 1$
until false positives by $T_i^-(\cdot; \boldsymbol{\Theta})$ is zero
$T_0^+(\cdot; \boldsymbol{\Theta}) = T_0(\cdot; \boldsymbol{\Theta})$, $\lambda = \lambda_0$, $i = 0$
repeat
  $\lambda \leftarrow \lambda / \beta$
  $T_{i+1}^+(\cdot; \boldsymbol{\Theta}) =$ TAO on $T_i^+(\cdot; \boldsymbol{\Theta})$ as initial tree with cost λ
  $i \leftarrow i + 1$
until false negatives by $T_i^+(\cdot; \boldsymbol{\Theta})$ is zero
return all trained trees $\{T_i^-(\cdot; \boldsymbol{\Theta})\}_i \cup \{T_i^+(\cdot; \boldsymbol{\Theta})\}_i$

Figure 7. Pseudocode of COC with decision trees.

E. MNIST with Synthetic Labels

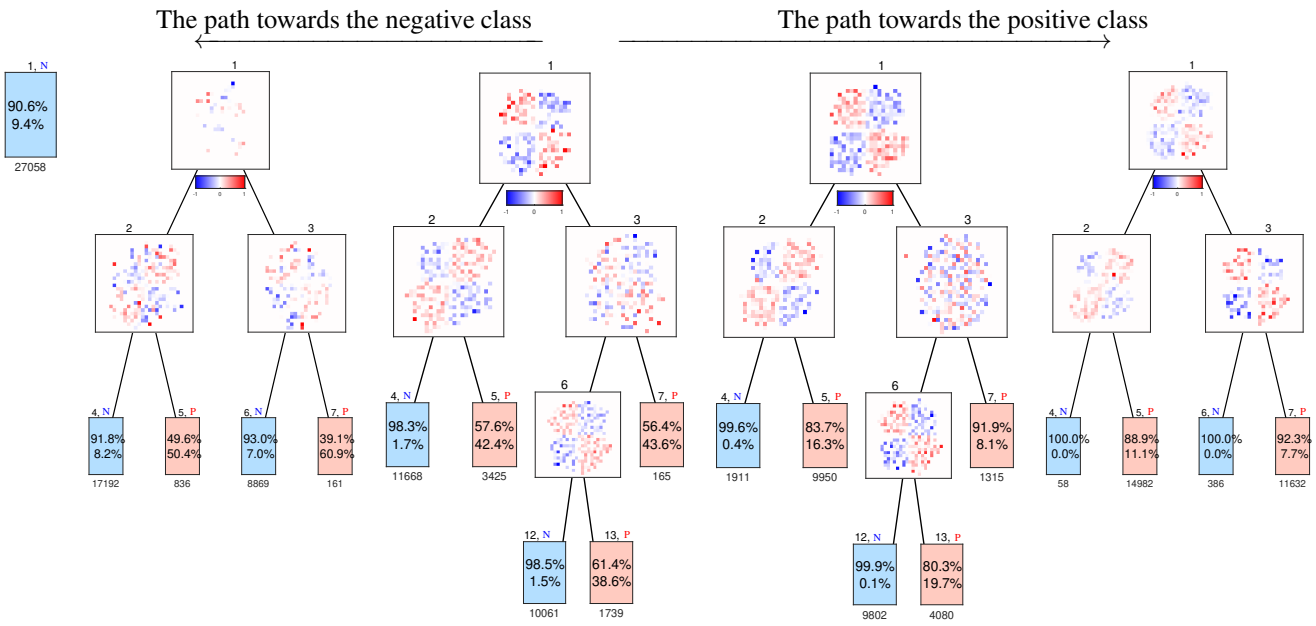


Figure 8. Visualization of trees for MNIST with synthetic labels. The color of the leaves show the class label: red is for positive class, blue is for negative class. The percentages in the leaves correspond to the proportion of negative (top number) and positive (bottom number) classes.

We turn MNIST into an imbalanced binary classification as follows: imagine splitting the 28×28 pixel image into 4 quadrants $\begin{smallmatrix}1&2\\3&4\end{smallmatrix}$ and let Q_i be the sum of the $[0,1]$ pixel intensities in quadrant i and $H = 18$. Then we define the positive class to be an image for which $(Q_1 + Q_4) - (Q_2 + Q_3) \geq H$ or $(Q_2 + Q_3) - (Q_1 + Q_4) \geq H$. That is, positive class is an "(anti)diagonally dominant" image like ▚ or ▞. This construction creates balanced classes for both training and test set, and then we randomly undersample the positive class and add 10% label noise in the training set to make the problem imbalanced and have some overlapping classes. This synthetic labels serve two purposes: it creates a more difficult binary classification problem in MNIST than simple methods such as grouping of classes or selecting pairs of digits; it allows to illustrate the superiority of oblique trees over axis-aligned ones for problems for which it is important to model the interaction of many features or pixels.

In fig. 8 we visualize some subset of trees in the COC path for this imbalanced MNIST problem. We plot the oblique hyperplane as a 28×28 image, where blue (negative) pixels contribute sending points to the left, and red (positive) pixels contribute sending points to the right. The middle tree corresponds to the starting model of the COC path. Notice how all the decision nodes except 3 is capturing this diagonal pattern, and as the path goes toward the positive class, all the decision nodes in the rightmost tree is capturing this pattern. And in the opposite path toward the negative class we can observe that the decision nodes start to lose this diagonal shape until the tree collapses to a single node.

F. Comparison with Sampling Techniques

Sampling based techniques such as under- or over-sampling or SMOTE are commonly used in imbalanced classification. Here we compare their performance in the ROC space. In a similar way we obtain COC curves by changing the λ, we can also under- or over-sample the necessary class to maintain the class ratio equal to λ, and produce a set of models in the ROC space. In fig. 10 we compare the performance of these sampling techniques against the cost-based method. The difference between those is not significant for the TAO algorithm, which makes sense because sampling techniques can be regarded as a reasonable approximation to costs. However we favor more the cost based approach because it is more principled and theoretically justified and performs well in practice.

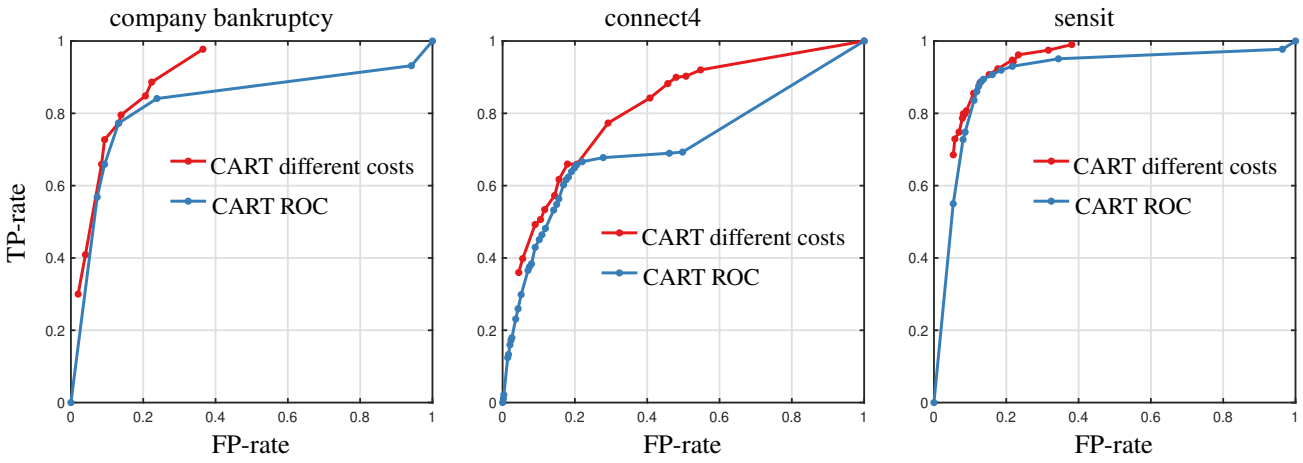


Figure 9. Comparison of using traditional ROC curve (based on changing the threshold) versus using different costs each time during the tree learning process for CART. The approach based on using different costs is better, and this is what we compare against in the experiments of fig. 4 in the main paper.

G. Experiments on Additional Datasets and Exploration of Hyperparameters

Fig. 11 contains experiments on additional datasets: SUSY with 4.5M training points and the CIFAR10 dataset with features extracted from ResNet18. Fig. 12 shows the effect of the hyperparameters in our model: Depth Δ of the tree and the ℓ_1 penalty regularization α on decision node weights.

H. Experimental setup

H.1. COC Curves and Tree Optimization

H.1.1. Implementation

We implement the TAO algorithm for oblique decision trees in C++. The algorithm requires to solve a weighed 0/1 binary classification problem over a hyperplane at a decision node, and for that we use ℓ_1 regularized logistic regression solver of LIBLINEAR (version 2.43 with support for instance weights). As an initial tree we use a complete tree of depth Δ and random (sampled from standard Normal distribution) initial parameters at decision nodes. We let the TAO algorithm to run for 20 iterations. To produce a range of models in the COC framework we first train the base TAO model for the cost of false positives $\lambda = N^+_{\text{train}}/N^-_{\text{train}}$ which corresponds to giving equal total cost for both classes. From this base model we then follow two regularization paths by using the following schedules on the cost λ: $\lambda \leftarrow \beta\,\lambda$ and $\lambda \leftarrow \lambda/\beta$. Although β can be tuned as any other hyperparameter, which would control how close or far we move in the COC curve, to keep experiments simple we fix it to $\beta = 1.5$. These two COC curve paths stop when FP becomes 0 in one direction and FN becomes 0 in the other direction in the training set.

H.1.2. Hyperparameters

The hyperparameters are the depth Δ of the tree and the regularization parameter in decision node weights α. We perform grid search $\alpha = \{0.1, 1.0, 10.0\}$ and $\Delta = \{1, 3, 4, 6\}$ for the base tree only, and use the found best parameters for the whole COC curve. We include depth $\Delta = 1$ tree in case a simple linear model happens to generalize better.

H.2. Baselines

CART We use scikit-learn's implementation of CART (version 1.2.2). We fully grow the tree using Gini index, but we implement our own cost-complexity pruning. This is because scikit-learn's implementation of pruning uses Gini index instead of 0/1 error, which is not what the original authors of CART use (Breiman et al., 1984). We pass the costs in the `sample_weight` parameter of the `fit()` method and also use the costs in our implementation of cost-complexity pruning.

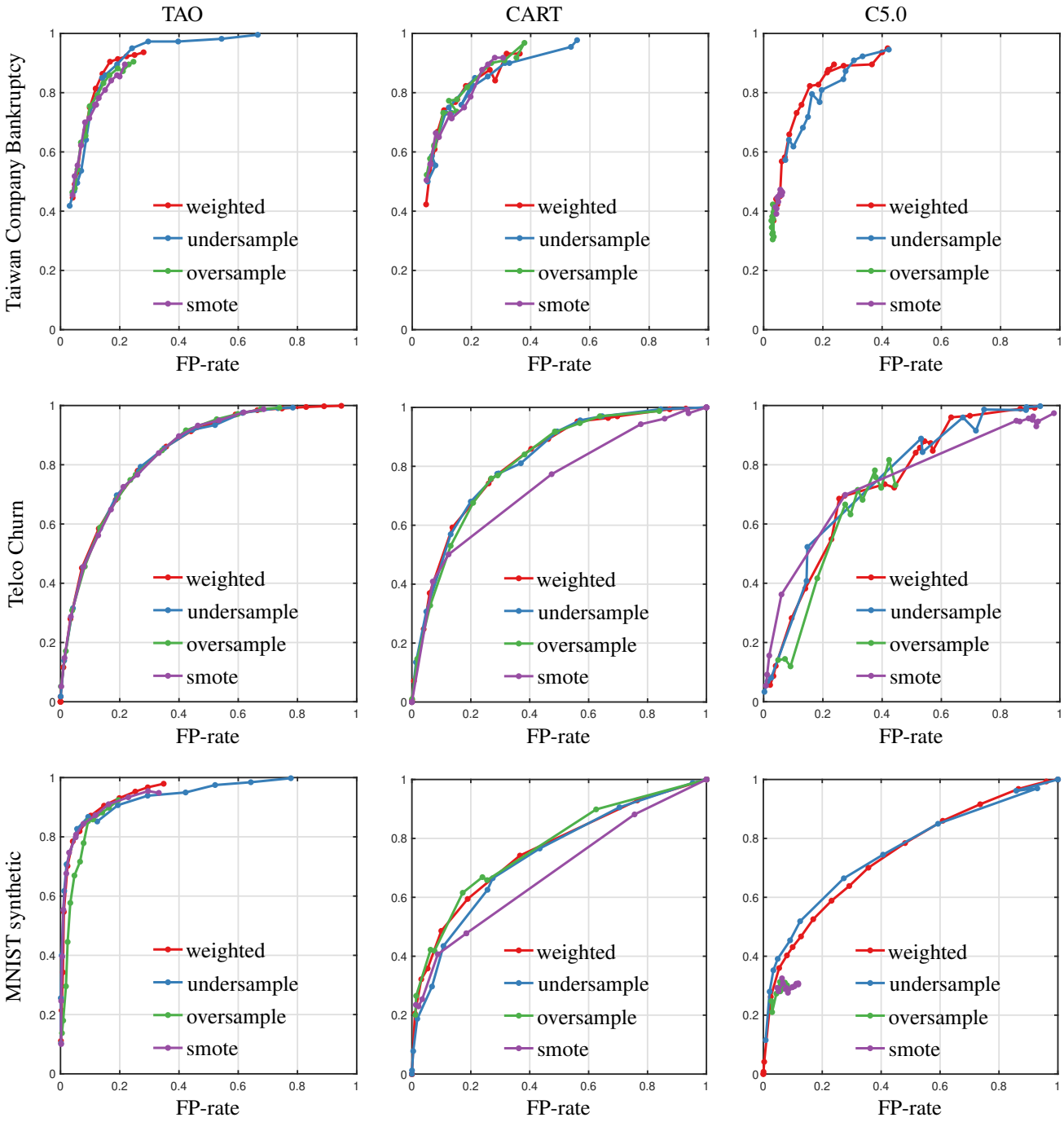


Figure 10. Comparison of weighted learning vs common sampling techniques for the 3 datasets. Results are averaged over 5 runs.

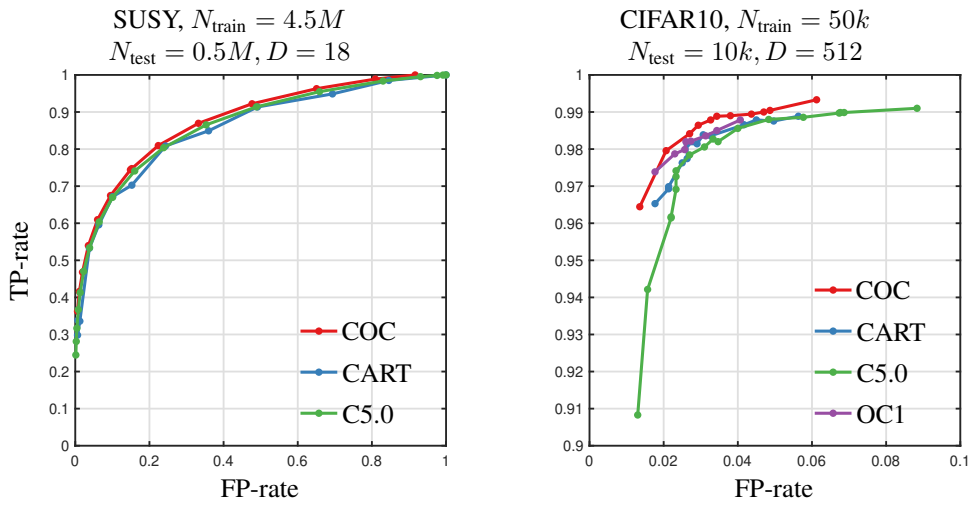


Figure 11. Experiments on additional datasets. For the SUSY dataset, the training times for CART and C5.0 curves take 1320 and 850 seconds, respectively. For the COC approach, the training time is 2300 seconds. OC1 code fails to produce any results for the SUSY dataset. For CIFAR10 dataset we use convolutional features from a pretrained ResNet18 model. We turn CIFAR10 into binary classification problem by using the classes "plane", "ship" and "truck" as the positive class, and the rest as negative.

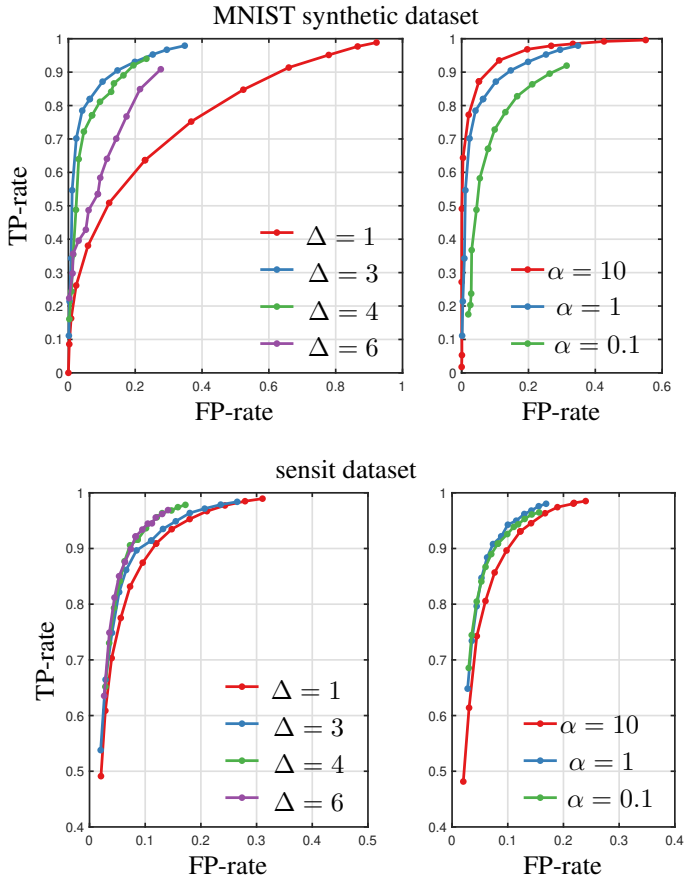


Figure 12. Effect of hyperparameters in COC for the two datasets: MNIST synthetic (top) and sensit (bottom). The two hyperparameters are the depth Δ of the tree and the ℓ_1 penalty regularization α on decision node weights.

Dataset	N_{train}	N_{test}	D	$N^+{:}N^-_{\text{train}}$	$N^+{:}N^-_{\text{test}}$
MNIST synthetic labels	32 469	10 000	784	1:10	1:1
SensIT	64 209	19 705	100	1:17	1:3
Taiwan Company Bankruptcy	5 455	1 364	95	1:30	1:30
Connect4	37 424	13 512	126	1:19	1:2
Telco Churn	4 922	2 110	45	1:3	1:3
Epsilon	202 174	100 000	2 000	1:100	1:1
ULB Credit Card Fraud	199 364	85 443	30	1:578	1:576
Email Spam	4 137	1 035	3 000	1:2.5	1:2.5
Covertype*	406 708	174 304	54	1:77	1:77
RCV1*	20 564	30 000	47 236	1:1800	1:1800

Table 2. Specifics of the datasets used in our experiments. N is a sample size, D is a feature dimension size. * for multiclass datasets we specify the class imbalance as ratio of the majority class to the minority class.

C5.0 We use the Linux binaries provided by the original developers[3]. Because the training procedure of C5.0 already performs on pruning and selects the best model on a validation set internally, we did not perform hyperparameter tuning. We attempted to tune the option `-c` which controls the severity of pruning but it did not result in better models than the default case. The implementation of C5.0 has support for class costs, and we provide it via the `data.costs` file.

OC1 We use the author's implementation in C. We use the option of only considering the oblique hyperplane. We tune the number of restarts from the list $\{10, 20, 50\}$ and the number of random jumps from $\{5, 10, 20\}$. In several cases the method fails to produce any result due to not being able to find the split. The implementation does not support costs, and so use either under- or over-sampling, and report the better one.

GOSDT When attempting to run this method, we use the author's original implementation[4]. However with default parameters it did not finish in 2 hours even for our smallest dataset Telco Churn ($N_{\text{train}} = 5k, D = 45$), and so we did not run it with any other dataset.

SMOTE For experiments with sampling techniques we use the implementation of SMOTE provided in `imbalanced-learn` Python package version 0.10.1.

H.3. Datasets

For datasets that do not have a separate test set we randomly sample (stratified) 20% of the data and use it as a test set. For categorical features we apply one-hot encoding. For datasets with features of different scale and magnitude, we scale them individually to have mean zero and variance one.

MNIST (LeCun et al., 1998) with synthetic labels We use raw pixel intensities scaled between 0 and 1 as input. The details on how we create the imbalanced binary labels are described in section E.

SensIT (Duarte & Hu, 2004) Vehicle classification task based on the features collected by wireless sensor networks. The dataset has 3 classes and we combine the two most populous classes into a majority (negative) class, and then we further undersample the minority class to create an imbalanced problem. Obtained from the LIBSVM dataset repository: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`

Taiwan company bankruptcy (Liang et al., 2016) The features are various stock market characteristics. The dataset is naturally imbalanced. Obtained from the UCI machine learning repository: `https://archive.ics.uci.edu/ml/datasets/Taiwanese+Bankruptcy+Prediction`

Connect4 The features are positions of the game, and the label is the outcome: win, draw or lose. We use the class "win" as the majority class, and merge the other two classes into one and undersample to create an imbalanced problem. We

[3] `https://www.rulequest.com/download.html`
[4] `https://github.com/Jimmy-Lin/GeneralizedOptimalSparseDecisionTrees`

download the dataset from the UCI machine learning repository:
https://archive.ics.uci.edu/ml/datasets/connect-4

Telco Churn The task is to predict which customer will churn. The features are various characteristics of the customer. The dataset is relatively imbalanced. Obtained from Kaggle:
https://www.kaggle.com/datasets/blastchar/telco-customer-churn.

Epsilon is one of the datasets used in the Large Scale Learning Challenge 2008. Obtained from the LIBSVM dataset repository. We undersample the positive to create a highly imbalanced problem.

ULB Credit Card Fraud One of the most popular datasets in Kaggle for fraud detection. The original features were PCA transformed to keep anonymity except for Amount and Time features. The dataset is highly imbalanced. The link in Kaggle:
https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Email Spam The task to classify emails as spam or not-spam. The original features are word counts of the most common 3000 words. We preprocess these raw counts by applying tf-idf normalization. The dataset is relatively imbalanced. Obtained from Kaggle:
https://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv

Covertype Classification task of pixels into 7 forest cover types based on attributes such as elevation, aspect, slope, hillshade, soil-type, and more. The dataset is naturally imbalanced with the following class counts: 211840, 283301, 35754, 2747, 9493, 17367, 20510. We use 30% of the dataset as test set, and perform zero-mean, variance-one normalization. Obtained from the UCI machine learning repository:
https://archive.ics.uci.edu/dataset/31/covertype

RCV1 Text categorization dataset (Lewis et al., 2004). We obtain it from the LIBSVM multiclass data collection: http://csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html. We use the same train/test split described there. The features are normalized log TF-IDF vectors; they are sparse. The dataset has many classes (=52) and is highly imbalanced with several classes containing less than 10 points while some containing 1000 points.