Convex Polytope Trees and its Application to VAE

Anonymous Author(s)
Affiliation
Address
email

Abstract

A decision tree is commonly restricted to use a single hyperplane to split the covariate space at each of its internal nodes. It often requires a large number of nodes to achieve high accuracy, hurting its interpretability. In this paper, we propose convex polytope trees (CPT) to expand the family of decision trees by an interpretable generalization of their decision boundary. The splitting function at each node of CPT is based on the logical disjunction of a community of differently weighted probabilistic linear decision-makers, which also geometrically corresponds to a convex polytope in the covariate space. We use a nonparametric Bayesian prior at each node to infer the community's size, encouraging simpler decision boundaries by shrinking the number of polytope facets. We develop a greedy method to efficiently construct CPT and a scalable end-to-end differentiable training algorithm for the tree parameters. Our proposed differentiable training method allows us to expand the scope of the decision trees to other differentiable downstream tasks. For example, we employ CPT in combination with VAEs for the generative modeling problem. Empirically, we demonstrate the advantages of CPT over existing state-of-the-art decision trees in several real-world classification and regression tasks. Furthermore, we show its effectiveness when employed for generative modeling.

1 Introduction

Decision trees [9] are highly interpretable models, which make them favorable in high-stakes domains such as medicine [37, 47] and criminal justice [6]. They are also resistant, if not completely immune, to the inclusion of many irrelevant predictor variables. However, trees usually do not have high accuracy, which somewhat limits their use in practice. Current main approaches to improve the performance of decision trees are making large trees or using ensemble methods [13, 14, 20, 55], such as bagging [7] and boosting [15, 18], which come with the price of harming model interpretability. There is a trade-off challenge between the accuracy and interpretability of a decision tree.

Prior work has attempted to address the aforementioned challenge and improve the performance of trees by introducing oblique tree models [21, 39]. These families of models are generalizations of classical trees, where the decision boundaries are hyperplanes that are not constrained to be axis-parallel and can have an arbitrary orientation. This change in the decision boundaries has been shown to reduce the size of trees. However, the tree size often remains too large in real datasets to make it amenable to interpretation. There has been an extensive body of research to improve the training of the oblique trees and enhance their performance [5, 11, 33, 50], yet their large size remains a challenge.

This paper proposes convex polytope decision trees (CPT) to expand the class of oblique trees by extending hyperplane cuts to more flexible geometric shapes. To be more specific, the decision boundaries induced by each internal node of CPT are based on noisy-OR [41] of multiple linear classifiers. Since noisy-OR has been widely accepted as an interpretable Bayesian model [42], our generalization keeps the interpretability of oblique trees intact. Furthermore, CPT's decision

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

boundaries geometrically resemble a convex polytope ($i.e.$, high dimensional convex polygon). Therefore, the decisions at each node have both logical and geometrical interpretations. We use the gamma process [16, 31, 54], a nonparametric Bayesian prior, to infer the number of polytope facets adaptively at each internal tree node and regularize the capacity of the proposed CPT. A realization of the gamma process consists of countably infinite atoms, each of which is used to represent a weighted hyperplane of a convex polytope. The gamma process's shrinkage property helps us encourage having simpler decision boundaries, therefore helps resist overfitting and improve interpretability.

Learning convex polytope boundaries is challenging due to the large number of possible ways that even single splits have. We present a scalable differentiable training procedure that mitigates the challenge of learning discrete decisions at each node. Our method applies to both classification and regression tasks; via mutual information maximization and variance minimization.

As a by-product, our differentiable training allows one to use modern stochastic gradient descent (SGD) based programming frameworks. To further illustrate this advantage, we combine CPT with VAE [29]. This experiment demonstrates the effectiveness of CPT in improving the performance of VAE models; it also emphasizes the importance of having fewer leaves, which is another unique benefit of CPT compared to oblique trees.

Experimentally, we compare the performance of CPT to state-of-the-art decision tree algorithms [11, 33] on a variety of representative regression and classification tasks. We experiment with several real-world datasets from diverse domains, such as computer vision, tabular data, and chemical property data. Experiments demonstrate that CPT outperforms state-of-the-art methods with higher accuracy and smaller size.

Our main contributions are: 1) We propose an interpretable generalization to the family of oblique decision trees models; 2) We regularize the expressive power of CPT, using a nonparametric Bayesian shrinkage prior for each node's split function; 3) We provide scalable and differentiable ways of learning CPT models (for classification, regression, and generative modeling) that efficiently search for the optimal tree; 4) We experimentally evaluate CPT on several different datasets, illustrating that this new approach outperforms the prior work 5) We extend the scope of decision trees to generative modeling, and show CPT's effectiveness both in terms of better inference performance and computational complexity compared to oblique trees.

2 Related work

Most decision tree literature focuses on training a single or an ensemble of trees [20] instead of making decision boundaries more flexible. One reason for this lack of research is the increase in the problem's computational complexity, even for simple generalizations of the decision boundary. For example, with N as the number of data points and d as the dimension of the input space, generalizing the coordinate-wise to an oblique hyperplane cut, increases the number of possible splits of data points from Nd to $\sum_{i=0}^{d} \binom{N}{i}$ just for a single node [48].

To have more flexible decision boundaries, some methods perform hyperplane cuts in an extended feature space, created by concatenating the original and newly generated features [1]. These new features are engineered or kernel-based and not designed for interpretability. Therefore unlike CPT, they are not an interpretable extension of the decision tree. We follow this section with a literature review of the recent training algorithms for axis-aligned and oblique trees.

Conventional methods for decision tree induction are greedy, where they grow the tree nodes one at a time. The greedy construction of oblique trees can be done by using coordinate descent to learn the parameters of each split [39], or by a projection of the feature space to a lower dimension then using coordinate-cut [38, 50]. However, the greedy procedure often leads to sub-optimal trees.

There have been several attempts to non-greedy optimization, which rely on either fuzzy or probabilistic split functions [24, 32, 44]. The probabilistic trees are sometimes referred to as soft decision trees [19, 22, 32]. In these methods, the assignment of a single sample to the leaf is fuzzy or probabilistic, and gradient descent is used to optimize the tree. Most of these algorithms remain probabilistic at test time, leading to uninterpretable models as the prediction for each sample will be based on multiple leaves of the tree instead of just one.

Other advances towards the training of an oblique tree are based on constructing neural networks that reproduce decision trees [33, 52]. Yang et al. [52] use a neural network with argmax activations for the representation of classic decision tree with coordinate cuts, but they are not scalable to high-dimensional data. Lee and Jaakkola [33] use the gradient of a ReLU network with a single hidden unit at each layer and skip-connections to construct an oblique decision tree. They achieve state-of-the-art results on some datasets but have to make complete trees with a high depth, consequently having a large number of leaves.

In contrast to our method, there are other training algorithms that require the tree's structure at the beginning. Some of the works in this direction like Bennett [4] and Bertsimas and Dunn [5] use linear programming, or mixed-integer linear programming, to find an optimum global tree. Therefore, these methods are computationally expensive and not scalable. Norouzi et al. [40] derive a convex-concave upper bound on the tree's empirical loss and optimize that loss using SGD. A recent work [11] proposes tree alternating optimization, where one directly optimizes the misclassification error over separable subsets of nodes, achieving the state-of-the-art empirical performance on several datasets in a comprehensive comparison [53].

We conclude this section by connecting our splitting rule at each internal node to relevant classification algorithms [2, 25, 35, 36, 49, 54]. The two most related works are convex polytope machine (CPM) [25] and infinite support hyperplane machine (iSHM) [54], which both exploit the idea of learning a convex polytope associated decision boundary. iSHM is like a single hidden layer NN and is restricted to the binary classification task.

3 Convex Polytope Tree and Its Inference Algorithms

Suppose we are given the training data $(\mathbf{X}, \mathbf{Y}) = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, where pairs of $(\boldsymbol{x}_n, y_n)$ are drawn independently from an identical and unknown distribution D. Each $\boldsymbol{x}_n \in \mathbb{R}^d$ is a d-dimensional data with a corresponding label $y_n \in \mathcal{Y}$. In the classification setting, $\mathcal{Y} = \{1, \cdots, K\}$ and in regression scenario $\mathcal{Y} = \mathbb{R}$. The aim is to learn a function $F : \mathbb{R}^d \to \mathcal{Y}$ that will perform well in predicting the label on samples from D. Decision tree methods construct the function F by recursively partitioning the feature space to yield a number of hierarchical, disjoint regions and assign a single label (value) to each region. In what follows, we will explain how we move beyond the oblique decision trees.

3.1 Convex Polytope Constrained Decision Boundary

By extending the idea of disjunctive interaction (noisy-OR, probabilistic-OR) [43, 23, 54] from probabilistic reasoning to the decision tree problem, we make the decision boundaries more flexible while preserving interpretability. To that end, we transform the problem of splitting at each node (right or left) to a committee of experts that make individual binary decisions ("Yes" or "No"). The committee votes "Yes" if and only if at least one expert votes "Yes", otherwise it votes "No". Thus, the final vote at each node is

$$vote = \bigvee_{i=1}^{K} vote_i,$$

where $\bigvee$ denotes the logical OR operator. We model each expert as a linear classifier who votes "Yes" with probability

$$P(vote_i = \text{"Yes"} \,|\, \{r_i, \boldsymbol{\beta}_i\}, \boldsymbol{x}) = 1 - (1 + e^{\boldsymbol{\beta}_i' \boldsymbol{x}})^{-r_i}, \tag{1}$$

where $r_i \geq 0$ and $\boldsymbol{\beta}_i \in \mathbb{R}^d$ are parameters of expert i. Now assuming that each expert votes independently, we can express the probability of the committee voting "Yes" as

$$P(vote = \text{"Yes"} \,|\, \{r_i, \boldsymbol{\beta}_i\}_i, \boldsymbol{x}) = 1 - \prod_{i=1}^{K}(1 - p_i) = 1 - e^{-\sum_{i=1}^{K} r_i \ln(1 + e^{\boldsymbol{\beta}_i' \boldsymbol{x}})}, \tag{2}$$

where p_i is the probability of expert i voting "Yes" and K is the total number of experts. We can now define the split function at each node by thresholding the committee voting probability:

$$A_{left} := \{\boldsymbol{x} \,|\, \boldsymbol{x} \in \mathbb{R}^d, P(vote = \text{"Yes"} \,|\, \{r_i, \boldsymbol{\beta}_i\}_i, \boldsymbol{x}) \leq q_{thr}\} \tag{3}$$

where A_{left} and $A_{right} := \mathbb{R}^d \setminus A_{left}$ are the related splits of the space.

To elaborate on the geometric shape and interpretability of the decision boundaries, consider $K = 1$ ($i.e.$ a single expert). In this scenario, the decision boundary becomes a hyperplane, which is

perpendicular to $\boldsymbol{\beta}$. In fact, the probability function for each expert is based on the signed distance of $\boldsymbol{x}$ to the hyperplane perpendicular to $\boldsymbol{\beta}$. Moreover, the parameter r controls how smoothly the probability transitions from 0 to 1, where a larger r leads to sharper changes. This class of models with $K = 1$ and $r = 1$ are identical to oblique trees, which are interpretable. Furthermore, when $K \geq 1$, the interpretability is provided by the fact that linear classifiers and probabilistic-OR operation are both interpretable [42].

To geometrically analyze the implied decision regions, we provide the following theorem.

Theorem 1. *For any $\{r_i, \boldsymbol{\beta}_i\}_{i=1}^K$, such that $r_i \in \mathbb{R}_+$ and $\beta_i \in \mathbb{R}^d$, let:*

$$A_{left} = \{\boldsymbol{x} \,|\, \boldsymbol{x} \in \mathbb{R}^d, \;\; f(\boldsymbol{x}) \leq q_{thr}\}, \quad \textit{where:} f(\boldsymbol{x}) = 1 - e^{-\sum_{i=1}^K r_i \ln(1+e^{\boldsymbol{\beta}_i' \boldsymbol{x}})} \tag{4}$$

where: then A_{left} is a convex set, confined by a convex polytope.

The proof is provided in the Appendix.

The above theorem shows that for $K \geq 1$, the decision region (A_{left}) is a convex set confined by a convex K-sided polytope. More precisely, each facet of the convex polytope is a hyperplane corresponding to an expert perpendicular to its $\boldsymbol{\beta}$. Also worth noting, an expert with a larger r has more effect on the decision boundary, making sharper changes to the probability function. This can also be perceived as the value of their decision in the committee. Therefore, our method not only has a strong relationship with probabilistic-OR type models that provide interpretability for the model parameters [3] but also has decision boundaries with interpretable geometric characteristics. We propose a class of models, Convex Polytope Trees (CPT), where each tree node follows the above splitting function.

Gamma Process Prior To regularize CPT, and motivate simpler decision boundaries, we use a nonparametric Bayesian shrinkage prior. Specifically, we put the gamma process prior [16, 31, 54] on the splitting function of each node in the tree. For more details, please refer to Appendix.

3.2 Training Algorithm

Finding an optimal CPT requires solving a combinatorial, non-differentiable optimization problem. This is due to the large number of possibilities that any single node can separate the data. We propose a continuous relaxation of the splitting rule at each node to alleviate this computationally challenging task. Each internal node makes probabilistic rather than deterministic decisions to send samples to its right or left branch. We set the probability of going right equal to (2), or any monotonic function of it. We use this probabilistic version to train the tree in a differentiable manner. Furthermore, at the test time, we threshold the splitting functions to provide a deterministic tree. In the following subsections, we explain in detail the proposed training algorithm for the splitting parameters (Section 3.2.1) and structure of the tree (Section 3.2.2).

3.2.1 Learning Split Parameters

Assuming the tree structure is given, we explain how to infer the tree parameters for each task.

Classification. For this task, we formulate the training as an optimization problem by considering the mutual information between the two random variables $\mathbf{Y}$ (category label) and $\mathbf{L}$ (leaf id) as our objective function. This differs from the previous work on learning a classical decision tree in two main ways: 1) we develop and optimize the mutual information for a probabilistic rather than a deterministic tree, and 2) we learn the parameters of all nodes jointly rather than learning them in a greedy fashion.

We model our probabilistic tree by letting

$$\ell \sim P_{\boldsymbol{\theta}}(\ell \,|\, \boldsymbol{x}_n) \qquad \text{such that} \qquad \ell \in S_{leaf}, \tag{5}$$

where S_{leaf} is the set of all leaf nodes and $P_{\boldsymbol{\theta}}(\ell \,|\, \boldsymbol{x}_n)$ is the probability of arriving at leaf ℓ given the sample feature $\boldsymbol{x}_n$. We assume each internal node makes decisions independent of the others and use the probabilities in (2) when sending a data sample to the left or right branch. This assumption lets us derive a formula for $P_{\boldsymbol{\theta}}$ as

$$P_{\boldsymbol{\theta}}(\ell \,|\, \boldsymbol{x}_n) = \prod\nolimits_{(v,d_v) \in \boldsymbol{\nu}_\ell} q_{\boldsymbol{\theta}_v}(d_v \,|\, \boldsymbol{x}_n), \tag{6}$$

where

$$q_{\boldsymbol{\theta}_v}(d_v = 1 \,|\, \boldsymbol{x}_n) = 1 - q_{\boldsymbol{\theta}_v}(d_v = 0 \,|\, \boldsymbol{x}_n)$$
$$= 1 - e^{-\sum_{i=1}^{K} r_i^{(v)} \ln\left(1+e^{\boldsymbol{\beta}_i^{(v)\prime} \cdot \boldsymbol{x}}\right)} \tag{7}$$

and $\boldsymbol{\nu}_\ell$ is a path from the root to leaf ℓ and $d_v \in \{0, 1\}$ encodes the right or left (0 or 1) direction taken at node v. The mutual information between $\mathbf{Y}$ and $\mathbf{L}$ can be expressed as

$$\mathcal{I}(\mathbf{Y}, \mathbf{L}) = \mathcal{H}(\mathbf{Y}) - \mathcal{H}(\mathbf{Y} \,|\, \mathbf{L}) = \mathcal{H}(\mathbf{Y}) - \sum\nolimits_{\ell \in S_{\text{leaf}}} p(\mathbf{L} = \ell)\mathcal{H}(\mathbf{Y} \,|\, \mathbf{L} = \ell),$$

where $\mathcal{H}(\cdot)$ indicates the entropy of a random variable, and $\mathbf{Y} \,|\, \mathbf{L}$ follows a categorical distribution. Notice that, since $\mathcal{H}(\mathbf{Y})$ does not depend on the tree parameters, to optimize mutual information, we only need to minimize $\mathcal{H}(\mathbf{Y} \,|\, \mathbf{L})$. However, the evaluation of the conditional entropy term requires knowledge of the entire data distribution, thus we can not directly optimize (8).

To make the training possible, we approximate the true data distribution with the empirical one to get

$$\hat{p}(\mathbf{L} = \ell) = \sum\nolimits_{n=1}^{N} P_{\boldsymbol{\theta}}(\ell \,|\, \boldsymbol{x}_n)/N. \tag{8}$$

Denote $\mathbf{1}_{[\cdot]}$ as an indicator function. By using Bayes' rule, we derive $\hat{\boldsymbol{\pi}}^\ell = (\hat{\pi}_1^\ell, \cdots, \hat{\pi}_C^\ell)$, the estimated probability vector of the categorical distribution for $\mathbf{Y} \,|\, \mathbf{L} = \ell$, as

$$\hat{\pi}_j^\ell = \frac{\sum_n \mathbf{1}_{[y_n = j]} P_{\boldsymbol{\theta}}(\ell \,|\, \boldsymbol{x}_n)}{\sum_n P_{\boldsymbol{\theta}}(\ell \,|\, \boldsymbol{x}_n)}, \;\; j = 1, \ldots, C \;\; . \tag{9}$$

Now by using (9), we can approximate the entropy term $\mathcal{H}(\mathbf{Y} \,|\, \mathbf{L} = \ell)$ as

$$\hat{\mathcal{H}}(\mathbf{Y} \,|\, \mathbf{L} = \ell) = -\sum\nolimits_{j=1}^{C} \hat{\pi}_j^\ell \log(\hat{\pi}_j^\ell). \tag{10}$$

Therefore, we can provide an estimator for the $\mathcal{H}(\mathbf{Y} \,|\, \mathbf{L})$ as

$$\hat{\mathcal{H}}(\mathbf{Y} \,|\, \mathbf{L}) = \sum\nolimits_{\ell \in S_{\text{leaf}}} \hat{p}(\mathbf{L} = \ell)\hat{\mathcal{H}}(\mathbf{Y} \,|\, \mathbf{L} = \ell). \tag{11}$$

By minimizing $\hat{\mathcal{H}}(\mathbf{Y} \,|\, \mathbf{L})$ with respect to $\boldsymbol{\theta}$, we are in fact maximize the mutual information $\hat{\mathcal{I}}(\mathbf{Y}, \mathbf{L})$. As discussed in Section A.1, we also regularize CPT by adding a penalty term to (11). We consider the negative log probability of the gamma process prior truncated with K atoms by letting

$$r_1, \ldots, r_K \overset{iid}{\sim} \text{Gamma}(\gamma_0/K, 1/c_0)$$

where $r_1, \ldots, r_K$ are the parameters of internal nodes splitting function. The penalty term for each internal node can be mathematically formulated as

$$\sum\nolimits_{k=1}^{K} \left(-(\tfrac{\gamma_0}{K} - 1)\ln r_k + c_0 e^{\ln r_k}\right) + (a_\beta + 1/2)\sum\nolimits_{j=0}^{d}\sum\nolimits_{k=0}^{K}[\ln(1 + \beta_{jk}^2/(2b_{\beta k}))]. \tag{12}$$

The above procedure provides a differentiable way of learning branch node parameters. At the end of the training algorithm, we also need to assign the leaf node parameters (that determine the tree's predictions at each leaf). We pass the whole training set through the tree and assign the empirical distribution of all categories at each leaf as its parameters. This way of determining the leaf parameters has been shown to achieve the highest AUC in binary-classification [17]. Algorithm 1 of the Appendix summarizes the training method.

Regression. For this task, we replace the mutual information optimization by a variance reduction criteria. To be more specific, we learn the tree parameters with

$$\underset{\boldsymbol{\theta}}{\text{argmin}}\{\sum\nolimits_n P_{\boldsymbol{\theta}}(\ell \,|\, \boldsymbol{x}_n)(y_n - \hat{\mu}^\ell)^2\} \tag{13}$$

such that

$$\hat{\mu}^\ell = \frac{\sum_n P_{\boldsymbol{\theta}}(\ell \,|\, \boldsymbol{x}_n) y_n}{\sum_n P_{\boldsymbol{\theta}}(\ell \,|\, \boldsymbol{x}_n)} \tag{14}$$

and the calculation of $P_{\boldsymbol{\theta}}$ is exactly the same as in the classification case. The leaf parameters are set as the mean response of the points arriving at that leaf.

Generative modeling. We motivate two key characteristics of CPT (i.e., shallower trees and differentiable training) by explicitly combining CPT and VAEs for the task of density estimation. To this end, we assign a full VAE to each of CPT's leaves and employ it to model the data points sent down to that leaf.

Our intuition behind the effectiveness of this combination comes from the properties of CPT, which can alleviate the challenges in training VAEs. Proper training of VAE is challenging, especially when the data distribution of interest shows highly multi-modal behavior. However, the severity of multi-modality can be reduced if we partition the data space into smaller segments. Since CPT is essentially a hierarchical space partitioner equipped with a differentiable training algorithm, it can effectively partition the space using the differentiable loss of VAE. Therefore, it can facilitate density estimation by providing an easier to learn distribution for each VAE.

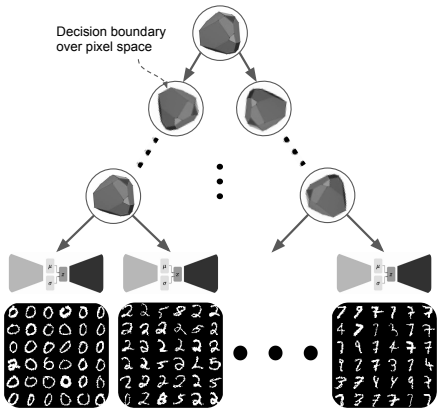


Figure 1: CPT-VAE overview.

An overview of our method, CPT-VAE, is shown in Figure 1. At a high level, to allow training of CPT and VAEs together, we use the probabilistic average of the *evidence lower bound* (ELBO) terms related to each VAE (at each leaf) as our objective function. More specifically, we maximize the following objective function:

$$\sum_{\ell \in S_{\text{leaf}}} P_{\boldsymbol{\theta}_{CPT}}(\ell \,|\, \boldsymbol{x}) \mathcal{L}(\boldsymbol{\theta}_\ell, \boldsymbol{\phi}_\ell; \boldsymbol{x}) \quad \text{such that:} \quad \mathcal{L}(\boldsymbol{\theta}_\ell, \boldsymbol{\phi}_\ell; \boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}|\boldsymbol{x})} \log \frac{p_{\boldsymbol{\theta}_\ell}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\phi}_\ell}(\boldsymbol{z}|\boldsymbol{x})} \tag{15}$$

where $\boldsymbol{\theta}_{CPT}$ are the node parameters of the decision tree as presented in the equation 6. And $\boldsymbol{\phi}_\ell, \boldsymbol{\theta}_\ell$ are the parameters of the inference (encoder) and generative (decoder) model for the VAE at leaf ℓ.

3.2.2 Topology Learning

We start by assuming the tree structure to be just a root node and its two child leaf nodes. We train this tree using the algorithm explained in Section 3.2.1. After training, we split the training set to two subsets (right and left), by thresholding the assigned probability. We calculate this threshold in the classification (regression) task, which achieves maximum mutual information (variance reduction) in the deterministic tree. To be more specific, for the classification task, we set the threshold q_{thr}, such that it minimizes

$$\frac{n_0}{N} \cdot \mathcal{H}(\{y_{(i)}\}_{i=1}^{n_0}) + \frac{N - n_0}{N} \cdot \mathcal{H}(\{y_{(i)}\}_{i=n_0+1}^{N}), \quad \text{where} \quad p_{(n_0)} \leq q_{thr} \leq p_{(n_0+1)}.$$

And $p_{(i)}$ is the i'th smallest probability assigned by the root node to the data samples. We further split each child node by considering it as root and applying the above algorithm using its data samples. We stop splitting a node with very few data samples and stop growing the tree when we reach a certain predefined maximum depth.

Note that during the proposed tree-structure greedy training, we do not perform any parameter refining using the method presented in Section 3.2.1. The parameter refining at each step of adding a new node can further improve the accuracy, but it will come with the price of increased computational complexity.

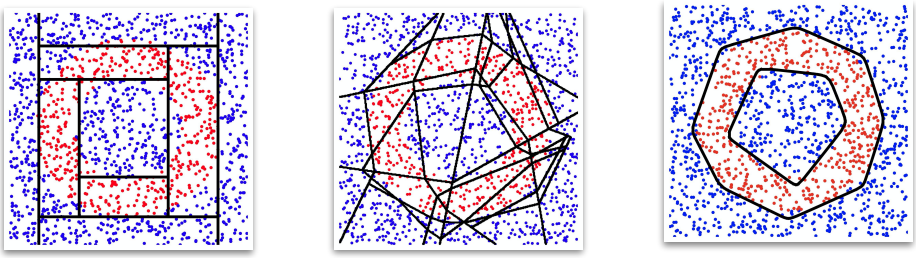
3.3 Test time

At the test time, however, we only make a hard decision at each node of the tree to send each data point right or left. Hard decision at each node is based on thresholding the probability of the noisy-OR of each internal node at 0.5. This distinguishes our method from soft (probabilistic) decision trees and puts it in the class of interpretable models.

For the generative modeling task, this way of sending the data points through the tree, makes each VAE only responsible for a part of the data distribution and facilitates the leaning the data distribution.

4 Experiments

In this section, we empirically assess the qualitative and quantitative performances of CPT on datasets from various domains. We show that CPT learns significantly smaller trees than its counterparts and



(a) CART: depth 10 tree (b) LCN: depth 10 tree (c) CPT: depth 2 tree

Figure 2: A visual comparison of the decision boundaries of CART, LCN and CPT on the synthetic 2D dataset. As demonstrated by Figures a and b, CART and LCN need to partition the space into many subspaces to achieve reasonable performance. This results in a multitude of smaller partitions which render the model harder interpret. In contrast, as shown in Figure c, CPT successfully partitions the data with a much simpler scheme (depth 2).

makes more robust and accurate predictions. That is partly due to the high variance of the leaf node's prediction in classical (oblique) trees, resulting from fewer data samples in each partition. However, since CPT usually has fewer leaf nodes, each partition has a significant proportion of the dataset.

4.1 Synthetic Dataset

In this section we compare CPT and other tree based approaches on toy examples to better demonstrate why CPT achieves better performance. We consider a dataset of 2,000 points, as shown in Figure 2. The data samples are independent draws of the uniform distribution on a two-dimensional space $[-1, 1] \times [-1, 1]$, with the data points labeled as red if they lay between two concentric circles, and blue otherwise.

We compare CPT with LCN [33], which is a state-of-the-art oblique tree method, and CART [9], which is a canonical axis-aligned tree algorithm. Figure 2 shows the decision boundaries of all three methods. CART and LCN are trained with maximum depth of 10, and CPT is trained with a maximum depth of 2. It is worth noting that both LCN and CART are restricted to partition the feature space into disjoint convex polytopes and assign each region to a leaf node. However, CPT does not have such a limitation, and each region can be the result of applying any set of logical operations on a set of convex polytopes. Figure 2 clearly shows that both LCN and CART need a large depth to successfully classify the data, while CPT only needs two splits. To be more specific, the AUC results for CART is 0.901 with 9 leaves, and 0.959 for LCN with 479 leaves.

In contrast, CPT achieves the AUC of 0.962 (the highest score) with just 3 leaf nodes at maximum depth 2. However, one limitation of our method is that it can not yet achieve competitive AUC to a multi-layer neural network.

Figure 2c shows that CPT uses a 8-sided 2D convex polytope for the first split and a 5-sided one for the second split to partition the space. This number was not fixed at training time, we only limit the maximum number of polytope sides to $K = 50$. The model owes this adaptive shrinkage to the gamma process prior, which improves the simplicity and interpretability of the model.

4.2 Classification and Regression

We evaluate the performance of CPT for regression, binary classification, and multi-class classification tasks. For regression and binary classification, we conduct experiments on datasets from MoleculeNet [51]. We follow the literature to construct features [33, 51] and use the same training, validation, and testing split as Lee and Jaakkola [33]. For multi-class classification, we perform experiments on four benchmark datasets from LibSVM [12], including MNIST, Connect4, SensIT, and Letter. We employ the provided training, validation, and testing sets when available; otherwise, we create them under the criterion specified in previous works [22, 40]. The datasets statistics are summarized in the Appendix Table 4.

Table 1: The performance of decision tree algorithms on regression (RMSE, lower is better) and classification (ACC or AUC, higher is better). The superscripts *, †, and ‡is used to to show the source of the quoted results as [53], [33], and [22].

Dataset	CART	HHCART	GUIDE	CO2	FTEM	TAO	LCN	CPT (K=1)	CPT
MNIST(ACC)	88.05±0.02*	90.1±1.2	78.52±0.20*	90±-*	96.12±-‡	94.74±0.11*	93.81±0.32	95.74±0.10	**97.01±0.20**
SensIT (ACC)	81.71±0.01*	-	79.25±0.33*	82±-*	81.61±-‡	85.12±0.20*	84.06±0.32	84.87±0.27	**85.77±0.55**
Connect4 (ACC)	78.29±0.21*	-	72.01±0.36*	78±-*	80.51±-‡	**81.09±0.39***	80.79±0.12	78.89±0.23	**81.16±0.51**
Letter (ACC)	86.07±0.14*	83.1±0.3	82.65±0.9*	87±-*	86.31±0.21	89.15±0.88*	89.64±0.75	84.13±0.32	**90.51±0.81**
PDBbind (RMSE)	1.573±0.00†	1.530±0.00†	-	-	Not Applicable	-	1.508±0.017†	1.453±-0.006	**1.413±0.004**
Bace (AUC)	65.2±2.4†	54.5±1.6†	-	-	81.03±1.5	73.4±0.0†	**83.9±1.3**†	82.06±2.4	**84.7±1.6**
HIV (AUC)	54.4±0.9†	63.6±0.0†	-	-	71.09±1.0	62.7±0.0†	**72.8±1.3**†	71.12±2.3	**73.1±1.4**

Table 2: Number of leaves and (the depth of trees). See Appendix for the full version.

Dataset	CART	HHCART	GUIDE	CO2	FTEM	TAO	LCN	CPT (K=1)	CPT
MNIST	805 (D19)*	-	**39** (D15)*	- (D14)*	357 (D12)	178 (**D8**)*	65536 (D16)	501 (D11)	98 **(D8)**
Connect4	5744 (D33)*	-	27 (D18)*	- (D16)*	257 (D10)	210 (D8)*	65536 (D16)	342 (D10)	**4 (D2)**

4.2.1 Compared Baselines

We evaluate the performance of CPT against several state-of-the-art decision tree methods, including FTEM (Hehn et al. [22], "End-to-end learning of decision trees and forests"), TAO (Carreira-Perpinán and Tavallali [11], "Oblique decision trees trained via alternating optimization"), and LCN (Lee and Jaakkola [33], "Oblique decision trees from derivatives of ReLU networks"). We also consider several additional baselines, including CART [8], HHCART [50], GUIDE [34], and CO2 [40]. Moreover, to empirically show the importance of flexible boundaries, we also added a baseline CPT with $K = 1$ (hyperplane cuts). More details about our implementation and the exact values of hyperparameters for each dataset are presented in the Appendix. Our code to reproduce the results is provided.

4.2.2 Experimental Results

Our algorithm is implemented in PyTorch and can be trained by gradient-based methods. We performed the classification and regression experiments on a laptop with a 2.5 GHz 6-Core Intel Core i7 CPU and 16 GB of RAM, for the generative modeling experiments we used an Nvidia V100 GPU. We use Adam [28] optimization for inferring the tree split parameters. A 10-fold cross-validation on the combined train and validation set is used to learn the hyperparameters, namely the maximum number of polytope sides, number of training epochs, learning rate, and batch-size. However, we decide the depth of the tree based on the performance of CPT on the validation set during training, which can be perceived as early stopping for trees. Following the literature, we use the Area Under the receiver operating characteristic Curve (AUC) on the test set as the evaluation metric for binary classification, accuracy (ACC) for multi-class classification, and root-mean-squared error (RMSE) for regression. Finally, we report the average and standard error of each method's performance by repeating our experiments for 10 random seed initializations.

Tables 1 and 2 present the results for a variety of decision tree based algorithms. Some results are quoted from previous works [22, 33, 53]. The depth and leaf numbers are averaged over 10 repetitions of training, and then rounded. For some large datasets, namely MNIST, SensIT, Connect4, and Letter, we fix the depth parameter as opposed to adaptively tuning it based on the validation set on each run. The reason for some missing values in the Table 2 is some methods like TAO did not provide their code, so we could not provide their performance on datasets they had not experimented. For each dataset, the best result and those with no statistically significant difference (by using two sample t-test and p-value of 0.05) are highlighted.

From the results, it is evident that the added flexibility in splitting rules combined with an efficient training algorithm allows CPT to outperform the baseline algorithms. Our method achieves the state-of-the-art performance, while, notably, using significantly shallower trees. For instance, CPT obtains the best performance in Connect4 with only depth 2 and 4 leaves, while other methods need a depth of at least 8.

Table 3: Test data log-likelihood scores for the **Binary MNIST** dataset, estimated with 100 importance samples. Other methods are excerpted from [26]. *OT-VAE* refers to the combination of oblique decision tree with VAE, and D is the depth of the tree.

VAE	OT-VAE($D = 4$)	CPT-VAE($D = 3$)	CPT-VAE($D = 4$)	SA	IAF	HF	RME
-84.49	-81.71	-81.52	**-81.21**	-83.64	-83.12	-83.87	-83.09

It also improves the regression performance on the PDBbind dataset by a large margin. Although LCN achieves competitive results in terms of accuracy on some datasets, it needs to grow the tree's size exponentially, significantly sacrificing the model interpretability. That is mainly because LCN, in contrast to our model, always learns a complete tree and generally needs to have a considerably large depth to achieve competitive results. For instance, consider its enormous size when trained on MNIST or Connect4 in Table 2.

4.3 CPT-VAE experiments

We evaluate the effectiveness of this combination on the standard benchmark binarized MNIST dataset [10] and compare our method with other recent VAE models, including RME [26], SA [27], IAF [30] and HF [46]. To be consistent and comparable with recent works, we follow a similar implementation of the convolutional VAE as in [26].

Table 3, presents the importance sampled estimate of the marginal likelihood for each model with 100 samples. Combining VAE with decision tree significantly outperforms other recent VAE baselines. This improvement in performance, together with the simplicity of our method of combining CPT-VAE, empirically demonstrates the advantages of using differentiable decision trees.

To ablate the effect of combining other decision trees versus using CPT, we also include OT-VAE as another baseline, which uses oblique cuts at each decision node. Table 3 shows that CPT outperforms OT-VAE even with a shallower tree (half the leaves). In other words, CPT still achieves slightly better test log-likelihood, with nearly half the computation power required due to requiring fewer VAEs, which are the computational bottleneck. This further highlights the importance of convex polytope cuts over oblique cuts. We also provided the contingency table for leaf ID versus digit labels in the Appendix, demonstrating CPT's effective partitioning of the space.

5 Conclusion

We propose convex polytope trees (CPT) as a generalization to the class of oblique trees that improves their accuracy and shrinks their size, which consequently provides more robust predictions. CPT owes its performance to two main components: flexible decision boundaries and an efficient training algorithm. The proposed training method further allowed us to broaden the applicability of decision trees to generative modeling.

We also demonstrate the efficacy of CPT on a variety of tasks and datasets. The empirical successes of CPT show promise to further investigate other interpretable generalizations of decision boundaries. Another promising direction for future work is to study the combination of CPT with various ensemble methods, such as bagging and boosting. Moreover, CPT-VAE demonstrates the potential advantages of combining differentiable trees with other downstream tasks.

6 Broader Impact

One often-neglected side effect of improving the interpretability of an ML model is the risk of automation bias and a false sense of trust in models. The ease of understanding the system increases the risk of misinterpretation, overtrust, and even incorrect use of the outputs. Our model's improvement of interpretability and accuracy of decision trees, which is usually employed in high stake domains, can further increase the chance of that risk.

References

[1] A. Ahmad. Decision tree ensembles based on kernel features. *Applied intelligence*, 41(3): 855–869, 2014.

[2] F. Aiolli and A. Sperduti. Multiclass classification with multi-prototype support vector machines. *Journal of Machine Learning Research*, 6(May):817–850, 2005.

[3] R. G. Almond, R. J. Mislevy, L. S. Steinberg, D. Yan, and D. M. Williamson. *Bayesian networks in educational assessment*. Springer, 2015.

[4] K. P. Bennett. Global tree optimization: A non-greedy decision tree algorithm. *Computing Science and Statistics*, pages 156–156, 1994.

[5] D. Bertsimas and J. Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

[6] S. Bitzer, O. Ribaux, N. Albertini, and O. Delémont. To analyse a trace or not? Evaluating the decision-making process in the criminal investigation. *Forensic science international*, 262:1–10, 2016.

[7] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[8] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[9] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[10] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *ICLR*, 2015.

[11] M. A. Carreira-Perpinán and P. Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. In *Advances in Neural Information Processing Systems*, pages 1211–1221, 2018.

[12] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines," 2001. software available at http://www. csie. ntu. edu. tw/˜ cjlin/libsvm. 2001.

[13] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[14] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine learning*, 40(2):139–157, 2000.

[15] H. Drucker and C. Cortes. Boosting decision trees. In *Advances in neural information processing systems*, pages 479–485, 1996.

[16] T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230, 1973.

[17] C. Ferri, P. Flach, and J. Hernández-Orallo. Learning decision trees using the area under the ROC curve. In *Icml*, volume 2, pages 139–146, 2002.

[18] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[19] N. Frosst and G. Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

[20] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media, 2009.

[21] D. Heath, S. Kasif, and S. Salzberg. Induction of oblique decision trees. In *IJCAI*, volume 1993, pages 1002–1007, 1993.

[22] T. M. Hehn, J. F. Kooij, and F. A. Hamprecht. End-to-end learning of decision trees and forests. *International Journal of Computer Vision*, pages 1–15, 2019.

[23] T. S. Jaakkola and M. I. Jordan. Variational probabilistic inference and the QMR-DT network. *Journal of artificial intelligence research*, 10:291–322, 1999.

[24] M. I. Jordan. A statistical approach to decision tree modeling. In *Machine Learning Proceedings 1994*, pages 363–370. Elsevier, 1994.

[25] A. Kantchelian, M. C. Tschantz, L. Huang, P. L. Bartlett, A. D. Joseph, and J. D. Tygar. Large-margin convex polytope machine. In *Advances in Neural Information Processing Systems*, pages 3248–3256, 2014.

[26] M. Kim and V. Pavlovic. Recursive inference for variational autoencoders. *Advances in Neural Information Processing Systems*, 2020.

[27] Y. Kim, S. Wiseman, A. Miller, D. Sontag, and A. Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2678–2687. PMLR, 2018.

[28] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *ICLR*, 2014.

[30] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving variational inference with inverse autoregressive flow. *Advances in Neural Information Processing Systems*, 2016.

[31] J. Kingman. *Poisson Processes*. Oxford Studies in Probability. Clarendon Press, 1992. ISBN 9780191591242.

[32] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.

[33] G.-H. Lee and T. S. Jaakkola. Oblique decision trees from derivatives of relu networks. *ICLR*, 2020.

[34] W.-Y. Loh. Fifty years of classification and regression trees. *International Statistical Review*, 82(3):329–348, 2014.

[35] N. Manwani and P. Sastry. Learning polyhedral classifiers using logistic function. In *Proceedings of 2nd Asian Conference on Machine Learning*, pages 17–30, 2010.

[36] N. Manwani and P. Sastry. Polyceptron: A polyhedral learning algorithm. *arXiv preprint arXiv:1107.1564*, 2011.

[37] A. Martinez-Nicolas, J. A. Madrid, F. García, M. Campos, M. T. Moreno-Casbas, P. F. Almaida-Pagán, A. Lucas-Sánchez, and M. A. Rol. Circadian monitoring as an aging predictor. *Scientific reports*, 8(1):1–11, 2018.

[38] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht. On oblique random forests. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 453–469. Springer, 2011.

[39] S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of artificial intelligence research*, 2:1–32, 1994.

[40] M. Norouzi, M. Collins, M. A. Johnson, D. J. Fleet, and P. Kohli. Efficient non-greedy optimization of decision trees. In *Advances in neural information processing systems*, pages 1729–1737, 2015.

[41] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[42] J. G. Richens, C. M. Lee, and S. Johri. Improving the accuracy of medical diagnosis with causal machine learning. *Nature communications*, 11(1):1–9, 2020.

[43] M. A. Shwe, B. Middleton, D. E. Heckerman, M. Henrion, E. J. Horvitz, H. P. Lehmann, and G. F. Cooper. Probabilistic diagnosis using a reformulation of the internist-1/qmr knowledge base. *Methods of information in Medicine*, 30(04):241–255, 1991.

[44] A. Suárez and J. F. Lutsko. Globally optimal fuzzy decision trees for classification and regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1297–1311, 1999.

[45] M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244, 2001.

[46] J. M. Tomczak and M. Welling. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*, 2016.

[47] G. Valdes, J. M. Luna, E. Eaton, C. B. Simone II, L. H. Ungar, and T. D. Solberg. MediBoost: A patient stratification tool for interpretable decision making in the era of precision medicine. *Scientific reports*, 6:37854, 2016.

[48] V. Vapnik and A. Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, 1971.

[49] Z. Wang, N. Djuric, K. Crammer, and S. Vucetic. Trading representability for scalability: Adaptive multi-hyperplane machine for nonlinear classification. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 24–32, 2011.

[50] D. Wickramarachchi, B. Robertson, M. Reale, C. Price, and J. Brown. HHCART: An oblique decision tree. *Computational Statistics & Data Analysis*, 96:12–23, 2016.

[51] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2): 513–530, 2018.

[52] Y. Yang, I. G. Morillo, and T. M. Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

[53] A. Zharmagambetov, S. S. Hada, and M. Á. Carreira-Perpiñán. An experimental comparison of old and new decision tree algorithms. *arXiv preprint arXiv:1911.03054*, 2019.

[54] M. Zhou. Parsimonious Bayesian deep networks. In *Advances in Neural Information Processing Systems*, pages 3190–3200, 2018.

[55] Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. CRC press, 2012.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 4; our results, in term of accuracy, is sub-optimal to neural networks.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Broader Impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Section 3 and A.1.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We reported the standard deviation of reported results in Table 1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix B.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix B.
   (b) Did you mention the license of the assets? [Yes] See supplemental materials.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]