Pruning Decision Trees via Max-Heap Projection

Zhi Nie* Binbin Lin † Shuai Huang ‡ Naren Ramakrishnan § Wei Fan ¶ Jieping Ye†

Abstract

The decision tree model has gained great popularity both in academia and industry due to its capability of learning highly non-linear decision boundaries, and at the same time, still preserving interpretability that usually translates into transparency of decision-making. However, it has been a longstanding challenge for learning robust decision tree models since the learning process is usually sensitive to data and many existing tree learning algorithms lead to overfitted tree structures due to the heuristic and greedy nature of these algorithms. Pruning is usually needed as an ad-hoc procedure to prune the tree structure, which is, however, not guided by a rigorous optimization formulation but by some intuitive statistical justification. Motivated by recent developments in sparse learning, in this paper, we propose a novel formulation that recognizes an interesting connection between decision tree post-pruning and sparse learning, where the tree structure can be embedded as constraints in the sparse learning framework via the use of a max-heap constraint as well as a sparsity constraint. This novel formulation leads to a non-convex optimization problem which can be solved by an iterative shrinkage algorithm in which the proximal operator can be solved by an efficient max-heap projection algorithm. A stability selection method is further proposed for enabling robust model selection in practice and guarantees the selected nodes preserve tree structure. Extensive experimental results demonstrate that our proposed method achieves better predictive performance than many existing benchmark methods across a wide range of real-world datasets.

1 Introduction

Recently there has been a renewed interest in tree based methods among researchers in the field of data mining and machine learning [2, 28, 12, 4]. This is probably due to the fact that the tree based methods have demonstrated impressive prediction performance in a great variety of recent applications like ranking [3], recognition [18, 7], recommendation [1]. On the other hand, unlike some nonlinear models such as kernel methods that generate predictions in a black-box fashion, the tree-based methods produce rules that can be easily interpreted and validated by domain knowledge or expert judgment, and can be flexibly revised based on new data or any other form of human knowledge intervention. The ease of interpretability usually leads to a higher likelihood of adaptability in real-world applications as critical decision-making tools. Furthermore, due to the unique hierarchical structure [14], tree-based models support sequential prediction, which is more cost-effective since only a few variables are needed to produce a prediction, while in other predictive models such as regression models, all the variables are needed to make a prediction.

Despite the aforementioned advantages, the tree models have been suffering from some longstanding limitations. Overfitting is one of the well known notorious problems. For example, off-the-shelf decision tree learning algorithms such as C5.0 and Classification And Regression Trees (CART, [6]) tend to create over-complex trees that may not generalize to unseen data very well. These methods also have several hard-to-tune parameters which result in barriers for their usage in real-world applications. Furthermore, due to the lack of an explicit optimization formulation which could oversee the tree learning process and gear the learned tree toward the defined optimality (i.e., that ensures better generalizability), many of the existing methods rely on heuristics that heavily depend on the training data, leading to unstable or unreliable tree models.

To mitigate the overfitting problem, one school of thought is to employ an ad hoc pruning procedure to prune the tree structure in the hope to preserve the major skeleton that can generalize well on new data. Over the past decades, there have been a number of benchmark pruning methods being developed, such as reduced error pruning [25], pessimistic error pruning [24], etc. Many of these methods date back to 1980s and are based more or less on heuristics with statistical justification rather than an integrated optimization formulation [21]. This is a common limitation for other pruning methods such as the ones that attempt to exploit some information theoretic measure to prune a decision tree [19].

*Arizona State University. zhi.nie@asu.edu
†University of Michigan, Ann Arbor. {bilin, jpye}@umich.edu
‡University of Washington. shuaih@u.washington.edu
§Virginia Tech. naren@cs.vt.edu
¶Baidu. fanwei03@baidu.com

More recently the RuleFit [10] algorithm has been developed that extracts rules from trees since rules can provide powerful basis functions to approximate highly nonlinear functions. For instance, RuleFit aims to build a prediction model as a weighted combination of the nodes in the decision trees learned by an ensemble learning method, where each node in a tree is regarded as a rule function. This rule function takes the form as an indicator function indicating whether or not the conjunction of conditions associated with edges on the path from the root node of the tree to the node concerned is satisfied. By viewing that each decision tree is a collection of rule functions, RuleFit primarily focuses on how to select a small subset of rules derived from multiple decision trees to best predict the response variable without giving consideration to the tree structure inherently existing among the rules. Without imposing the tree structure in learning the best rule combination, RuleFit leads to non-exclusive rules where other prediction mechanisms such as regression models need to be used to combine the learned rules to generate a prediction. Another work along this line of research is the Regularized Greedy Forest [12] which does take into account the structure of the trees. However, it does so by merely introducing into the objective function a regularization term which specifically concern with preventing the trees in the forest from growing too deep. Appel et al. [2] made another attempt to mitigate the problem from another perspective. In their effort to speed up tree training, they prune those underachieving features through training on progressively larger subsets of samples. Therefore, there is still a lack of methodology, particularly, a lack of explicit optimization formulation, that can achieve optimal balance between the control of the tree complexity and the integrity of the tree structure, motivating the proposed research in this paper.

Recent developments in sparse learning and optimization enable us to address the tree pruning problem by formulating it as a sparse optimization problem. In this paper, we concern ourselves with post-pruning of a single decision tree. We follow the same practice adopted by the RuleFit by treating each node in the decision tree as a rule function, but take into consideration the tree structure that exists among the rules. Specifically, we propose a novel non-convex formulation for post-pruning of a decision tree that induces sparsity of weights of the nodes by appending a l_1 regularization term and requires the weights of nodes to satisfy the max-heap constraint. That is, as the tree structure implies, for each edge connecting a parent node and child node, the absolute value of the weight of the parent node should be no less than the absolute value of the weight of the child node. It can be shown that the feasible set of this problem actually consists of a union of subspaces, leading to a non-convex optimization problem. In spite of the non-convex nature of this problem, we show that by the use of the concept known as proximal map, we can convert the non-convex optimization problem into a series of optimization problems that are not only convex but also smooth and can be efficiently solved by the method proposed in [17]. In this way, we could easily extend this sparse optimization model to study a broad class of general tree pruning problems by incorporating the prior information as a regularizer or constraint. Moreover, in order to overcome the model selection problem and enable robust performance of the proposed method, we propose a stability selection approach to select a robust weight vector among different subsampling and regularization parameters. We also prove that the selected weight vector will satisfy the tree constraint. Finally, through extensive experiments, we demonstrate that our proposed method achieves better predictive performance than many existing benchmark pruning methods across a wide range of real-world datasets.

The main contributions of this paper are as follows: (1) We propose a novel non-convex formulation for post-pruning of a decision tree based on the ℓ_1 regularization and the max-heap constraint; (2) We develop an efficient algorithm to solve the proposed formulation based on the proximal method; (3) We propose a stability selection approach to improve the robustness of the resulting decision tree model; (4) We conduct extensive experiments using 19 data sets to evaluate the effectiveness of the proposed approach.

2 Background

In this section, we will review the basic concept about the rule function and show how it can be applied to represent a tree, which actually lays the first step to connect the tree models with sparse learning as adopted in RuleFit.

2.1 Rule function Let $T = (V, E)$ be a decision tree obtained through some decision tree training procedure such as CART. It consists of a node set $V = \{v_0, v_1, v_2, \cdots, v_p\}$ and an edge set $E = \{(v_i, v_j) | v_j \text{ is a child node of } v_i\}$. Each node v_i represents a "rule function" defined by the conjunction of all conditions associated with the edges on the path from the root of the tree to that node. As illustrated in Figure 1, the rule functions at nodes 1, 4, 10 are

$$v_1(\mathbf{d}) = I(\mathbf{d}[20] \leq 0.5);$$
$$v_4(\mathbf{d}) = I(\mathbf{d}[20] \leq 0.5) \cdot I(\mathbf{d}[5] > 1);$$
$$v_{10}(\mathbf{d}) = I(\mathbf{d}[20] > 0.5) \cdot I(\mathbf{d}[6] > 0) \cdot I(\mathbf{d}[20] > 1.5).$$

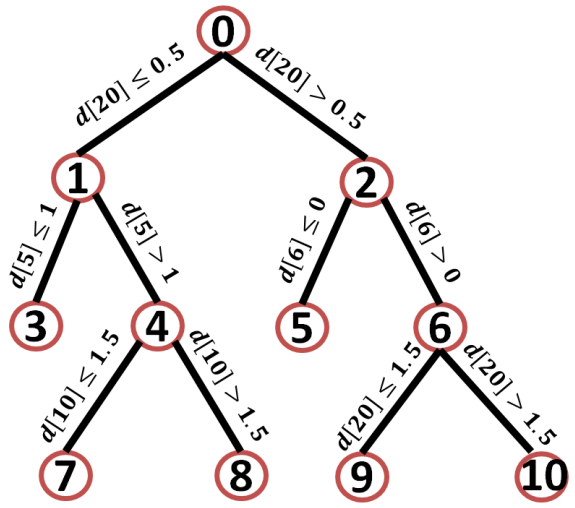


Figure 1: A decision tree. The rule function corresponding to each node can be represented as the product of indicator functions associated with edges on the path connecting the node to the root.

where $\mathbf{d}$ is a row vector representing a sample and I is the indicator function. For simplicity, we only show the cases where the nodes are associated with continuous features. The rule function can be defined with respect to categorical features, e.g., instead of specifying ranges for continuous features, non-trivial subset of domain of the corresponding categorical features can be used in defining the rule functions.

With the use of the rule function, a tree can be represented by a collection of rule functions (each node v_i contributes a rule function except the root node). This collection of rule functions can be viewed as a new basis so any sample can be represented using this basis. Specifically, suppose that we have a dataset $\mathbf{D} = [\mathbf{d}_1; \mathbf{d}_2; \cdots; \mathbf{d}_n]$ with their response denoted as $\mathbf{y} = [y_1; y_2; \cdots; y_n]$ where n is the number of samples in the dataset (here we use the MATLAB syntax ";" to denote vertical concatenation). Then, denote $\mathbf{x}_i = v_i(\mathbf{D})$ which is actually the projection of the dataset onto the basis v_i. v_i maps each row of $\mathbf{D}$ into 0 or 1 depending on whether the corresponding sample satisfies all the conditions included in v_i. Thus, with $p+1$ nodes in the tree T and their corresponding the rule functions, we can get a new representation of the original dataset $\mathbf{D}$ which we write as $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_p]$ while the response is still $\mathbf{y}$ (the MATLAB syntax "," is used to denote horizontal concatenation). By using this new representation of the original dataset $\mathbf{D}$, we can formulate rule selection problem as a feature selection problem as adopted in RuleFit. Note that since there is no rule associated with the root, we define $v_0(\mathbf{d}) = 1$ and x_0 as a column vector of 1s.

2.2 RuleFit RuleFit seeks to learn an ensemble of rule functions derived from a set of trees learned by an ensemble learning method such as random forest. It further uses linear combination of the rules to generate predictions, so each rule function v_i is associated with a weight w_i whose magnitude reflects the significance of the corresponding rule. With a collection of K rules $\{v_1, v_2, \cdots, v_K\}$ derived from a set of M trees, $\{T_1, T_2, \cdots, T_M\}$, we can formulate RuleFit as the following optimization problem:

(2.1)
$$\min_{\{w\}_{k=1}^K, b} \quad \sum_{i=1}^{n} L\left(y_i, b + \sum_{k=1}^{K} w_k v_k(\mathbf{d}_i)\right) + \lambda \sum_{k=1}^{K} |w_k|$$

where L is a general loss function. As the formulation of RuleFit implies, in its course of seeking a small subset of rules to explain the observed response, it completely ignores the inherent tree structure existing among the rules. Using the package provided by its authors, we thoroughly studied the RuleFit on some real-world datasets and found that the best prediction performance is usually attained when the collection of trees mainly consists of tree stumps and the selected set of rules is large. This is probably due to the fact that the decision tree tends to overfit the training samples when it is built to a great depth. The l_1-norm regularization term used in RuleFit does not effectively control the structural risk since nodes deep down the tree tend to be selected in order to have a sparse solution with small loss instead of those nodes that are closer to the root which are less complex and are less prone to overfitting. Furthermore, without imposing the tree structure in learning the best rule combination, RuleFit leads to non-exclusive rules where other prediction mechanisms such as regression models need to be used to combine the learned rules to generate a prediction.

3 The Proposed Decision Tree Pruning Algorithm

In this paper, we focus on post-pruning a single decision tree. The unique aspect of our idea of pruning a decision tree is to parameterize the pruning process by using the rule functions to represent a tree and further translating the hierarchical structure of the tree into max-heap constraints. By parameterizing the tree learning process, it paves the way for developing sparse learning formulations. Specifically, recall that each node (except the root node) corresponds to a rule while each rule is associated with a weight, the max-heap constraint requires that the magnitude of the weight of an ancestor node to be greater than or equal to that of an descendant node. In other words, for any i, j $(1 \leq i, j \leq p)$, if (v_i, v_j) is an edge in T, then the

weight vector $\mathbf{w} = (w_0, w_1, \cdots, w_p)^T$ satisfies max-heap constraint if $|w_i| \geq |w_j|$. For notational convenience, let us denote the set of weights that satisfy the max-heap constraint as $P = \{\mathbf{w} \mid |w_i| \geq |w_j|, \text{if } (v_i, v_j) \in E\}$. Pruning the tree could be achieved by imposing both the sparsity constraint and max-heap constraint on $\mathbf{w}$ since once the weight of a node becomes zero, the weight of all its descendants node will be zero.

3.1 Proposed Formulation Following the aforementioned idea, we can formulate the problem of tree pruning via max-heap projection as the optimization problem below:

$$\min_{\mathbf{w},b} L(\mathbf{y}, F(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_p; \mathbf{w}, b)) + \lambda \|\mathbf{w}\|_1 \quad s.t. \mathbf{w} \in P, \tag{3.2}$$

where λ is the regularization parameter controlling the sparsity of $\mathbf{w}$, L is a general loss function with respect to x_i, $\mathbf{w}$ and b, and prediction function F takes the following form:

$$F(\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_p; \mathbf{w}, b) = b\mathbf{1}_n + \sum_{i=0}^{p} w_i x_i.$$

Once a decision tree has been learned, the features and the threshold associated with each node is fixed. In other words, all $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_p$ are fixed during the decision tree post-pruning stage. For simplicity of notation, we denote $\mathbf{X} = [\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_p]$ and write $L(\mathbf{y}, F(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_p; \mathbf{w}, b))$ as $H_{(\mathbf{X},\mathbf{y})}(\mathbf{w}, b)$. With this, we can rewrite (3.2) as

$$\min_{\mathbf{w},b} H_{(\mathbf{X},\mathbf{y})}(\mathbf{w}, b) + \lambda \|\mathbf{w}\|_1 \quad s.t. \mathbf{w} \in P. \tag{3.3}$$

Note that to preserve integrity of the tree structure, we keep the constant x_0 and its coefficient w_0 in our model. However, their effects can be compensated by the intercept term on which no constraint has been imposed. In our experiments, we use least square loss for $H_{(\mathbf{X},\mathbf{y})}(\mathbf{w}, b)$.

3.2 Proposed Algorithm Due to the fact that the feasible set of the resulting optimization problem formulated above can be represented as a union of multiple non-overlapping convex cones, it is essentially a non-convex problem. One effective approach to solving this problem is to employ the General Iterative Shrinkage Threshold (GIST) [11] framework. At the heart of GIST framework lies in solving the proximal map for the problem through which a sequence of points will be generated towards a local optimal solution under certain conditions. We found that by decomposing the proximal map into an element-wise product of signs and magnitudes, finding proximal map can be converted into a smooth convex problem which can be efficiently solved by some existing tools [16].

Let

$$\sigma(\mathbf{w}) = \begin{cases} \lambda \|\mathbf{w}\|_1, & \text{if } \mathbf{w} \in P \\ +\infty, & \text{if } \mathbf{w} \notin P. \end{cases} \tag{3.4}$$

At the point $\left(\mathbf{w}^k, b^k\right)$, the proximal map is

$$\begin{aligned} &\left(\mathbf{w}^{k+1}, b^{k+1}\right) \\ &= \arg\min_{\mathbf{w},b} \left\langle \nabla_{\mathbf{w}} H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right), \mathbf{w} - \mathbf{w}^k \right\rangle \\ &\quad + \left\langle \nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right), b - b^k \right\rangle + \frac{t^k}{2} \|\mathbf{w} - \mathbf{w}^k\|_2^2 \\ &\quad + \frac{t^k}{2} \|b - b^k\|_2^2 + \sigma(\mathbf{w}) \end{aligned} \tag{3.5}$$

where $t^k > 0$ can be found through line search. After rearranging the terms and removing some constant terms, we have

$$\begin{aligned} &\left(\mathbf{w}^{k+1}, b^{k+1}\right) \\ &= \arg\min_{\mathbf{w},b} \frac{t^k}{2} \left\| \mathbf{w} - \left(\mathbf{w}^k - \frac{\nabla_{\mathbf{w}} H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right)}{t_k} \right) \right\|_2^2 + \sigma(\mathbf{w}) \\ &\quad + \frac{t^k}{2} \left\| b - \left(b^k - \frac{\nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right)}{t^k} \right) \right\|_2^2 \end{aligned} \tag{3.6}$$

Obviously, $b^{k+1} = b^k - \frac{\nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right)}{t^k}$. Let $\mathbf{u}^k = \mathbf{w}^k - \frac{\nabla_{\mathbf{w}} H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^k, b^k\right)}{t_k}$. Finding $\mathbf{w}^{k+1}$ amounts to solving the following problem:

$$\begin{aligned} \mathbf{w}^{k+1} &= \arg\min_{\mathbf{w}} \frac{1}{2} \left\| \mathbf{w} - \mathbf{u}^k \right\|_2^2 + \frac{\lambda}{t^k} \|\mathbf{w}\|_1 \\ s.t. \quad & \mathbf{w} \in P \end{aligned} \tag{3.7}$$

Next, we show that the optimal solution $\mathbf{w}^{k+1}$ yielded by (3.7) shares the same sign as $\mathbf{u}^k$ in an element-wise way. To simplify notation, let us define sign$(\cdot)$ as the sign function of a vector that returns the element-wise sign of the vector.

LEMMA 1. *Let $\mathbf{w}^{k+1}$ be the optimal solution yielded by (3.7), then we have* $\text{sign}\left(\mathbf{w}^{\mathbf{k+1}}\right) = \text{sign}\left(\mathbf{u}^{\mathbf{k}}\right)$.

Proof. This can be easily verified by contradiction. Suppose that the j-th element of the $\mathbf{w}^{k+1}$, w_j^{k+1} does not have the same sign as the j-th element of $\mathbf{u}^k$, u_j^k. Let

Algorithm 1 Max-heap Based Decision tree Pruning Algorithm

Require:
$X = \{x_0, x_1, \ldots, x_p\}$, $\lambda \in \mathbb{R}$, $\eta > 1$,initial weight and intercept $\mathbf{w}^{(0)}$, $b^{(0)}$, t_{min}, t_{max} with $0 < t_{min} < t_{max}$.

$k \leftarrow 0$
repeat
  $t^{(k)} \in [t_{min}, t_{max}]$
  repeat
    $\mathbf{u}^k = \mathbf{w}^{(k)} - \frac{\nabla_{\mathbf{w}} H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^{(k)}, b^{(k)}\right)}{t^{(k)}}$
    $\mathbf{u}^{(k)^+} = \text{abs}\left(\mathbf{u}^{(k)}\right)$
    $\mathbf{w}^{(k+1)^+} \leftarrow$ solution to the problem (3.7)
    $\mathbf{w}^{(k+1)} \leftarrow \text{sign}(\mathbf{u}^{(k)}) \circ \mathbf{w}^{(k+1)^+}$
    $\mathbf{b}^{(k+1)} = b^{(k)} - \frac{\nabla_b H_{(\mathbf{X},\mathbf{y})}\left(\mathbf{w}^{(k)}, b^{(k)}\right)}{t^{(k)}}$
    $t^{(k)} \leftarrow \eta t^{(k)}$
  until some line search criterion is satisfied.
  $k \leftarrow k + 1$
until some stop criterion is satisfied
Output: $\mathbf{w}^{(k)}, b^{(k)}$.

$\hat{\mathbf{w}}^{k+1}$ be a column vector with the i-th element denoted as $\hat{w}_i^{k+1}$. Let $\hat{w}_i^k = \hat{w}_i^{k+1}, \forall i \neq j$, $\hat{w}_j^{k+1} = -\hat{w}_j^{k+1}$. Since the magnitude of each element of $\hat{\mathbf{w}}^{k+1}$ is the same as that of corresponding entry in $\mathbf{w}^{k+1}$, $\hat{\mathbf{w}}^{k+1}$ is a feasible solution and has the same 1-norm as $\mathbf{w}^{k+1}$, but $\left\|\mathbf{w}^{k+1} - \mathbf{u}^k\right\|_2^2 \geq \left\|\hat{\mathbf{w}}^{k+1} - \mathbf{u}^k\right\|_2^2$. Thus $\mathbf{w}^{k+1}$ is a less optimal solution than $\hat{\mathbf{w}}^{k+1}$, which is in contradiction with our assumption.

Denote the vector obtained from taking element-wise absolute value of $\mathbf{u}^k$ as $\mathbf{u}^{k^+}$, we have $\mathbf{u}^k = \text{sign}(\mathbf{u}^k) \circ \mathbf{u}^{k^+}$, where "$\circ$" represents element-wise multiplication. Knowing that the optimal solution $\mathbf{w}^{k+1}$ shares the same sign as $\mathbf{u}^k$, we can further convert the problem (3.7) to

$$\begin{aligned} \mathbf{w^{k+1}}^+ = &\arg\min_{\mathbf{w}} \frac{1}{2}\left\|\mathbf{w} - \mathbf{u}^k\right\|_2^2 + \frac{\lambda}{t^k}\mathbf{w}^T\mathbf{1}_{p+1} \\ s.t. \quad &\mathbf{w} \succeq 0 \\ &w_i \geq w_j, \quad \forall (v_i, v_j) \in E, 0 \leq i, j \leq p. \end{aligned} \tag{3.8}$$

The optimal solution $\mathbf{w}^{k+1} = \text{sign}(\mathbf{u}^k) \circ \mathbf{w^{k+1}}^+$. The problem (3.8) can be efficiently solved by the method proposed in [17]. The framework of the algorithm is described in Algorithm 1.

As described in [17], the key step in solving the problem of finding proximal map is to recursively identify the maximal root-tree and remove it from the original tree. It can be shown that such a process can lead to unique optimal solution. As mentioned in the paper, this process has an expected linear complexity. We refer interested readers to [17] for more details.

4 Stability Selection

In order to optimize the decision tree structure, we have proposed a max-heap projection method to solve the sparse optimization problem with the tree constraint in the previous section. The sparsity of the weight vector, which determines the tree structure, is controlled by the regularization parameter. In practice, the selection of the regularization parameter is usually data-dependent and sometimes sensitive to the noise of the data. To obtain a more robust tree model, we propose to apply stability selection for model selection. Stability selection is a technique for feature selection. But here, since each feature represents a tree node, we adapt it to be a method for tree structure selection that guarantees the selected nodes constitute a tree.

Specifically, given a data set $\mathbf{X}$ and a response variable vector $\mathbf{y}$, we split $\mathbf{Z} = \{\mathbf{X}, \mathbf{y}\}$ to three parts, training data set $\mathbf{Z}_{tr}$, pruning data set $\mathbf{Z}_{pr}$ and testing data set $\mathbf{Z}_{te}$. Since the tree pruning process is equivalent to the learning of the weight vector $\mathbf{w}$, to account for the uncertainty of $\mathbf{w}$ induced by noise in training data set $\mathbf{Z}_{tr}$ and different choices of λ, the basic idea of stability selection is to resample the training data set $\mathbf{Z}_{tr}$ many times and learn the weight vector $\mathbf{w}$ on these resampled datasets across a range of λ. Then, the uncertainty of the elements in $\mathbf{w}$ can be evaluated (i.e., via the support vector of $\mathbf{w}$ that will be defined below) and only the elements that tend to be selected most frequently should be kept in the final tree structure.

4.1 Select the Tree Structure for a Fixed λ

Given any subsampled training set $\mathbf{Z}^{(i)} = \{\mathbf{X}^{(i)}, \mathbf{y}^{(i)}\}$ from $\mathbf{Z}_{tr}$, the weight vector can be obtained as follows:

$$\mathbf{w}(\mathbf{Z}^{(i)}, \lambda) = \underset{\mathbf{w} \in P}{\text{argmin}}\, H_{\mathbf{Z}^{(i)}}(\mathbf{w}, b) + \lambda\|\mathbf{w}\|_1$$

We assume there are s subsampled training data sets, denoted by $\mathbf{Z}^{(1)}, \ldots, \mathbf{Z}^{(s)} \subset \mathbf{Z}_{\text{tr}}$. The number of data points in each $\mathbf{Z}^{(i)}$ is about half of the number of data points $\mathbf{Z}_{tr}$. Given a fixed λ, for each $\mathbf{Z}^{(t)}$ there is a weight vector $\mathbf{w}(\lambda, \mathbf{Z}^{(t)})$. The support of the weight vector, which means the locations of nonzero entries in the weight vector, determines the tree structure. Let $\mathbf{s}(\lambda, \mathbf{Z}^{(t)})$ denote a binary vector such that $\mathbf{s}(\lambda, \mathbf{Z}^{(t)}) = 1$ if $\mathbf{w}(\lambda, \mathbf{Z}^{(t)}) \neq 0$, $\mathbf{s}(\lambda, \mathbf{Z}^{(t)}) = 0$ otherwise. Examples of the support vectors are shown in Figure 2. Clearly $\mathbf{s}(\lambda, \mathbf{Z}^{(t)})$ belongs to P and $\mathbf{s}(\lambda, \mathbf{Z}^{(t)})$ induces the same tree structure as $\mathbf{w}(\lambda, \mathbf{Z}^{(t)})$ does.

Given a fixed λ, we aim to learn a support vec-

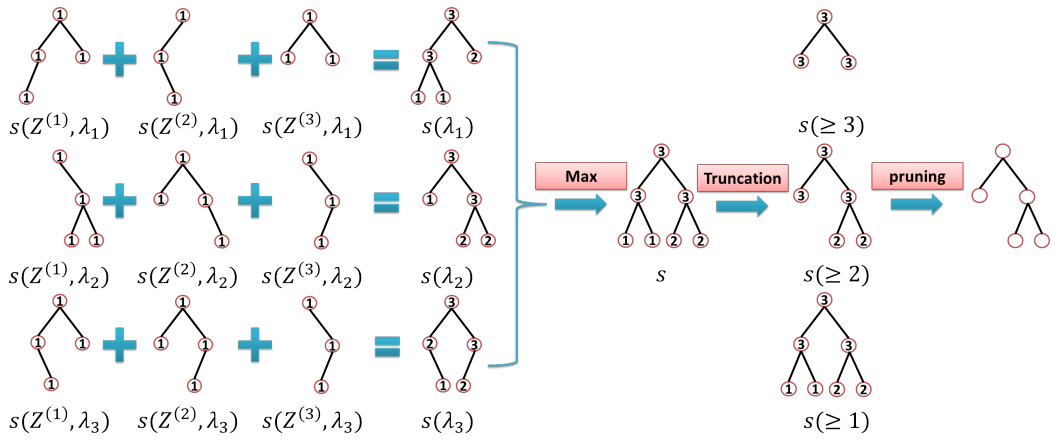


Figure 2: Stability selection. Given subsampled data sets $\mathbf{Z}^{(1)}, \mathbf{Z}^{(2)}, \mathbf{Z}^{(3)}$ and regularization parameters $\{\lambda_1, \lambda_2, \lambda_3\}$, we first calculate $\{\mathbf{s}(\lambda_1), \mathbf{s}(\lambda_2), \mathbf{s}(\lambda_3)\}$ by adding from $\mathbf{s}(\mathbf{Z}^{(1)}, \lambda_i)$ to $\mathbf{s}(\mathbf{Z}^{(3)}, \lambda_i)$. Then we apply the max operator on $\{\mathbf{s}(\lambda_1), \mathbf{s}(\lambda_2), \mathbf{s}(\lambda_3)\}$ to obtain $\mathbf{s}$. By truncating different values on $\mathbf{s}$, we get several candidates of the tree structure. The tree structure will be selected by evaluating the performance on the pruning data set.

tor $\mathbf{s}(\lambda)$ from $\{\mathbf{s}(\lambda, \mathbf{Z}^{(1)}), \ldots, \mathbf{s}(\lambda, \mathbf{Z}^{(s)})\}$. Since the frequency of the components of the weight vector reflects its importance and probability, we simply define the new support vector as a summation of these support vectors,

$$\mathbf{s}(\lambda) := \sum_{t=1}^{s} \mathbf{s}(\lambda, \mathbf{Z}^{(t)}). \tag{4.9}$$

We can show that the new support vector $\mathbf{s}(\lambda)$ still has a tree structure.

PROPOSITION 4.1. *The new support vector* $\mathbf{s}(\lambda) \in P$.

Proofs of Proposition 4.1 and Proposition 4.2 are included in the supplement.

4.2 Select the Tree Structure for Multiple λ's

Next we would like to select a robust tree structure among multiple support vectors learned from different λ. Given a set of regularization parameters $\{\lambda_1, \ldots, \lambda_k\}$, there are k support vectors $\mathbf{s}(\lambda_1), \ldots, \mathbf{s}(\lambda_k)$, which can be computed by (4.9). In this case, the max selection rule is widely used and has a strong theoretical guarantee [20]. Therefore, we propose to select the maximum value among $\{\mathbf{s}(\lambda_1), \ldots, \mathbf{s}(\lambda_k\}$ for each node of the tree. Specifically, we define the new support vector as follows:

$$\mathbf{s}_i := \max\{\mathbf{s}_i(\lambda_1), \ldots, \mathbf{s}_i(\lambda_k)\}, \ \forall i.$$

Interestingly, the new support vector still has a tree structure.

PROPOSITION 4.2. *Let* $\mathbf{s}$ *be formed by taking maximum at each entry over the support vectors computed from multiple* λ*'s. Then* $\mathbf{s} \in P$.

By truncating different values on the support vector, we obtain several candidates of the tree structure. The final tree structure will be selected by evaluating the performance on the pruning data set. The whole procedure is illustrated in Figure 2.

5 Experiments

In this section, we evaluate the proposed method for pruning decision trees on a variety of real-world datasets of different sizes and levels of difficulty from UCI machine learning repository [15] and some other sources. We compare the predictive performance of our proposed method with that of the baseline methods on both task of classification and regression. For classification, we focused on binary classification. Multi-class classification problems can be simply converted to multiple binary ones by conducting pair-wise binary classification or one versus rest binary classification.

5.1 Datasets and Experimental Setup

In our experiments, all the datasets that are used in the task of classification are from the UCI machine learning repository. All missing values were imputed with column mean. In the training stage, we use the scikit-learn package [23] to build the decision tree. It implements an optimized version of CART. For classification, samples from each of the two classes are equally divided into three folds. One is used for building the basic decision tree, one for pruning and one for testing. For a fair evaluation of different pruning methods, we carry out this random split ten times and report the average performance over these ten random splits. For our method, we learn the weights of the nodes in decision trees on the training set. Table 1 shows the data sets, their statis-

tics, respective task types and the average number of nodes in the decision trees built on training sets.

For regression, we simply divide all the samples equally into three folds, one for training the initial decision tree, one used as the independent pruning set, and the last fold used for testing. This random split is also carried out for ten times and the average performance is reported. In building the initial decision tree for pruning, we set the maximum depth of the tree to 15. We use the squared root of mean squared error (RMSE) as the metric for evaluation of performance of different pruning methods for regression. To reduce the effect of the scale of response, we center the response vector to zero and then divide it by its infinity norm such that the range of the response for all the samples is $[-1, 1]$.

In making predictions, we use weighted sum of rule functions of the tree nodes as the discriminant function. Note that this discriminant function can be written into an equivalent form of weighted sum of rule functions corresponding only to the leaf nodes. As is pointed out in [12], the rule function of an internal node can be decomposed into the sum of the rule functions of its two child nodes, i.e. $v_i(x) = v_j(x) + v_k(x)$ where node i is an internal node and j and k are its two child nodes. By applying this rule recursively over all the internal nodes of the pruned tree, the weights associated with all the internal nodes will eventually pass down onto leaf nodes. When it comes to making a prediction for a specific test sample, only one leaf node will actively participate in determining its label since the values of rule functions of all other leaf nodes at this sample are zero.

For comparison in terms of prediction performance of post-pruned decision trees, we use several well-known methods including Reduced Error Pruning (REP) [25], Pessimistic Error Pruning (PEP) [24], Cost Complexity Pruning (CCP)[6], and Minimum Error Pruning (MEP) as the baseline methods. We refer interested readers to [8, 21] for a comprehensive analysis of these methods.

In the task of regression, we replace the classification error in REP, PEP and CCP by squared difference between the predicted response and the ground-truth response. As for MEP, we omit it from our experiments on regression since the definition of expected error rate explicitly involves the number of classes.

5.2 Results The prediction performance in the task of classification is shown in Table 1. We can see from the table the prediction performance of our proposed method is better than the best performance of the baseline methods on most of the datasets. The average number of nodes in the decision tree before and after pruning by different methods are shown in Table 3 in the supplement. We can also see that REP generally performs worse than other methods and usually results in an over-pruned decision tree, which is consistent with the observation made in [8].

The prediction performance in the task of regression is shown in Table 2. We can see from the table that our proposed method also achieves the best performance on most of the datasets. In most cases, unlike the classification, REP did not yield an over-pruned tree in task of regression. Our method generally produce smaller decision trees with better prediction performance. From Table 3 in the supplement, it is not difficult to observe that PEP prunes the entire tree in most cases. This is not surprising since in this method, the decision regarding whether a sub-tree should be pruned or not depends on a standard error term, while the magnitude of this standard error term relies on the scale of the response, making this method inappropriate for the task of regression.

5.3 Stability Selection In our experiments, we found that the model that gives the best prediction performance on the independent pruning set does not usually yield the best performance on the testing set, even though the pruning set consists of $1/3$ of samples in every dataset. This problem becomes even more evident when the number of samples in the dataset is small. This motivates us to look for a method that produces a more stable performance on the testing set when using an independent pruning set to determine the structure of the tree. We proposed to use stability selection to solve this problem of model selection. Furthermore, we have proved that the model returned by this stability selection procedure still preserves the tree structure.

To demonstrate the effectiveness of stability selection, we use the dataset "housing" as an example on which our method did not achieve the best performance. Figure 3 shows that how RMSE changes on the pruning set and testing set, respectively, by using the different models produced by different λ values in our experiments and models given by stability selection based on two different random splits of data. We can see from this figure that the best model selected by the independent pruning set using cross validation does not perform well on the testing set. In contrast, the best model selected by the independent pruning set using stability selection is very close to the best model on the testing set.

6 Conclusion

In this paper, we proposed a novel optimization formulation for the decision tree post-pruning problem. With the use of the indicator functions to represent a tree

Table 1: Statistics of datasets used in the task of classification and performance of various methods. The classification performance is measured in terms of the Area Under Curve (AUC)

datasets	# features	# samples	REP(AUC)	PEP(AUC)	CCP(AUC)	MEP(AUC)	Maxheap(AUC)
sonar	60	208	0.7031	0.7167	0.7021	0.7124	**0.7233**
biodeg	41	1055	0.5212	0.7948	0.7616	0.7842	**0.8034**
bank note	4	1372	0.9109	0.9764	0.9769	0.9784	**0.9796**
Musk1	166	476	0.6663	0.7486	0.7378	0.7374	**0.7576**
Musk2	200	200	0.5	0.9389	0.9266	0.9364	**0.9468**
EEGEyeState	14	14980	0.6628	0.8491	0.8445	0.8488	**0.8513**
LSVT	313	126	0.6703	0.7071	0.7037	0.7050	**0.7154**
masses	5	961	0.8171	0.8390	0.8431	0.8699	**0.8843**
arcene	10000	200	0.6386	0.6484	0.6404	**0.6517**	0.6502
madelon	500	2600	0.7136	0.7693	0.767	0.7725	**0.7762**

Table 2: Statistics of datasets used in the task of regression and performance of various methods. The classification performance is measured in terms of the Root Mean Squared Error (RMSE)

datasets	# features	# samples	REP(RMSE)	PEP(RMSE)	CCP(RMSE)	Maxheap(RMSE)
housing	13	506	**0.1686**	0.3424	0.1699	0.1691
parkinsonupds	16	5875	0.4163	0.4437	0.4165	**0.4054**
CMB	16	11934	0.1196	0.3445	0.1208	**0.1061**
skillcraft	18	3395	0.2814	0.4003	0.2830	**0.2754**
redwine quality	11	1599	0.2701	0.3052	0.2658	**0.2586**
whitewine quality	11	4898	0.2463	0.2829	0.2448	**0.2388**
bank data	32	8192	0.1667	0.2403	0.1680	**0.1647**
family data	32	8192	0.1673	0.2408	0.1690	**0.1655**
cpusmall	12	8192	0.0445	0.2194	0.0455	**0.0430**

that was adopted in RuleFit, we are the first one to develop a systematic optimization formulation that can encapsulate the tree structure existing among the rules into a sparse learning framework by using the max-heap constraint as well as the sparsity constraint. This novel formulation leads to a non-convex optimization problem which can be addressed by an efficient max-heap projection algorithm that solves convex and smooth problems in each iteration. Since the selection of the regularization parameters has been a practical barrier for many sparse learning methods, an efficient stability selection method is further proposed for enabling robust model selection in practice. We conducted extensive experiments using a wide range of real-world datasets which demonstrated that the proposed method outperforms many existing benchmark pruning methods, leading to better prediction accuracy.

Acknowledgement

This work was supported in part by research grants from NIH (RF1AG051710) and NSF (IIS-0953662 and III-1421057).

References

[1] X. Amatriain. Mining large streams of user data for personalized recommendations. *ACM SIGKDD Explorations Newsletter*, 14(2):37–48, 2013.

[2] R. Appel, T. Fuchs, P. Dollár, and P. Perona. Quickly boosting decision trees-pruning underachieving features early. In *ICML*, pages 594–602, 2013.

[3] N. Asadi and J. Lin. Training efficient tree-based models for document ranking. In *Advances in Information Retrieval*, pages 146–157. Springer, 2013.

[4] A. Bifet, G. Holmes, B. Pfahringer, and E. Frank. Fast perceptron decision tree learning from evolving data streams. In *PAKDD*, pages 299–310, 2010.

[5] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001.

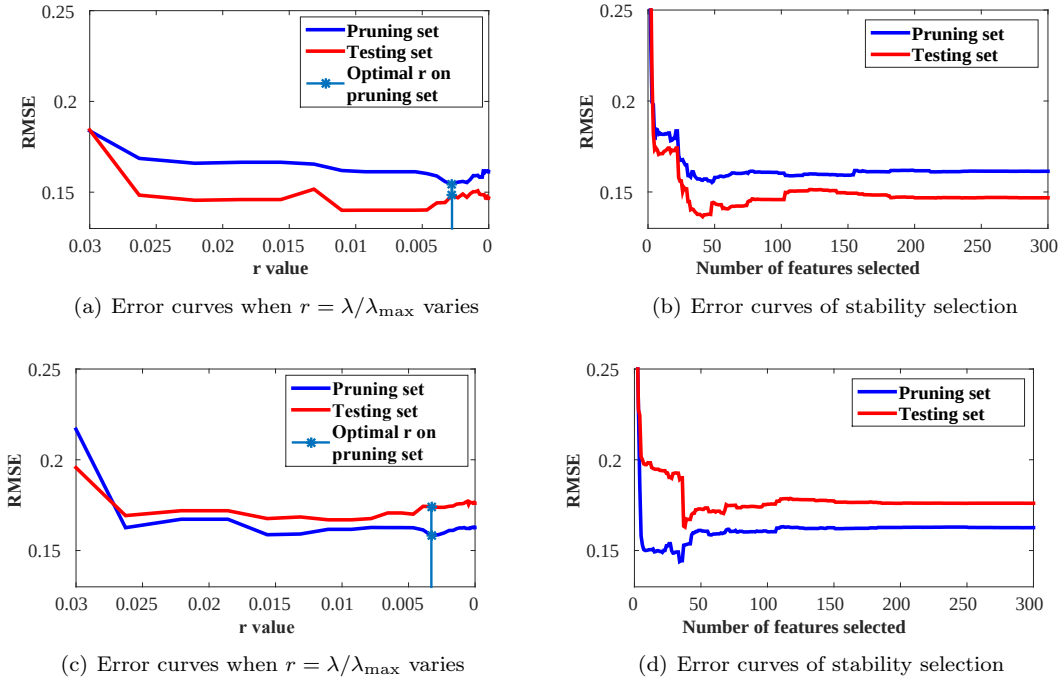
[6] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[7] X. P. Burgos-Artizzu, P. Dollár, D. Lin, D. J. Anderson, and P. Perona. Social behavior recognition in continuous video. In *CVPR*, pages 1322–1329, 2012.

[8] F. Esposito, D. Malerba, G. Semeraro, and J. Kay. A comparative analysis of methods for pruning decision trees. *PAMI, IEEE Transactions on*, 19(5):476–491, 1997.

[9] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[10] J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *Ann. Appl. Stat.*, 2(3):916–954, 09



(a) Error curves when $r = \lambda/\lambda_{\max}$ varies

(b) Error curves of stability selection

(c) Error curves when $r = \lambda/\lambda_{\max}$ varies

(d) Error curves of stability selection

Figure 3: (a) and (c) show the Root Mean Square Error (RMSE) curve when $r = \lambda/\lambda_{\max}$ varies on two data sets respectively. (b) and (d) show the RMSE error curves of stability selection when the number of selected features varies on two data sets respectively. (a) and (c) are generated using one random split of the dataset "housing". (c) and (d) are generated using another random split of the same dataset. It can be seen that 1) the performance of stability selection is quite stable; (2) the models selected by stability selection perform well on the testing set, which is comparable or even better than the performance of the models selected by cross validation.

2008.

[11] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. In *ICML*, pages 37–45, 2013.

[12] R. Johnson and T. Zhang. Learning nonlinear functions using regularized greedy forest. *PAMI, IEEE Transactions on*, 36(5):942–954, 2014.

[13] G. V. Kass. An exploratory technique for investigating large quantities of categorical data. *Applied statistics*, pages 119–127, 1980.

[14] C.-C. Lee, E. Mower, C. Busso, S. Lee, and S. Narayanan. Emotion recognition using a hierarchical binary decision tree approach. *Speech Communication*, 53(9):1162–1171, 2011.

[15] M. Lichman. UCI machine learning repository, 2013.

[16] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*. Arizona State University, 2009.

[17] J. Liu, L. Sun, and J. Ye. Projection onto a nonnegative max-heap. In *NIPS*, pages 487–495, 2011.

[18] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool. Face detection without bells and whistles. In *ECCV*, pages 720–735. Springer, 2014.

[19] M. Mehta, J. Rissanen, and R. Agrawal. Mdl-based decision tree pruning. In *KDD*, pages 216–221, 1995.

[20] N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

[21] J. Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine learning*, 4(2):227–243, 1989.

[22] T. Niblett and I. Bratko. Learning decision rules in noisy domains. In *Proceedings of Expert Systems' 86, The 6Th Annual Technical Conference on Research and development in expert systems III*, pages 25–34. Cambridge University Press, 1987.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[25] J. R. Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[26] R. Rastogi and K. Shim. Public: a decision tree classifier that integrates building and pruning. In *VLDB*, volume 98, pages 24–27, 1998.

[27] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. 1990.

[28] Z. Xu, G. Huang, K. Q. Weinberger, and A. X. Zheng. Gradient boosted feature selection. In *KDD*, pages 522–531, 2014.