Abstract

In this paper we study the problem of pruning a binary tree by minimizing, over all pruned subtrees of the given tree, an objective function that combines an additive cost term with a penalty term that depends only on tree size. We present algorithms for general size-based penalties, although our focus is on sub-additive penalties (roughly, penalties that grow more slowly than linear penalties with increasing tree size). Such penalties are motivated by recent results in statistical learning theory for decision trees, but may have wider application as well. We show that the family of pruned subtrees induced by a sub-additive penalty is a subset of the family induced by an additive penalty. This implies (by known results about additive penalties) that the family induced by a sub-additive penalty (1) is nested; (2) is unique; and (3) can be computed efficiently. It also implies that, when a single tree is to be selected by cross-validation from the family of prunings, sub-additive penalties will never present a richer set of options than an additive penalty.

1 Introduction

Tree-based methods are one of the most widely applied techniques in all of applied mathematics and engineering, from nonparametric statistics and machine learning to source coding and multiscale signal and image processing. In this paper we focus on pruning trees via complexity regularization, a task that often occurs in the design of tree-based methods. Rather than focusing on a specific application or pruning problem, we present general results that apply in a number of different settings.

The formal statement of our problem is as follows. In graph theory a tree is a connected graph without cycles. We consider a specific kind of tree that we call a rooted binary tree which has the following properties:

- there exists a unique node with degree 2
- all other nodes have degree 1 or 3.
The degree of a node is the number of edges linking that node to other nodes (called neighbors). The node with degree 2 is called the root node. Nodes with degree greater than 1 are called internal nodes and nodes with degree 1 are called terminal or leaf nodes. The depth of a node $t$ is the number of edges traversed along the path between $t$ and the root node. Every node $t$ except the root has a parent which is the unique neighbor of $t$ whose depth is one less than $t$'s. Every internal node $t$ has two children which are the two neighbors of $t$ having depth one more than $t$'s. Rooted binary trees are readily envisioned by picturing the root node at the top of the graph and the remaining nodes “dangling” down in such a way that parents are above their children and one child always branches left while the other branches right.

The set of leaf nodes of $T$ is denoted $L(T)$. The size of a tree $T$ is the number of leaf nodes and denoted $|T|$. A subtree of $T$ is a subgraph $S \subseteq T$ that is a rooted binary tree in its own right. If $S$ is a subtree that contains the root of $T$ we say $S$ is a pruned subtree of $T$ and write $S \preceq T$.

For the remainder of the paper let $T$ be a fixed rooted binary tree. Let $\rho$ be a functional mapping subtrees of $T$ to the positive reals. Let $\Phi$ be a mapping from the positive integers to the positive reals. We make the following assumptions on $\rho$ and $\Phi$.

- $\rho$ is monotonically nonincreasing, that is, $S_1 \preceq S_2 \Rightarrow \rho(S_1) \geq \rho(S_2)$
- $\rho$ is additive, that is,
  $$\rho(S) = \sum_{t \in L(S)} \rho(t)$$
- $\Phi$ is monotonically increasing, that is, $k_1 < k_2 \Rightarrow \Phi(k_1) < \Phi(k_2)$.

We are interested in algorithms computing and theorems describing two kinds of pruning problems. The first is

$$T^* = \arg \min_{S \preceq T} \rho(S) + \Phi(|S|).$$

If multiple trees achieve the minimum, choose $T^*$ to be one with smallest size. Note that $T^*$ is still not necessarily unique. The problem of solving (1) is called single pruning, in contrast with family pruning described below.

We refer to $\rho(S)$ and $\Phi(|S|)$ as the cost and penalty of $S$, respectively. Conceptually, every $S \preceq T$ is a model that explains some observed phenomenon. Typically $S = T$ is the most complicated model while the root node is the simplest. The idea behind pruning is to find a model that appropriately balances the complexity of $S$ with the fidelity of $S$ to an observed phenomenon.

One of the earliest and perhaps most widely known examples of this kind of pruning problem comes from the method of Classification and Regression Trees (CART) of Breiman, Friedman, Olshen and Stone [1]. In CART, a training dataset $(X_i, Y_i)_{i=1}^n$ is given, where the $X_i$ are feature vectors and the response variables $Y_i$ satisfy $Y_i \in \{1, \ldots, M\}$ for classification and $Y_i \in \mathbb{R}$ for regression. The training data is used to construct an initial tree $T$ that “overfits” the training data (for example, classifying every training sample correctly), and the purpose of pruning is to select a tree $S \preceq T$ that generalizes to accurately predict the correct $Y$ for unlabeled $X$ observed in the future.

For classification trees, each node $t \in T$ is assigned a class label $y_t$ by majority vote over the training samples reaching $t$, and $\rho(S)$ is taken to be the empirical error

$$\rho(S) = \frac{1}{n} \sum_{t \in L(S)} \sum_{i: X_i \in t} \mathbb{I}_{\{Y_i \neq y_t\}}.$$
Here \( I \) denotes the indicator function. For regression trees each \( t \in T \) is assigned the empirical average (or perhaps some more general function of the samples in \( t \))

\[
y_t = \frac{1}{|\{i : X_i \in t\}|} \sum_{i : X_i \in t} Y_i
\]

and \( \rho(S) \) is the average empirical squared error

\[
\rho(S) = \frac{1}{n} \sum_{t \in L(S)} \sum_{i : X_i \in t} (Y_i - y_t)^2.
\]

For a penalty CART uses \( \Phi(|S|) = \lambda |S| \) where \( \lambda > 0 \) is some constant. Additional examples of costs and penalties for tree structured source coding may be found in Chou, Lookabaugh, and Gray [2].

In many applications it is not known precisely how to calibrate \( \rho \) with respect to \( \Phi \) so as to achieve an optimal pruning. In such cases it is customary to introduce a tuning parameter \( \alpha \), solve

\[
T^*(\alpha) = \arg \min_{S \subseteq T} \rho(S) + \alpha \Phi(|S|).
\]

for several different values of \( \alpha \), and choose the best \( \alpha \) by cross-validation. This is the second pruning problem we consider in this paper.

Since \( T \) is in general finite it follows that there exist constants \( 0 = \alpha_0 < \alpha_1 < \ldots < \alpha_m = \infty \) and pruned subtrees \( R_1, \ldots, R_m \) such that

\[
\alpha \in [\alpha_{\ell-1}, \alpha_\ell) \Rightarrow T(\alpha) = R_\ell.
\]

Because \( \Phi \) is increasing it follows that \( |R_1| > \cdots > |R_m| = 1 \), although these tree are not nested in general (see Section 2 for further discussion). We refer to \( R_1, \ldots, R_m \) as the family of prunings of \( T \) with respect to \( \rho \) and \( \Phi \). The problem of computing these subtrees and thresholds is called family pruning. Note that in single pruning no generality is gained by introducing a scalar multiplier \( \alpha \) of \( \Phi \), for such a scalar may simply be absorbed into \( \Phi \).

### 1.1 Motivation

Single and family pruning have been studied extensively in the case where \( \Phi \) is additive, by which we mean \( \Phi(|S|) = \lambda |S| \) for some \( \lambda > 0 \) (for family pruning it suffices to take \( \lambda = 1 \)). Additive penalties are by far the most popular choice for \( \Phi \), owing in large part to the existence of computationally efficient algorithms (which we review) for computing \( T^* \) and the family of prunings of \( T \). Moreover, the family of prunings satisfies the desirable properties that the trees \( R_\ell \) are unique and nested. In many cases, however, the choice of an additive penalty appears to have no other grounding besides computational convenience.

Roughly speaking, sub-additive penalties are penalties that grow more slowly than additive penalties as a function of tree size (a precise definition is given in Section 4). For example, \( \Phi(k) \propto k^\tau \) for \( 0 < \tau \leq 1 \) defines a sub-additive penalty. Several theoretical results, many of them recent, suggest that sub-additive penalties may be more appropriate than additive penalties for certain applications. Barron [3] demonstrates risk bounds that, when applied to classification or regression trees, imply a penalty of \( \Phi(k) \propto \sqrt{k} \). Mansour and McAllester [4], Nobel [5], and Scott and Nowak [6] also derive risk bounds for classification trees with \( \Phi(k) \propto \sqrt{k} \). Mansour and McAllester [4] and Langford [7] derive penalties for classification trees that vary between \( \Phi(k) \propto k \) and \( \Phi(k) \propto \sqrt{k} \).
Meanwhile, classification risk bounds for additive penalties are only known for the special “zero error” case (when the optimal classifier is correct with probability one) and under the more general but still quite restrictive “identifiability” assumption of Blanchard, Schäfer, and Rozenholc [8]. In summary, sub-additive penalties appear to have a much stronger theoretical foundation than additive ones in certain settings, especially classification.

Additive penalties naturally arise in estimation problems that employ a squared-error or $L^2$ loss function. For example, in regression the squared bias grows linearly with degrees of freedom in a linear model. In classification, however, the probability of error can be expressed by an $L^1$ loss function. It thus seems reasonable to expect that non-additive penalties would be more appropriate from a theoretical point of view for applications that use a non-$L^2$ loss function, such as source coding with an $L^1$ distortion.

1.2 Overview

The purpose of this paper is to present algorithms and relevant properties for single and family pruning with non-additive, and in particular sub-additive penalties. One of our main results is that the family of prunings generated by a sub-additive penalty is a subset of the family of prunings generated by the additive penalty. Positive implications of this fact are that sub-additive families are nested and unique. It also leads to a simple algorithm for generating the family. A negative implication, however, is that when a tree is to be selected from the family of prunings by cross-validation (a very common strategy [1]), sub-additive penalties never provide a richer class of options than the additive penalty.

The paper is organized as follows. In Section 2 we study pruning with general size-based penalties. We give explicit algorithms for single pruning and family pruning and provide a geometric framework for interpreting the family of prunings. This section brings together several known results and perspectives, adds a few new insights, and sets the stage for our later discussion of sub-additive penalties. In Section 3 we review algorithms and properties related to pruning with additive penalties. In Section 4 we define dominating and sub-additive penalties and prove a general theorem about nested families of prunings. We also explore in more detail the implications of this theorem as outlined above. Section 5 contains discussion and conclusions.

2 General Sized-Based Penalties

We first present a general algorithm for single pruning when $\Phi(k)$ is arbitrary. This first algorithm applies even if $\Phi$ is not necessarily increasing. The algorithm should not be considered novel; its key components have appeared previously in other guises as discussed below.

For each $k = 1, 2, \ldots, |T|$, define $T^k$ to be a pruned subtree $S \preceq T$ (there may be more than one) minimizing $\rho(S)$ subject to $|S| = k$. These trees are referred to as minimum cost trees. Observe that

$$T^* = \arg\min \{ \rho(T^k) + \Phi(k) : k = 1, 2, \ldots, |T| \}$$

(3)

is a solution to (1). In other words, it suffices to construct the sequence $T^k$ and minimize the objective function over this collection. For the remainder of the paper, fix choices of $T^k$ whenever $T^k$ is not unique.
2.1 Computing Minimum Cost Trees

Since $T$ is binary it has $|T| - 1$ internal nodes. Let the nodes of $T$ be indexed $1$ through $2|T| - 1$ in such a way that children have a larger index than their parents. (We refer to nodes and their indices interchangeably.) Let $T_t$ denote the subtree rooted at node $t$ and containing all of $t$’s descendants in $T$ (thus $T = T_1$). Let $l(t)$ and $r(t)$ denote the left and right children of node $t$, respectively. If $U$ and $V$ are pruned subtrees of $T_{l(t)}$ and $T_{r(t)}$, let $\text{merge}(t, U, V)$ denote the pruned subtree of $T_t$ having $U$ and $V$ as its left and right subtrees, respectively. Finally, let $T^k_t$ denote the pruned subtree of $T_t$ having minimum cost among all pruned subtrees of $T_t$ with $k$ leaf nodes.

The algorithm for computing minimum cost trees is based on the following fact: If we know $T^i_{l(t)}$ and $T^j_{r(t)}$ for $i = 1, 2, \ldots, |T_{l(t)}|$ and $j = 1, 2, \ldots, |T_{r(t)}|$, it is a simple matter to find $T^k_t$, $k = 1, 2, \ldots, |T_t|$. For each $k = 1, 2, \ldots, |T_t|$, there exist $i, j$ with $i + j = k$ such that $T^k_t = \text{merge}(t, T^i_{l(t)}, T^j_{r(t)})$. This follows from additivity of $\rho$. Moreover, if $T^k_t = \text{merge}(t, T^i_{l(t)}, T^j_{r(t)})$, then $\rho(T^k_t) = \rho(T^i_{l(t)}) + \rho(T^j_{r(t)})$. We may then set $T^k_t = \text{merge}(t, T^i_{l(t)}, T^j_{r(t)})$, where $i^*, j^*$ minimize $\rho(T^i_{l(t)}) + \rho(T^j_{r(t)})$ over all $i, j$ such that $i + j = k$, $1 \leq i \leq |T_{l(t)}|$, and $1 \leq j \leq |T_{r(t)}|$. Note that $i^*, j^*$ are determined by exhaustive search.

This step may be applied at each level of $T$, working from the bottom up, and leads to an algorithm for computing the minimum cost trees, and hence for determining $T^*$. The complete algorithm is presented in Fig. 1. The computational complexity of computing the minimum cost trees is $O(|T|^2)$. This was proved by Bohanec and Bratko [9]. The algorithm takes longer to run when $T$ is more balanced. If $T$ is maximally lopsided, e.g., all right children are terminal nodes, the algorithm computes the minimum cost trees in $O(|T|)$ operations.

The procedure described above for determining minimum cost trees is essentially the dual of the algorithm described in [9]. They considered the problem of finding the pruned subtree with smallest size among all pruned subtrees with empirical error below a certain threshold. This procedure was apparently known to the CART authors. As reported in [9], “Leo Breiman (Private Communication, December 1990) did implement such an algorithm for optimal pruning; he was satisfied that it worked, but no further development was done, and the algorithm was not published.” As far as we know, the present work is the first to point out the use of minimum cost trees for single pruning with general size-based penalties.

2.2 Geometric aspects of family pruning

To each $S \subseteq T$ associate the function $f_S : [0, \infty) \to \mathbb{R}$ defined by $f_S(\alpha) = \rho(S) + \alpha \Phi(|S|)$. In this way each pruned subtree $S$ maps to a line with $y$-intercept $\rho(S)$ and slope $\Phi(S)$, as shown in Fig. 2. Define

$$f^*(\alpha) = \min_{S \subseteq T} f_S(\alpha).$$

Clearly $f^*$ has the form

$$f^*(\alpha) = f_{R_{\ell}}(\alpha), \quad \alpha \in [\alpha_{\ell-1}, \alpha_{\ell})$$

for some constants $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_m = \infty$ and subtrees $R_{\ell} \subseteq T, \ell = 1, 2, \ldots, m$. Moreover, since $\Phi$ is monotonically increasing, $\Phi(R_{\ell-1}) > \Phi(R_{\ell})$ implies $|R_{\ell-1}| > |R_{\ell}|$. These observations are summarized as follows.

**Proposition 1.** If $\Phi(k)$ is monotonically increasing in $k$, then there exist constants $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_m = \infty$ and pruned subtrees $R_{\ell} \subseteq T, \ell = 1, 2, \ldots, m$, with $|R_1| > \cdots > |R_m| = 1$, such that $T(\alpha) = R_{\ell}$ whenever $\alpha \in [\alpha_{\ell-1}, \alpha_{\ell})$. 
Input:
Initial tree \( T \)

Main Loop:
For \( t = 2|T| - 1 \) downto 1
Set \( T_1^t = \{ t \} \);
If \( t \) is not a terminal node, Then
For \( k = 2 \) to \( |T_t| \)
Set \( \text{mincost} = \infty \);
For \( i = \max(1, k - |T_{r(t)}|) \) to \( \min(|T_{l(t)}|, k - 1) \)
Set \( j = k - i \);
Set \( \text{cost} = \rho(T_i^t) + \rho(T_j^t) \);
If \( \text{cost} < \text{mincost} \), Then
Set \( \text{mincost} = \text{cost} \);
Set \( T_k^t = \text{merge}(t, T_i^t, T_j^t) \);
End If
End For
End If
End For

Output:
The minimum cost trees \( T_k^t = T_1^t, k = 1, 2, \ldots, |T| \)

Figure 1: An algorithm for computing minimum cost trees. The limits for the innermost “For” loop ensure that \( i, j \) satisfy \( i + j = k, 1 \leq i \leq |T_{l(t)}|, \) and \( 1 \leq j \leq |T_{r(t)}| \).
Figure 2: Hypothetical plots of $f_S(\alpha) = \rho(S) + \alpha \Phi(|S|)$ as a function of $\alpha$ for all $S \preceq T$. Pruned subtrees coinciding with the minimum of these functions (shown in bold) over a range of $\alpha$ minimize the pruning criterion for those $\alpha$. 

\[ S=R_1 \]
\[ S=R_2 \]
\[ S=R_3 \]
\[ S=R_4 \]
This picture also provides us with an algorithm for determining the \( \alpha_\ell \) and \( R_\ell \). Observe that each \( R_\ell \) must be a minimum cost tree \( T^k \). Therefore

\[
f^*(\alpha) = \min_k f_{T^k}(\alpha).
\]

Clearly \( R_1 = T^{k_1} \) where \( k_1 \) is the smallest \( k \) such that \( \rho(T^k) = \rho(T) \). Now observe that, assuming \( i < j \), \( f_{T_i} \) and \( f_{T_j} \) intersect at the point

\[
\gamma_{i,j} = \frac{\rho(T^i) - \rho(T^j)}{\Phi(j) - \Phi(i)}.
\]

Therefore, if \( R_\ell = T^{k_\ell} \), then \( R_{\ell+1} = T^{k_{\ell+1}} \), where

\[
k_{\ell+1} = \arg \min_{k < k_\ell} \gamma_{k,k_\ell}.
\]

If multiple \( k \) minimize the right-hand side, let \( k_{\ell+1} \) be the smallest. Furthermore, we have

\[
\alpha_\ell = \gamma_{k_{\ell+1},k_\ell}.
\]

The algorithm is summarized in Fig. 3. This is the first algorithm of which we are aware for family pruning with a general size-based penalty.

We also highlight a property inherent in the definition of \( k_\ell \) that will be of use later.

**Lemma 1.** If \( k < k_\ell \), then \( \gamma_{k_{\ell+1},k_\ell} \leq \gamma_{k,k_\ell} \). If \( k < k_{\ell+1} \), then \( \gamma_{k_{\ell+1},k_\ell} < \gamma_{k,k_\ell} \).

A second geometric picture due to Chou et al. [2], who consider only additive or affine penalties, offers essentially equivalent insights into the family of prunings of \( T \). Consider the set of points \( P = \{ p(S) = (\rho(S), \Phi(|S|)) \mid S \preceq T \} \subset \mathbb{R}^2 \), as depicted in Fig. 4. The point corresponding to \( R_m \) (= the root of \( T \)) is furthest down and to the right. The point corresponding to \( R_1 \) is furthest up and to the left (assuming \( R_1 = T \)). Moreover, the points corresponding to \( R_\ell, \ell = 1, \ldots, m \), are the vertices of the lower boundary of the convex hull of \( P \), listed counterclockwise. Thus, \( \alpha_\ell \) is the negative of the slope of the line segment connecting \( p(R_\ell) \) to \( p(R_{\ell+1}) \). The algorithm described above for generating \( \alpha_\ell \) and \( R_\ell \) can now be rederived in this setting by starting with \( R_1 \) and successively learning faces of the lower boundary of the convex hull of \( P \) in a counterclockwise fashion.

### 3 Additive Penalties

When \( \Phi(|T|) = \lambda |T| \) for some \( \lambda > 0 \), there exist faster algorithms for single and family pruning than those described in the previous section. Moreover, the optimally pruned trees satisfy certain nice properties. The material in this section is taken from [1, chap. 10].

When \( \lambda \) is known, \( T^* \) may be computed by a simple bottom-up procedure. In particular, denoting

\[
T^*_t = \arg \min_{S \preceq T_t} \rho(S) + \lambda |S|,
\]

we have \( T^*_t = \{ t \} \) for leaf nodes and for internal nodes

\[
T^*_t = \arg \min \{ \rho(S) + \lambda |S| : S = \{ t \} \text{ or } S = \text{MERGE}(t, T^*_t(t), T^*_r(t)) \}.
\]
Input:
Minimum cost trees $T^k$, $k = 1, \ldots, |T|$

Initialization:
Set $k_1 = \arg\min\{k : \rho(T^k) = \rho(T)\}$;
Set $R_1 = T^{k_1}$;
Set $\ell = 1$;

Main Loop:
While $k_\ell > 1$
   Set $\alpha_\ell = \infty$;
   For $k = k_\ell - 1$ downto 1
      Set $\gamma_{k,k_\ell} = (\rho(T^k) - \rho(T^{k_\ell}))/ (\Phi(k_\ell) - \Phi(k))$;
      If $\gamma_{k,k_\ell} \leq \alpha_\ell$
         Set $\alpha_\ell = \gamma_{k,k_\ell}$;
         Set $k_{\ell+1} = k$;
      End If
   End For
   Set $\ell = \ell + 1$;
   Set $R_\ell = T^{k_\ell}$;
End While

Output:
The family of prunings $R_\ell$ and thresholds $\alpha_\ell$.

Figure 3: An algorithm generating the family of prunings and associated thresholds for an arbitrary increasing penalty.
Figure 4: Hypothetical plot of points \((\rho(S), \Phi(S))\) for all \(S \preceq T\). The family of prunings corresponds to the vertices of the lower boundary of the convex hull, and the (negative) slopes between vertices correspond to the thresholds \(\alpha_\ell\).
This last fact follows easily by additivity of $\rho$ and $\Phi$ and by induction on $t$. This leads to an $O(|T|)$ algorithm for computing $T^*$, much faster than the more general $O(|T|^2)$ algorithm described previously. Moreover, $T^*$ can be shown to be unique [1].

Breiman et al. [1] also prove the following theorem about the family of prunings generated by an additive penalty.

**Theorem 1 (Breiman, Friedman, Olshen and Stone).** If $\Phi(k) = k$, then there exist weights $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_m = \infty$ and subtrees $T \gg R_1 \gg \cdots \gg R_m = \{\text{root}\}$ such that $T(\alpha) = R_\ell$ whenever $\alpha \in [\alpha_{\ell-1}, \alpha_\ell)$.

In particular, the result is an improvement over Proposition 1 because the family of prunings is nested. We refer to the family $\{R_\ell\}$ in Theorem 1 as the CART trees.

The nesting property leads to an algorithm for finding these weights and subtrees [1, 2]. The algorithm has a worst case running time of $O(|T|^2)$, the worst case being when $T$ is unbalanced. However, when $T$ is balanced (i.e., when the depth of $T$ is proportional to $\log |T|$), then the algorithm runs in time $O(|T| \log |T|)$. This yields an improvement (relative to the algorithm in Fig. 3) for many problems, such as those in signal and image processing, where the initial tree is often balanced.

The proof of Theorem 1 given by [1] is algebraic; an alternative geometric proof is given by [2]. These authors also extend the theorem to affine costs and penalties. A separate algebraic account may be found in [10].

### 4 Sub-additive penalties

In this, the main section of the paper, we introduce sub-additive penalties and show that for such penalties, the family of prunings is a subset of the CART trees. Thus, these trees are also unique, nested, and may be computed using the CART trees.

**Definition 1.** Let $\Phi^1$ and $\Phi^2$ be two increasing penalties. We say $\Phi^1$ dominates $\Phi^2$, denoted $\Phi^1 \gg \Phi^2$, if, for all positive integers $a > b > c$, we have

$$\frac{\Phi^2(a) - \Phi^2(b)}{\Phi^2(a) - \Phi^2(c)} \leq \frac{\Phi^1(a) - \Phi^1(b)}{\Phi^1(a) - \Phi^1(c)}, \quad (4)$$

If $\Phi^1(k) = k$ and $\Phi^1 \gg \Phi^2$, we say $\Phi^2$ is sub-additive.

An important example of a sub-additive penalty is the square root penalty $\Phi^2(k) = \sqrt{k}$. To see that this is indeed sub-additive, observe that for $a > b > c$,

$$\frac{\Phi^2(a) - \Phi^2(b)}{\Phi^2(a) - \Phi^2(c)} = \frac{\sqrt{a} - \sqrt{b}}{\sqrt{a} - \sqrt{c}} < \frac{\sqrt{a} - \sqrt{b}}{\sqrt{a} - \sqrt{c}} \cdot \frac{\sqrt{a} + \sqrt{b}}{\sqrt{a} + \sqrt{c}} = \frac{a - b}{a - c} = \frac{\Phi^1(a) - \Phi^1(b)}{\Phi^1(a) - \Phi^1(c)}$$

where $\Phi^1(k) = k$.

More generally, the following result characterizes a large class of penalties with $\Phi^1 \gg \Phi^2$. 

Proposition 2. Let $f, g$ be real valued, twice differentiable, increasing functions on $(0, \infty)$, and for $k = 1, 2, \ldots$, set $\Phi^1(k) = f(k)$ and $\Phi^2(k) = g(k)$. Let $x_0$ be a positive real number. If $0 < g'(x) \leq f'(x)$ and $g''(x) \leq f''(x)|g'(x)/f'(x)|$ for all $x \geq x_0$, then the inequality (4) is satisfied for all real numbers $a > b > c \geq x_0$. Therefore, if $x_0 \leq 1$, then $\Phi^1 \gg \Phi^2$.

Proof. The proof has two main steps: First, we prove the lemma for the special case $f(x) = x$, and then we use the first step to establish the general case by reparametrizing $\mathbb{R}$.

Assume for now $f(x) = x$. Then, by assumption, $g'(x) \leq 1$ and $g''(x) \leq 0$ for $x \geq x_0$. Let $a > b > c \geq x_0$. Note that inequality (4) is equivalent to

$$\frac{\Phi^2(a) - \Phi^2(b)}{a - b} \leq g'(b) \leq \frac{\Phi^2(b) - \Phi^2(c)}{b - c},$$

which can be seen by writing $\Phi^\kappa(a) = \Phi^\kappa(a) - \Phi^\kappa(b) + (\Phi^\kappa(b) - \Phi^\kappa(c))$, for $\kappa = 1, 2$, and simplifying. Also note that $\Phi^1$ and $\Phi^2$ are monotonically increasing from the assumption on the first derivative. By the fundamental theorem of calculus, $\Phi^2(a) - \Phi^2(b) = \int_b^a g'(x)dx \leq g'(b)(a - b)$, where we use the concavity of $g$ in the last step. Similarly, $\Phi^2(b) - \Phi^2(c) = \int_c^b g'(x) \geq g'(b)(b - c)$. Summarizing, we have shown

$$\frac{\Phi^2(a) - \Phi^2(b)}{a - b} \leq g'(b) \leq \frac{\Phi^2(b) - \Phi^2(c)}{b - c},$$

which by (5) implies the theorem for this special case.

Now consider the general case. Define $\tilde{f}(x) = x$ and $\tilde{g}(x) = g(f^{-1}(x))$. Now $\tilde{f}'(x) = 1$, while $\tilde{g}'(x) = g'(f^{-1}(x))/f'(f^{-1}(x))$, which is $\leq 1$ provided $x \geq \tilde{x}_0 := f(x_0)$. In addition, $\tilde{f}''(x) = 0$ and

$$\tilde{g}''(x) = \frac{g''(f^{-1}(x)) - f''(f^{-1}(x))|g'(f^{-1}(x))/f'(f^{-1}(x))|}{(f'(f^{-1}(x)))^2},$$

which is $\leq 0$ if $x \geq \tilde{x}_0$. Thus, we may apply the previous case to $\tilde{f}$ and $\tilde{g}$. For all real numbers $x > y > z \geq f(x_0)$, we have

$$\frac{g(f^{-1}(x)) - g(f^{-1}(y))}{g(f^{-1}(y)) - g(f^{-1}(z))} \leq \frac{x - y}{y - z}.$$

By taking $x = f(a), y = f(b)$, and $z = f(c)$, and by monotonicity of $f$, we conclude

$$\frac{g(a) - g(b)}{g(b) - g(c)} \leq \frac{f(a) - f(b)}{f(b) - f(c)},$$

which is what we wanted to show. \qed

The following corollary gives a concrete example of a family of penalties to which Proposition 2 applies.

Corollary 1. Let $\sigma, \tau > 0$, and set $\Phi^1(k) = k^\sigma$ and $\Phi^2(k) = k^\tau$. If $\sigma \geq \tau$, then $\Phi^1 \gg \Phi^2$. Hence, if $0 < \tau \leq 1$, then $\Phi^2$ is sub-additive.

Proof. Define $f(x) = x^\sigma$ and $g(x) = x^\tau$. For $x \geq 1 = x_0$

$$g'(x) = \tau x^{\tau - 1} \leq \sigma x^\sigma - 1 = f'(x).$$
Furthermore, for \( x \geq 1 \),
\[
    g''(x) = \tau(\tau - 1)x^{\tau - 2} \\
    \leq \tau(\sigma - 1)x^{\tau - 2} \\
    = \left( \sigma(\sigma - 1)x^{\sigma - 2} - \frac{\tau x^{\tau - 1}}{\sigma x^{\sigma - 1}} \right) \\
    = f''(x) \left( \frac{g'(x)}{f'(x)} \right).
\]

Now apply Proposition 2.

Further examples of dominating and sub-additive penalties may be derived from the following result.

**Proposition 3.** If \( f \), \( g \), and \( x_0 \) satisfy the hypothesis of Proposition 2, then so do \( \tilde{f}(x) = f(h(x)) \), \( \tilde{g}(x) = g(h(x)) \), and \( \tilde{x}_0 = h^{-1}(x_0) \) where \( h : (0, \infty) \to (0, \infty) \) is any twice differentiable function such that \( h'(x) > 0 \) for all \( x > 0 \).

**Proof.** Observe that for any \( x \geq \tilde{x}_0 \)
\[
    0 < \tilde{g}'(x) = g'(h(x)) \cdot h'(x) \leq f'(h(x)) \cdot h'(x) = \tilde{f}'(x)
\]
and
\[
    \tilde{g}''(x) = g''(h(x)) \cdot (h'(x))^2 + g'(h(x)) \cdot h''(x) \\
    \leq f''(x) \left( \frac{\tilde{g}'(x)}{\tilde{f}'(x)} \right) \cdot (h'(x))^2 + f'(h(x)) \left( \frac{\tilde{g}'(x)}{\tilde{f}'(x)} \right) \cdot h''(x) \\
    = \left( f''(h(x)) \cdot (h'(x))^2 + f'(h(x)) \cdot h''(x) \right) \left( \frac{\tilde{g}'(x)}{\tilde{f}'(x)} \right) \\
    = \tilde{f}''(x) \left( \frac{\tilde{g}'(x)}{\tilde{f}'(x)} \right).
\]

4.1 Main result

For \( \kappa = 1, 2, \) and \( \alpha \in \mathbb{R} \), define
\[
    T^\kappa(\alpha) = \arg \min_{S \in T} \rho(S) + \alpha \Phi^\kappa(|S|).
\]

By Proposition 1, there exist scalars \( \alpha_0 < \alpha_1 < \ldots < \alpha_m \), and \( \beta_0 < \beta_1 < \ldots < \beta_n \), and subtrees \( U_1, \ldots, U_m \), and \( V_1, \ldots, V_n \), such that

- \( \alpha \in [\alpha_{\ell - 1}, \alpha_\ell) \Rightarrow T^1(\alpha) = U_\ell \)
- \( |U_1| > \cdots > |U_m| = 1 \)
- \( \beta \in [\beta_{\ell - 1}, \beta_\ell) \Rightarrow T^2(\beta) = V_\ell \)
- \( |V_1| > \cdots > |V_n| = 1 \).
Theorem 2. With the notation defined above, if \( \Phi^1 \) and \( \Phi^2 \) are two increasing penalties such that \( \Phi^1 \gg \Phi^2 \), then \( \{V_1, \ldots, V_n\} \subseteq \{U_1, \ldots, U_m\} \). In other words, for each \( \beta \), there exists \( \alpha \) such that \( T^2(\beta) = T^1(\alpha) \).

An immediate application of the theorem is an alternate algorithm for pruning using a sub-additive penalty. Let \( \Phi^1(k) = k \) and let \( R_1, \ldots, R_m \) denote the CART trees. These may be computed efficiently by the algorithm of [1] or [2]. By Theorem 2, if \( \Phi^2 \) is sub-additive, then \( T^2(\beta) \) is one of these \( R_\ell \). Therefore,

\[
T^2(\beta) = \arg \min_{S \in \{R_1, \ldots, R_m\}} \rho(S) + \beta \Phi^2(|S|).
\]

This last minimization may be solved by exhaustive search over the \( m \leq |T| \) CART trees. We are unaware of a more direct way to do single pruning for sub-additive penalties.

The theorem also implies a new algorithm for family pruning when \( \Phi^2 \) is sub-additive. The procedure is exactly like the one described in Fig. 3, except that one only needs to consider \( k \) (see line 3 of the main loop) such that

\[
k \in \{i : i = |R_j| \text{ for some } j > \ell \}.
\]

Thus it is not necessary to compute all minimum cost trees, only the CART trees, which can often be done more efficiently.

We have two distinct algorithms for computing the family of prunings induced by a sub-additive penalty. Both algorithms have worst case running time \( O(|T|^2) \). The first algorithm, discussed in Section 2, is slower when \( T \) is more balanced, but prunes totally lop-sided trees in \( O(|T|) \) time. The second algorithm, just discussed, is slower when \( T \) is unbalanced, and runs in \( O(|T| \log |T|) \) time when \( T \) is balanced. Conceivably, one could devise a test that determines how balanced a tree is in order to choose which of the two algorithms would be faster on a given tree.

Other properties for pruning with sub-additive penalties follow from Theorem 2 and known results about the CART trees. For example, pruning with a sub-additive penalty always produces unique pruned subtrees, and the family of pruned subtrees is nested.

Families of prunings are useful when the appropriate family member needs to be chosen by cross-validation. When this is the case, Theorem 2 implies that sub-additive penalties will never provide a richer class of options than an additive penalty.

Finally, we remark that the proof of Theorem 2 only requires \( \rho \) to be nonincreasing, not necessarily additive. The theorem may also be of practical use in this more general setting.

### 4.2 Proof of Theorem 2

We require the following lemma. Recall that for \( i < j \) we define

\[
\gamma_{i,j} = \frac{\rho(T^i) - \rho(T^j)}{\Phi(j) - \Phi(i)}.
\]

**Lemma 2.** Let \( a, b, c \) be positive integers with \( a > b > c \). The following are equivalent:

(i) \( \gamma_{c,a} < \gamma_{c,b} \)

(ii) \( \gamma_{b,a} < \gamma_{c,b} \)

(iii) \( \gamma_{b,a} < \gamma_{c,a} \)
The three statements are also equivalent if we replace \(<\) by \(\leq, >, \geq, \) or \(=\).

**Proof.** A straightforward calculation establishes

\[
(\Phi(a) - \Phi(c))(\Phi(b) - \Phi(c)) [\gamma_{c,a} - \gamma_{c,b}] = (\Phi(a) - \Phi(b))(\Phi(b) - \Phi(c)) [\gamma_{b,a} - \gamma_{c,b}] = (\Phi(a) - \Phi(b))(\Phi(a) - \Phi(c)) [\gamma_{b,a} - \gamma_{c,a}].
\]

The lemma follows from these identities and the fact that \(\Phi\) is increasing.

The lemma may also be established by geometric considerations. Consider the three points \(p_a, p_b, p_c \in \mathcal{P}\) defined by \(T^a, T^b, T^c\), respectively (see Section 2.2). Note that \(p_a\) is above and to the left of \(p_b\), which is above and left of \(p_c\). Then \(\gamma_{a,b}\) is the negative slope of the line segment connecting \(p_a\) and \(p_b\), and similarly for the other two combinations of points. Then the statements in (i), (ii), and (iii) are all true if and only if \(p_b\) is strictly above the line connecting \(p_a\) and \(p_c\). Similarly, all three statements hold with equality if and only if \(b\) lies on the line joining \(p_a\) and \(p_c\), and so on.

To prove the theorem, first notice that \(U_1 = V_1 = 1\) is the smallest tree \(S \preceq T\) such that \(\rho(S) = \rho(T)\). The theorem follows by induction if we can show \(V_j \in \{U_1, \ldots, U_m\} \Rightarrow V_{j+1} \in \{U_1, \ldots, U_m\}\). To show this, we suppose it is not true and arrive at a contradiction. Assume there exists \(j\) such that \(V_j \in \{U_1, \ldots, U_m\}\) but \(V_{j+1} \notin \{U_1, \ldots, U_m\}\). Then \(V_j = U_i\) for some \(i\). Moreover, there exists \(k \geq 0\) such that \(|U_{i+k}| > |V_{j+1}| > |U_{i+k+1}|\).

Introduce the notation

\[
\gamma_{i,j}^\kappa = \frac{\rho(T^i) - \rho(T^j)}{\Phi^\kappa(j) - \Phi^\kappa(i)},
\]

for \(\kappa = 1, 2\), and \(i < j\). Define \(p = |U_i| = |V_j|, q = |U_{i+k}|, r = |V_{j+1}|,\) and \(s = |U_{i+k+1}|\). Then \(p \geq q > r > s\). We will show

\[
\gamma_{s,r}^2 \leq \gamma_{r,q}^2 < \gamma_{r,p}^2 < \gamma_{s,r}^2,
\]

thus arriving at our desired contradiction. Denote these inequalities by I1, I2, and I3, respectively.

To establish I1, observe

\[
\gamma_{s,q}^2 = \frac{\rho(T^s) - \rho(T^q)}{\Phi^2(q) - \Phi^2(s)} = \frac{\rho(T^s) - \rho(T^q)}{\Phi^1(q) - \Phi^1(s)} \cdot \frac{\Phi^1(q) - \Phi^1(s)}{\Phi^2(q) - \Phi^2(s)} \leq \frac{\rho(T^r) - \rho(T^q)}{\Phi^1(q) - \Phi^1(r)} \cdot \frac{\Phi^1(q) - \Phi^1(r)}{\Phi^2(q) - \Phi^2(s)} = \frac{\rho(T^r) - \rho(T^q)}{\Phi^2(q) - \Phi^2(r)} \cdot \frac{\Phi^1(q) - \Phi^1(r)}{\Phi^2(q) - \Phi^2(s)} \leq \gamma_{r,q}^2.
\]

where the first inequality follows from Lemma 1 and the second inequality comes from the definition of \(\Phi^1 > \Phi^2\). Since \(\gamma_{s,q}^2 \leq \gamma_{r,q}^2\), Lemma 2 (iii \(\Rightarrow\) ii) implies \(\gamma_{s,r}^2 \leq \gamma_{s,q}^2\), establishing I1.

To show I2, assume \(p \neq q\) (otherwise the inequality is trivial). Note that Lemma 1 implies \(\gamma_{r,p}^2 \leq \gamma_{q,p}^2\). Lemma 2 (iii \(\Rightarrow\) i) then implies I2.

Finally, by Lemma 1, we have \(\gamma_{r,p}^2 < \gamma_{s,p}^2\). I3 follows from Lemma 2 (iii \(\Rightarrow\) ii).
5 Conclusion

We have presented two polynomial time algorithms for pruning and generating families of prunings using non-additive penalties. The first algorithm applies to arbitrary penalties, while the second algorithm applies to sub-additive penalties. Both algorithms have a worst-case run time of $O(|T|^2)$. The first algorithm achieves the worst-case for balanced trees (i.e., when depth$(T) \propto \log |T|$), and only requires $O(|T|)$ operations for lop-sided trees (e.g., when every left descendant is a leaf node). The second algorithm has its worst case when $T$ is unbalanced, and runs in $O(|T| \log |T|)$ time for balanced trees.

The second algorithm is based on a general theorem that, as a special case, implies that the family of prunings induced by a sub-additive penalty is a subset of the family induced by an additive penalty. This implies that sub-additive families are unique and nested. It also implies a negative result: when cross-validation is to be used to select the best member from a family of prunings, sub-additive penalties will never offer a richer set of options than an additive penalty. It does not imply, however, that sub-additive penalties should never be used in conjunction with cross-validation. In principle it is possible that the extra trees gained by an additive penalty actually have poorer performance, in which case it is better to not even consider them as a possibility.

An immediate impact of this work is in the area of classification tree design. It has recently been shown that sub-additive penalties are more appropriate than an additive penalty for pruning classification trees (as discussed in the introduction). The work presented here provides for efficient implementation of such strategies and characterizes the resulting pruned subtrees.

It is quite possible that other machine learning and signal processing tree-based methodologies employ an additive penalty simply for convenience, when perhaps a non-additive penalty would be more appropriate. We hope the present work might lead to a reassessment of such problems.

Acknowledgment

The author would like to thank Robert Nowak for helpful comments.

References


