Binary Partitions with Approximate Minimum Impurity

Eduardo S. Laber [*] [1] **Marco Molinaro** [*] [1] **Felipe de A. Mello Pereira** [1]

Abstract

The problem of splitting attributes is one of the main steps in the construction of decision trees. In order to decide the best split, impurity measures such as Entropy and Gini are widely used. In practice, decision-tree inducers use heuristics for finding splits with small impurity when they consider nominal attributes with a large number of distinct values. However, there are no known guarantees for the quality of the splits obtained by these heuristics. To fill this gap, we propose two new splitting procedures that provably achieve near-optimal impurity. We also report experiments that provide evidence that the proposed methods are interesting candidates to be employed in splitting nominal attributes with many values during decision tree/random forest induction.

1. Introduction

Decision Trees as well as ensemble methods that use them (e.g. Random Forests and Gradient Boosted Trees) are among the most popular methods for classification tasks. It is widely known that decision trees, specially small ones, are easy to interpret while ensemble methods usually yield to more stable/accurate classifications.

When building a decision tree, in each node, one needs to address two problems: which attribute shall be used for branching, and how to split the chosen attribute, i.e., which values of the attribute go to each branch. For the first problem we refer the reader to (Hothorn et al., 2006; Nowozin, 2012). Here we consider the latter, which is a well-studied problem (Breiman et al., 1984; Nadas et al., 1991; Chou, 1991; Burshtein et al., 1992; Coppersmith et al., 1999; Elomaa & Rousu, 2004). More specifically, we focus on nominal attributes (i.e. finite set of possible values with no additional structure such as order).

An important design choice is whether to use multiway splits or binary splits. One possibility is splitting a nominal attribute with n distinct values into n branches, one for each value. When n is large, this option may lead to a severe data fragmentation, which makes the classification task harder and increases the risk of data overfit since we may have only a few examples associated with each branch. Note that any decision tree obtained via multiway splits can be simulated by a decision tree that only uses binary splits. Thus, we focus our study on binary splits.

The standard approach for deciding the split is to search for 'pure' partitions of the set of examples, that is, partitions in which each branch is very homogeneous with respect to the class distribution of its examples. To measure how impure each branch is, impurity measures are often employed. An impurity measure maps a vector $\mathbf{u} = (u_1, \ldots, u_k)$, counting how many examples of each class we have in a node (branch), into a non-negative scalar [1]. Arguably, two of the most classical impurity measures are the *Gini* impurity

$$i_{Gini}(\mathbf{u}) = \sum_{i=1}^{k} \frac{u_i}{\|\mathbf{u}\|_1} \left(1 - \frac{u_i}{\|\mathbf{u}\|_1}\right),$$

which is used in the CART package (Breiman et al., 1984), and the *Entropy* impurity

$$i_{Entr}(\mathbf{u}) = -\sum_{i=1}^{k} \frac{u_i}{\|\mathbf{u}\|_1} \log\left(\frac{u_i}{\|\mathbf{u}\|_1}\right),$$

that, along with its variants, is used in the C4.5 decision tree inducer (Quinlan, 1992). Given an attribute and an impurity measure, the goal is then to find a binary split (L^*, R^*) for the attribute values that induces a binary partition of the set of examples with minimum weighted impurity, where the weights are given by the number of examples that lie into each of the two branches.

For classification tasks with only two classes, Breiman et al. (Breiman et al., 1984) proposed an algorithm that finds a partition with minimum weighted impurity in $O(n \log n)$ time for a family of impurity measures that include both Gini and Entropy. For nominal attributes with a small number of distinct values n, the best partition can be found in $O(2^n)$

[*] Equal contribution [1] Departamento de Informática, PUC-RIO, Brazil. Correspondence to: Eduardo Laber <eduardo.laber1@gmail.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

[1] In the original definition an impurity measure maps a vector of probabilities into a non-negative scalar.

time by an exhaustive search. However, when $k > 2$ and n is large (e.g. states of a country, letters of the alphabet, breed of an animal), these methods are not effective. Thus, heuristics are commonly used (Nadas et al., 1991; Chou, 1991; Mehta et al., 1996; Coppersmith et al., 1999; Loh, 2009). Despite the importance of this problem, little is known about its computational complexity and the quality (approximation guarantee) of its heuristics. Therefore, our goal here is contributing to fill this gap.

1.1. Problem Description

Given an impurity measure i (e.g. i_{Gini}), define I as $I(\mathbf{v}) = \|\mathbf{v}\|_1 \cdot i(\mathbf{v})$ for all vectors $\mathbf{v}$. This scaled impurity I is called *frequency-weighted impurity measure* in (Coppersmith et al., 1999) and will be used to formalize our problem.

Consider a nominal attribute A that may take n possible values $a_1, \ldots, a_n$. The ℓ-ary Partition with Minimum Weighted Impurity Problem (ℓ-*PMWIP*) can be described abstractly as follows. We are given a collection of n vectors $V \subset \mathbb{R}^k$, where the ith coordinate of the jth vector counts the number of examples in class i for which the attribute A has value a_j. We are also given a scaled impurity measure I. The goal is to partition V into ℓ disjoint groups of vectors $V_1, \ldots, V_\ell$ so as to minimize the sum of the weighted impurities

$$\sum_{m=1}^{\ell} I\Bigg(\sum_{\mathbf{v} \in V_m} \mathbf{v} \Bigg).$$

We focus on binary partitions (2-*PMWIP*) and on a broad class of impurity measures that includes both Gini and Entropy. These impurities have the form

$$I(\mathbf{v}) = \|\mathbf{v}\|_1 \left(\sum_{i=1}^{k} f\left(\frac{v_i}{\|\mathbf{v}\|_1} \right) \right),$$

where f is a strictly concave function that satisfies a certain property related to its curvature. The formal definition of this class is postponed to Section 2.1.

1.2. Our Results

In this paper we propose new splitting procedures that provably achieve near-optimal impurity. Our starting point is one of the results presented in (Burshtein et al., 1992; Coppersmith et al., 1999) that states that for every instance of 2-*PMWIP*, where the impurity I satisfies certain conditions, there exists an optimal binary partition that is induced by a homogeneous hyperplane in $\mathbb{R}^k$. Building upon this result we prove that an optimal binary partition can be obtained by a non-homogeneous hyperplane whose normal direction belongs to the box $[0, 1]^k$. Then, motivated by this observation, we propose and analyze two methods that belong to a family of algorithms that search for binary partitions with reduced impurity by considering hyperplanes in $\mathbb{R}^k$ whose normal lie in the hypercube $\{0, 1\}^k$.

Our first algorithm, the `Hypercube Cover` (`HcC` for short), is closely related with the well established `Twoing` method proposed in (Breiman et al., 1984). We prove that `HcC` has a 2-approximation for every impurity measure in our class. A drawback of this method, however, is its running time proportional to 2^k. Given this limitation, we present `LargestClassAlone` (`LCA` for short), a simple algorithm that runs in $O(nk + n \log n)$ time and provides a $(3 + \sqrt{3})$-approximation for every impurity measure in our class. This material is covered in Section 3. Furthermore, in Section 4, we show that the approximation ratio of `LCA` for Gini and Entropy impurities is indeed much better, being at most 2 for the former and at most 3 for the latter. We also show that, unless $P = NP$, it is not possible to find the partition with minimum impurity in polynomial time, even for the Entropy impurity.

To complement our theoretical findings, in Section 5 we present a set of experiments where we compare the proposed methods with `PCext` and `SLIQext`, the two splitting methods that obtained the best results in the study reported in (Coppersmith et al., 1999). Our experiments provide evidence that both methods proposed in this paper are interesting candidates to be used in splitting nominal attributes with many values during decision tree/ random forest induction: `HcC` is preferable when the number of classes is small and `LCA` is a good alternative when speed is an issue.

We believe that our set of results contributes to improving the current knowledge on a classical and still relevant problem for both the Machine Learning and Data Mining communities.

1.3. Related Work

There have been theoretical investigations on methods to compute the best split efficiently (Breiman et al., 1984; Chou, 1991; Burshtein et al., 1992; Coppersmith et al., 1999; Kurkoski & Yagi, 2014). As mentioned above, for the 2-class problem, Breiman et. al. (Breiman et al., 1984) presented a simple algorithm that finds the best binary partition in $O(n \log n)$ time for impurity measures in a certain class that includes both Gini and Entropy. The correctness of this algorithm relies on a theorem, also proved in (Breiman et al., 1984), which is generalized for $k > 2$ classes and multiway partitions in (Chou, 1991; Burshtein et al., 1992; Coppersmith et al., 1999). Basically, these theorems provide necessary conditions for partitions with minimum impurity and can be used to restrict the set of partitions that need to be considered, as in the family of algorithms we study here. However, despite their usefulness, these conditions do not yield a method that has running time polynomial on n and k.

Some heuristics for computing suboptimal partitions are available in the literature (Breiman et al., 1984; Nadas et al., 1991; Mehta et al., 1996; Coppersmith et al., 1999; Loh, 2009). For none of them approximation guarantees are available. The conclusion of the experiments reported in (Coppersmith et al., 1999) is that `PCext`, one of the methods proposed in that paper, overcomes `Flip Flop` (Nadas et al., 1991) and `SLIQ` (Mehta et al., 1996) in terms of running time and the impurity of the partitions found.

Recently, motivated by applications on signal processing (e.g. construction of polar codes (Tal & Vardy, 2013)), the problem of computing the quantization of the output of a Discrete Memoryless Channel (DMC) that provides the maximum mutual information with the DMC's input has attracted a considerable attention in the Information Theory community (Tal & Vardy, 2013; Kurkoski & Yagi, 2014; Kartowsky & Tal, 2017; Pereg & Tal, 2017; Nazer et al., 2017). Kurkoski and Yagi (Kurkoski & Yagi, 2014) observed that this problem is equivalent to ℓ-*PMWIP* when the impurity measure is the Entropy, and proved that it can be solved in polynomial time when $k = 2$. In (Gülcü et al., 2016; Nazer et al., 2017; Pereg & Tal, 2017; Kartowsky & Tal, 2017), upper and lower bounds on the difference between the entropy impurity of the n-ary partition and the optimal ℓ-ary partition are proved. These bounds do not imply constant approximations for the problem we consider here.

2. Preliminaries

We start defining some notations employed throughout the paper. An input for 2-*PMWIP* is a pair (V, I), where V is a collection of non-null vectors in $\mathbb{R}^k$ with non-negative integer coordinates and I is a scaled impurity measure. We assume that the vector $\sum_{\mathbf{v}\in V}\mathbf{v}$ has no zero coordinates for otherwise we would have an instance with less than k classes. For a set of vectors L, the impurity $I(L)$ of L is given by $I(\sum_{\mathbf{v}\in L}\mathbf{v})$. The impurity of a binary partition (L, R) of the set V is then $I(L)+I(R)$. We use $\mathsf{opt}_I(V)$ to denote the minimum possible impurity for a binary partition of V and, whenever the context is clear, we omit I from $\mathsf{opt}_I(V)$. We say that a partition (L^*, R^*) is optimal for input (V, I) iff $I(L^*) + I(R^*) = \mathsf{opt}_I(V)$.

We use bold face to denote vectors. Given two vectors $\mathbf{u} = (u_1, \ldots, u_k)$ and $\mathbf{v} = (v_1, \ldots, v_k)$ we use $\mathbf{u}\cdot\mathbf{v}$ to denote their inner product and $\mathbf{u}\circ\mathbf{v} = (u_1v_1, \ldots, u_kv_k)$ to denote their component-wise (Hadamard) product. We use $\mathbf{0}$ and $\mathbf{1}$ to denote the vectors in $\mathbb{R}^k$ with all coordinates equal to 0 and 1, respectively. For a non-null vector $\mathbf{v}\in\mathbb{R}^k_+$ we use $\pi(\mathbf{v}) = \mathbf{v}/\|\mathbf{v}\|_1$ to denote the vector obtained by normalizing $\mathbf{v}$ w.r.t. to the ℓ_1 norm. We use $[m]$ to denote the set of the first m positive integers.

Due to space constraints we omit most of the proofs. However, all of the them can be found in the full version available in the supplementary material.

2.1. Impurity Measures

We are interested in the class $\mathcal{C}$ of scaled impurity measures I that satisfy

$$I(\mathbf{u}) = \|\mathbf{u}\|_1 \sum_{i=1}^{dim(\mathbf{u})} f\left(\frac{u_i}{\|\mathbf{u}\|_1}\right), \tag{P0}$$

where $dim(\mathbf{u})$ is the dimension of vector $\mathbf{u}$ and $f : \mathbb{R}\to\mathbb{R}$ is a function satisfying the following conditions:

1. $f(0) = f(1) = 0$ (P1)
2. f is strictly concave in the interval [0,1] (P2)
3. For all $0 < p \le q \le 1$

$$f(p) \le \frac{p}{q}\cdot f(q) + q\cdot f\left(\frac{p}{q}\right). \tag{P3}$$

Impurity measures satisfying conditions (P0)-(P2) are called *frequency-weighted impurity measures with concave functions* (Coppersmith et al., 1999). These impurities measures are superadditive.

Lemma 2.1 (Lemma 1 in (Coppersmith et al., 1999))**.** *If I satisfies (P0)-(P2) then for every vectors $\mathbf{u}_L$ and $\mathbf{u}_R$ in $\mathbb{R}^k_+$, we have $I(\mathbf{u}_L+\mathbf{u}_R) \ge I(\mathbf{u}_L) + I(\mathbf{u}_R)$.*

Although property (P3) is not particularly intuitive it can be shown that if a simple constraint ($xf''(x)$ is non-increasing in [0,1]) is imposed on the second derivative f'' of f then (P3) is also satisfied.

From (Coppersmith et al., 1999) we know that both I_{Gini} and I_{Entr} satisfy (P0)-(P2). It is not difficult to verify that they also satisfy (P3).

Lemma 2.2. *The Gini measure I_{Gini} and the Entropy measure I_{Entr} belong to $\mathcal{C}$.*

The last lemma of this subsection shows that the impurity measures of our class satisfy a subsystem property. It will be used in our analysis to relate the impurity of partitions for instances with k classes with the impurity of partitions for instances with 2 classes.

Lemma 2.3 (Subsystem Property)**.** *Let I be an impurity measure in $\mathcal{C}$. Then, for every $\mathbf{u}\in\mathbb{R}^k_+$ and every $\mathbf{d}\in[0,1]^k$,*

$$I(\mathbf{u}) \le I\Big((\mathbf{u}\cdot\mathbf{d}, \mathbf{u}\cdot(\mathbf{1}-\mathbf{d}))\Big) + I(\mathbf{u}\circ\mathbf{d}) + I(\mathbf{u}\circ(\mathbf{1}-\mathbf{d}))$$

Proof's sketch. Note that by the definition of I, the desired inequality is invariant to scaling $\mathbf{u}$; thus, we assume without loss of generality that $\|\mathbf{u}\|_1 = 1$. The left-hand side of the inequality is then $\sum_i f(u_i)$.

The result is then obtained by applying the following steps: (i) The subadditivity of f is used to obtain $f(u_i) \leq f(d_i u_i) + f((1-d_i)u_i)$; (ii) Property (P3) is used, with $p = d_i u_i$ and $q = \mathbf{d} \cdot \mathbf{u}$, to upper bound $f(d_i u_i)$; (iii) The same property is used with $p = (1-d_i)u_i$ and $q = (\mathbf{1}-\mathbf{d})\cdot\mathbf{u}$ to upper bound $f(u_i(1-d_i))$ and (iv) The upper bound on $f(u_i)$ derived in the previous steps is added over all i to get the right-hand side of the desired inequality. □

2.2. Necessary conditions for optimal partitions

We end our section of preliminaries presenting two results that give necessary conditions for optimal partitions. The first one, proved in (Breiman et al., 1984), yields to an $O(n \log n)$ time algorithm for 2-*PMWIP* when the number of classes k is 2. The second one works for 2-*PMWIP*, with arbitrary k, and it is a reduced version of a more general theorem that also works for ℓ-ary partitions (Burshtein et al., 1992; Coppersmith et al., 1999). Both results are stated using our notation.

Theorem 2.4 (Theorem 4.5 of (Breiman et al., 1984))**.** *Let I be an impurity measure satisfying properties (P0)-(P2) and let $V_2 \subseteq \mathbb{R}^2_+$. Moreover, for every $\mathbf{v} = (v_1, v_2) \in V_2$ let $r(\mathbf{v}) = v_1/\|\mathbf{v}\|_1$. Furthermore, let P_j be the set containing the first j vectors of V_2 when those are sorted with respect to $r()$. Then $(P_j, V_2 \setminus P_j)$, for some $j \in [n-1]$, is an optimal partition for the instance (V_2, I).*

Lemma 2.5 (Hyperplanes Lemma (Burshtein et al., 1992; Coppersmith et al., 1999))**.** *Let I be an impurity measure satisfying properties (P0)-(P2). If (L^*, R^*) is an optimal partition for an instance (V, I), then there is a vector $\mathbf{d}^* \in \mathbb{R}^k$ such that $\mathbf{d}^* \cdot \pi(\mathbf{v}) < 0$ for every $\mathbf{v} \in L^*$ and $\mathbf{d}^* \cdot \pi(\mathbf{v}) > 0$ for every $\mathbf{v} \in R^*$.*

3. Constant Approximations for Impurity Measures in $\mathcal{C}$

In this section we present approximation algorithms for finding binary partitions with reduced impurity. We first analyze a general hyperplane-based procedure, and later specialize it to obtain different approximation algorithms.

3.1. Analysis of a general hyperplane-based procedure

A direct consequence of the Hyperplanes Lemma (Lemma 2.5) above is that the search of the optimal partition can be reduced to the search of a direction in $\mathbb{R}^k$. In fact, it is easy to see that we can normalize these directions to be in $[0,1]^k$, at the expense of working with non-homogeneous hyperplanes, as it is shown in the next proposition.

Proposition 3.1. *Let (L^*, R^*) be an optimal partition for input (V, I) and let $\mathbf{d}^* \in \mathbb{R}^k$ be such that $\mathbf{d}^* \cdot \pi(\mathbf{v}) < 0$ for every $\mathbf{v}$ in L^* and $\mathbf{d}^* \cdot \pi(\mathbf{v}) > 0$ for every $\mathbf{v}$ in R^*.*

Then, there is a direction $\mathbf{d} \in [0,1]^k$ and a constant C such that $\mathbf{d} \cdot \pi(\mathbf{v}) < C$ for every $\mathbf{v}$ in L^* and $\mathbf{d} \cdot \pi(\mathbf{v}) > C$ for every $\mathbf{v}$ in R^*.

The previous observation motivates the definition of a family of algorithms indexed by a direction $\mathbf{d} \in [0,1]^k$. The algorithm $\mathcal{B}_\mathbf{d}$ searches for a partition with reduced impurity by considering all the $n-1$ partitions of the input set V induced by the hyperplanes with normal $\mathbf{d}$ (Algorithm 1).

```
Algorithm 1 $\mathcal{B}_\mathbf{d}$ ($V$: collection of vectors, $I$: impurity measure)
For each $\mathbf{v}$ in $V$ let $r(\mathbf{v}) = (\mathbf{d} \cdot \mathbf{v})/\|\mathbf{v}\|_1$
Rank the vectors in $V$ according to $r(\mathbf{v})$
for $j = 1, \ldots, n-1$ do
    $P_j \leftarrow$ subset of $V$ containing the $j$ vectors with the largest value of $r(\cdot)$
    Evaluate the impurity of partition $(P_j, V \setminus P_j)$
end for
Return the partition $(P_{j^*}, V \setminus P_{j^*})$ with the smallest impurity found in the loop
```

We present a general analysis of the quality of solution produced by algorithm $\mathcal{B}_\mathbf{d}$ when $\mathbf{d} \in \{0,1\}^k$. More specifically, we prove the following theorem:

Theorem 3.2. *Let $I(\mathcal{B}_\mathbf{d})$ be the impurity of the partition returned by $\mathcal{B}_\mathbf{d}$ for an instance (V, I). Then, for every direction $\mathbf{d} \in \{0,1\}^k$ we have*

$$\frac{I(\mathcal{B}_\mathbf{d})}{\mathsf{opt}(V)} \leq 1 + \frac{I(\mathbf{u} \circ \mathbf{d}) + I(\mathbf{u} \circ (\mathbf{1}-\mathbf{d}))}{\min_{\mathbf{d}' \in \{0,1\}^k} \{I(\mathbf{u} \circ \mathbf{d}') + I(\mathbf{u} \circ (\mathbf{1}-\mathbf{d}'))\}} \tag{1}$$

The bound given by this theorem is the basis for the approximation algorithms obtained in the next subsections since it motivates the use of a direction $\mathbf{d}$ such that $I(\mathbf{u} \circ \mathbf{d}) + I(\mathbf{u} \circ (\mathbf{1}-\mathbf{d}))$ is minimized. The remainder of this section is dedicated to prove Theorem 3.2.

One of the key ideas of the proof is to establish a relation between the impurity of the partition obtained by $\mathcal{B}_\mathbf{d}$ for the k-class instance (V, I) and the optimal impurity for the *2-class* instance obtained by collapsing all the classes corresponding to the coordinates of $\mathbf{d}$ with value 0 into one "super class", and all classes corresponding to the coordinates of $\mathbf{d}$ with value 1 into another super class. Recall that each vector $\mathbf{v} \in V \subseteq \mathbb{R}^k_+$, which corresponds to an attribute value, counts in its coordinates the number of examples of each of the k classes with the given attribute value. Then, in the collapsed 2-class instance, this vector count becomes simply $(\sum_{i:\mathbf{d}_i=1} v_i, \sum_{i:\mathbf{d}_i=0} v_i) = (\mathbf{v}\cdot\mathbf{d}, \mathbf{v}\cdot(1-\mathbf{d}))$. Thus, define the operation $collapse_{\mathbf{d}} : \mathbb{R}^k_+ \to \mathbb{R}^2_+$ that maps $\mathbf{v} \mapsto (\mathbf{v}\cdot\mathbf{d}, \mathbf{v}\cdot(1-\mathbf{d}))$. Moreover, for a set of vectors S, define $collapse_{\mathbf{d}}(S)$ as the set obtained applying $collapse_{\mathbf{d}}()$ to each vector of S. Therefore, from a k-class instance (V, I) and a direction $\mathbf{d} \in \{0,1\}^k$, we obtain the collapsed 2-class instance $(collapse_{\mathbf{d}}(V), I)$.

The main motivation for looking at 2-class instances is that we know from Theorem 2.4 that an optimal partition can be obtained by sweeping the vectors according to some order, which is very similar to what algorithm $\mathcal{B}_{\mathbf{d}}$ is doing. To make this connection precise, let $\mathcal{A}_{\mathbf{d}}$ be the algorithm obtained by modifying Line 5 of $\mathcal{B}_{\mathbf{d}}$ so that the impurity of the binary partition $(collapse_{\mathbf{d}}(P_j), collapse_{\mathbf{d}}(V \setminus P_j))$ is evaluated, rather than the impurity of $(P_j, V \setminus P_j)$. The following proposition states that the impurity $\mathcal{B}_{\mathbf{d}}$ is at most that of $\mathcal{A}_{\mathbf{d}}$ and that essentially the latter solves optimally the collapsed 2-class instance.

Proposition 3.3. *Let (L, R) be the partition returned by $\mathcal{A}_{\mathbf{d}}$ for the k-class instance (V, I) and let $I(\mathcal{A}_{\mathbf{d}})$ be the impurity of (L, R). Then: (i) $I(\mathcal{B}_{\mathbf{d}}) \le I(\mathcal{A}_{\mathbf{d}})$ and (ii) $(collapse_{\mathbf{d}}(L), collapse_{\mathbf{d}}(R))$ is an optimal partition for the 2-class instance $(collapse_{\mathbf{d}}(V), I)$.*

Given the first item of the above proposition, to prove Theorem 3.2, it suffices to upper bound the impurity of the partition (L, R) returned by $\mathcal{A}_{\mathbf{d}}$. To simplify the notation, let $\mathbf{u} = \sum_{\mathbf{v}\in V} \mathbf{v}$, $\mathbf{u}_L = \sum_{\mathbf{v}\in L} \mathbf{v}$, and $\mathbf{u}_R = \sum_{\mathbf{v}\in R} \mathbf{v}$. Also, for a direction $\mathbf{d}$ in $\{0,1\}^k$ let $\bar{\mathbf{d}} = \mathbf{1} - \mathbf{d}$. The impurity of the partition (L, R) constructed by $\mathcal{A}_{\mathbf{d}}$ is

$$I(\mathcal{A}_{\mathbf{d}}) = I(\mathbf{u}_L) + I(\mathbf{u}_R).$$

From the Subsystem Property (Lemma 2.3) we get

$$\begin{aligned} I(\mathcal{A}_{\mathbf{d}}) \le\ & I\left((\mathbf{d}\cdot\mathbf{u}_L, \bar{\mathbf{d}}\cdot\mathbf{u}_L)\right) + I(\mathbf{d}\circ\mathbf{u}_L) + I(\bar{\mathbf{d}}\circ\mathbf{u}_L) \\ & + I\left((\mathbf{d}\cdot\mathbf{u}_R, \bar{\mathbf{d}}\cdot\mathbf{u}_R)\right) + I(\mathbf{d}\circ\mathbf{u}_R) + I(\bar{\mathbf{d}}\circ\mathbf{u}_R) \\ & = \mathsf{opt}(collapse_{\mathbf{d}}(V)) + I(\mathbf{d}\circ\mathbf{u}_L) + I(\bar{\mathbf{d}}\circ\mathbf{u}_L) + \\ & I(\mathbf{d}\circ\mathbf{u}_R) + I(\bar{\mathbf{d}}\circ\mathbf{u}_R), \end{aligned} \tag{2}$$

where the last identity follows from the item (ii) of Proposition 3.3 because $(\mathbf{d}\cdot\mathbf{u}_L, \bar{\mathbf{d}}\cdot\mathbf{u}_L) = \sum_{\mathbf{v}\in collapse_{\mathbf{d}}(L)} \mathbf{v}$ and $(\mathbf{d}\cdot\mathbf{u}_R, \bar{\mathbf{d}}\cdot\mathbf{u}_R) = \sum_{\mathbf{v}\in collapse_{\mathbf{d}}(R)} \mathbf{v}$.

Now we need to upper bound the last four terms in the RHS of the equation (2). Using the superadditivity of I (Lemma 2.1) we have

$$I(\mathcal{A}_{\mathbf{d}}) \le \mathsf{opt}(collapse_{\mathbf{d}}(V)) + I(\mathbf{d}\circ\mathbf{u}) + I(\bar{\mathbf{d}}\circ\mathbf{u}) \tag{3}$$

Now we devise lower bounds on $\mathsf{opt}(V)$. The first lower bound captures the intuitive fact that the impurity in the multi-class problem is at least as large as that in the collapsed 2-class problem.

Lemma 3.4. *For any input V and $\mathbf{d} \in \{0,1\}^k$, we have $\mathsf{opt}(V) \ge \mathsf{opt}(collapse_{\mathbf{d}}(V))$.*

For our second lower bound, we consider the relaxed problem where each example corresponds to a distinct attribute value and the full class distribution is equal to that of V. Formally, from the original instance (V, I) we consider the instance (V', I), where V' contains u_i copies of the standard basis vector $\mathbf{e}_i$ for $i = 1, \ldots, k$.

Let $\mathsf{opt}(V')$ be the optimal solution to this relaxed problem. It is clear that $\mathsf{opt}(V') \le \mathsf{opt}(V)$, since any partition for the collection V can be realized in the collection V'.

It follows from Lemma 2.5 that in the optimal partition of V' all vectors associated with the same class (standard basis vectors) end up on the same side of the partition. Thus, the optimal solution for instance (V', I) corresponds to a partition of the set of classes, and so

$$\mathsf{opt}(V) \ge \mathsf{opt}(V') = \min_{\mathbf{d}'\in\{0,1\}^k} \{I(\mathbf{u}\circ\mathbf{d}') + I(\mathbf{u}\circ(\mathbf{1}-\mathbf{d}'))\}.$$

Thus, by using the upper bound given by equation (3), the lower bound given by Lemma 3.4 and the previous inequality we obtain that the RHS of inequality (1) is an upper bound on the approximation ratio of algorithm $\mathcal{A}_{\mathbf{d}}$. This bound together with the first item of Proposition 3.3 establish Theorem 3.2.

3.2. The `Hypercube Cover` procedure

The `HcC` method simply returns the best $\mathcal{B}_{\mathbf{d}}$ among all possible directions $\mathbf{d} \in \{0,1\}^k$, hence it equals $\mathcal{B}_{\mathbf{d}'}$ for $\mathbf{d}'$ satisfying

$$I(\mathcal{B}_{\mathbf{d}'}) = \min_{\mathbf{d}\in\{0,1\}^k} I(\mathcal{B}_{\mathbf{d}}).$$

To find $\mathbf{d}'$, the algorithm examines all the 2^k binary vectors in $\{0,1\}^k$.

Given the general analysis provided by Theorem 3.2, it follows directly that `HcC` has the following approximation guarantee.

Theorem 3.5. *`HcC` is a 2-approximation algorithm for every impurity measure in $\mathcal{C}$.*

We shall mention that `HcC` is closely related with the `Twoing` method proposed in (Breiman et al., 1984). In fact, `Twoing` considers all 2^k possibilities of grouping the k classes into 2 super classes and, for each possibility, optimally solves the 2-class problem; the best partition w.r.t. the 2-class problem is then returned. In our notation, `Twoing` executes algorithm $\mathcal{A}_{\mathbf{d}}$, rather than $\mathcal{B}_{\mathbf{d}}$, for all directions $\mathbf{d} \in \{0,1\}^k$ and returns the partition, among the 2^k generated, with minimum impurity with respect to the collapsed problem.

It is also interesting to note that in (Breiman et al., 1984) it was proved that if `Twoing` considers the Gini impurity for solving its 2-class problems then it finds a partition of attribute values that optimizes a specific objective function for the k-class problem that is significantly different from I_{Gini}.

3.3. `LargestClassAlone`: an $O(nk + n\log n)$-time constant approximation

A limitation of `HcC` is its running time, which is exponential on the number of classes k. To address this issue, we show that a simple algorithm with $O(nk + n\log n)$ running time has a constant approximation for our class of impurity measures.

Recall that we are using $\mathbf{u}$ to denote $\sum_{\mathbf{v}\in V}\mathbf{v}$. Given $\mathbf{u}$, let i^* be the index of the coordinate corresponding to the largest value in $\mathbf{u}$, that is, $u_{i^*} \geq u_i$ for all i. Moreover, let $\mathbf{e}_{i^*}$ be the direction where all coordinates are 0 but coordinated i^* whose value is 1. We use `LargestClassAlone` (`LCA` for short) to denote the algorithm $\mathcal{B}_{\mathbf{e}_{i^*}}$. The next result, which also relies on Theorem 3.2, shows that `LCA` has a constant approximation for our class.

Theorem 3.6. `LCA` *is an* $(3+\sqrt{3})-$*approximation for every impurity measure in the class* $\mathcal{C}$.

4. Improved Approximations for Gini and Entropy

Here we show that we can obtain better approximations, with polynomial running time on n and k, when we focus on specific impurity measures. We consider both Gini and Entropy.

The key idea for the improvement is to characterize the direction that minimizes the denominator of the upper bound on the approximation ration given by Theorem 3.2. It will be interesting to observe how Gini and Entropy behave significantly different in this sense, with the latter favoring balanced partition.

4.1. Gini

We prove that `LCA` is a 2-approximation algorithm for I_{Gini}. To achieve this goal we show that, when $I = I_{Gini}$, $\mathbf{e}_{i*}$ is a direction that minimizes the expression $I(\mathbf{u}\circ\mathbf{d}') + I(\mathbf{u}\circ(\mathbf{1}-\mathbf{d}'))$ that appears on the denominator of the righthand side of inequality (1).

Lemma 4.1. *The direction* $\mathbf{e}_{i^*}$ *satisfies*

$$I_{Gini}(\mathbf{u}\circ\mathbf{e}_{i^*}) + I_{Gini}(\mathbf{u}\circ(\mathbf{1}-\mathbf{e}_{i^*})) = \min_{\mathbf{d}\in\{0,1\}^k}\{I_{Gini}(\mathbf{u}\circ\mathbf{d}) + I_{Gini}(\mathbf{u}\circ(\mathbf{1}-\mathbf{d}))\}.$$

A direct consequence of the previous lemma and Theorem 3.2 is that `LCA` gives a 2-approximation for I_{Gini}.

Theorem 4.2. `LCA` *is a 2-approximation for the Gini impurity measure.*

A natural question is whether the analysis is tight. The instance $V = \{(x,0,0),(0,x,0),(c,0,c)\}$, $x >> c > 0$, shows that this is the case for algorithm $\mathcal{A}_{\mathbf{e}_{i^*}}$. For `LCA` (which is $\mathcal{B}_{\mathbf{e}_{i^*}}$) we are not aware whether the approximation is tight or not. The worst example we know has impurity $4/3$ larger than the optimal one.

4.2. Entropy

In this section we show that, for the Entropy impurity, `LCA` achieves an approximation ratio better than the one given by Theorem 3.6.

Let us define the balance of a direction $\mathbf{d}$ in $\{0,1\}^k$ with respect to $\mathbf{u}$ as $\min\{\mathbf{u}\cdot\mathbf{d}, \mathbf{u}\cdot(\mathbf{1}-\mathbf{d})\}$. The next lemma implies that the most balanced direction with respect to $\mathbf{u}$ is the one that minimizes the denominator on the approximation ratio given by inequality (1).

Lemma 4.3. *Let* $\mathbf{d}$ *and* $\mathbf{d}'$ *be directions in* $\{0,1\}^k$. *Then,*

$$I_{Ent}(\mathbf{u}\circ\mathbf{d}) + I_{Ent}(\mathbf{u}\circ(\mathbf{1}-\mathbf{d})) < I_{Ent}(\mathbf{u}\circ\mathbf{d}') + I_{Ent}(\mathbf{u}\circ(\mathbf{1}-\mathbf{d}')) \quad (4)$$

if and only if $\mathbf{d}$ *is more balanced than* $\mathbf{d}'$ *with respect to* $\mathbf{u}$, *that is,* $\min\{\mathbf{d}\cdot\mathbf{u}, (\mathbf{1}-\mathbf{d})\cdot\mathbf{u}\} > \min\{\mathbf{d}'\cdot\mathbf{u}, (\mathbf{1}-\mathbf{d}')\cdot\mathbf{u}\}$.

Let $\mathbf{d}^*$ be the most balanced direction in $\{0,1\}^k$ with respect to $\mathbf{u}$. The previous result together with Theorem 3.2 guarantee that algorithm $\mathcal{B}_{\mathbf{d}^*}$ is a 2-approximation for the Entropy impurity. The direction $\mathbf{d}^*$ can be constructed in $O(k\sum_{\mathbf{v}\in V}\|\mathbf{v}\|_1)$ time using an algorithm for the subset sum problem (Cormen et al., 1998).

Theorem 4.4. *There exists a 2-approximation algorithm for the entropy impurity measure that runs in* $O(k\sum_{\mathbf{v}\in V}\|\mathbf{v}\|_1)$ *time.*

For `LCA` we manage to prove the following bound.

Theorem 4.5. `LCA` *is a* 3*-approximation for the Entropy impurity measure.*

Given Lemma 4.3, a straightforward reduction from PARTITION problem shows that 2-*PMWIP* is NP-hard even when I is the Entropy measure.

Theorem 4.6. *The* 2-PMWIP *for the Entropy impurity measure is NP-Hard.*

The complexity of the problem for the case where I is the scaled Gini impurity measure remains open.

5. Experiments

To complement our theoretical study we report a number of experiments with the methods proposed/analyzed in the previous sections.

5.1. Evaluation of Splits Impurity

Our first set of experiments is very similar to the set presented in (Coppersmith et al., 1999) except for a few details. All experiments are Monte Carlo simulations with 10,000 runs, each using a randomly-generated contingency table for the given number of values n and classes k. By a contingency table we mean a matrix where each row corresponds to a distinct vector of the input V. Each table was created by uniformly picking a number in $\{0, \dots, 7\}$ for each entry. This guarantees a substantial probability of a row/column having some zero frequencies, which is common in practice. Differing from (Coppersmith et al., 1999), if all the entries corresponding to a value or a class are zero, we re-generate the contingency table, since otherwise the number of actual values and classes would not match n and k.

We compared the following splitting methods: HcC, LCA, SLIQext and PCext. SLIQext is a variant, presented in (Coppersmith et al., 1999), of the SLIQ method proposed in (Mehta et al., 1996). It starts with the partition $(V, \emptyset)$ and then it greedily moves, from the 'left' to the 'right' partition, the vector that yields to the partition with minimum impurity until the the partition $(\emptyset, V)$ is reached; the best partition found in this process is returned. PCext is a method proposed in (Coppersmith et al., 1999) that defines the partition for the vectors in V by using a hyperplane in $\mathbb{R}^k$ whose normal direction is the principal direction of a certain contingency table associated with the instance. According to the experiments reported in (Coppersmith et al., 1999), PCext and SLIQext outperformed other available methods, such as the Flip Flop method (Nadas et al., 1991), in terms of the impurity of the partitions found.

Table 1 and 2 show, for different values of n and k, the percentage of times each method is at least as good as the other competitors for Gini and Entropy, respectively. We only show results for $k \leq 9$ because, for larger values of k, HcC becomes non-practical due to its running time. Furthermore, we do not present results for small values of n because, in this case, the optimal partition can be found reasonably quick through an exhaustive search, hence there is no motivation for heuristics.

In general, we observe an advantage of HcC for both the impurity measures, being much more evident for Entropy impurity. We also observe that LCA presents the worst results. Additional experiments where we set the maximum possible value in the contingency table to 2 and 15, rather than to 7, presented similar behavior.

Table 1. Percentage of wins for PCExt, HcC, SLIQext and LCA for Gini impurity. The best result for each configuration is bold faced.

Methods	k n	3	5	7	9
HcC	12	**97.3**	**99.2**	**99.9**	**100.0**
PCext		91.2	88.0	86.6	85.0
SliqExt		89.9	81.9	78.3	75.5
LCA		42.8	19.1	11.5	8.5
HcC	25	**73.9**	**65.8**	**73.3**	**85.3**
PCext		72.7	62.4	58.8	53.2
SliqExt		78.8	64.6	57.9	52.5
LCA		24.3	5.9	2.0	0.8
HcC	50	51.4	33.1	31.0	33.9
PCext		50.6	41.1	40.7	37.8
SliqExt		**68.1**	**53.1**	**47.1**	**42.9**
LCA		16.0	3.3	1.0	0.4

Table 2. Percentage of wins for PCExt, HcC, SLIQext and LCA for Entropy impurity. The best result for each configuration is bold faced.

Methods	k n	3	5	7	9
HcC	12	**98.3**	**99.4**	**100**	**100**
PCext		80.2	74.2	73.2	72.4
SliqExt		87.5	78.2	75.2	72.8
LCA		33.5	13.6	8.3	6.8
HcC	25	**83.3**	**76.9**	**81.0**	**87.7**
PCext		54.4	42.7	39.2	37.7
SliqExt		71.8	57.1	52.1	47.1
LCA		18.5	5.3	2	1.3
HcC	50	**70.0**	**57.4**	**53.5**	**52.5**
PCext		29.5	22.0	21.7	22.1
SLIQext		55.1	42.5	38.8	36.2
LCA		13.4	3.7	1.6	0.8

It is also interesting to observe how far the impurity of the partition generated by a splitting method may be with respect to the best partition found by the other methods. To measure this distance, let us define the relative excess (in percentage) of a partition P w.r.t. a partition Q as $100 \times (I(P)/I(Q) - 1)$. The maximum relative excess observed for the partitions generated by HcC, with respect to the partitions generated by the other methods, was 2% and 1.9% for Gini and Entropy, respectively. For SLIQext, the maximum relative excess observed was 9.4% for Gini and 14% for Entropy. For PCext, we observed 3.7% for Gini and 21.6% for Entropy. Finally, for LCA, we had 22.3% for Gini and 43.6% for Entropy. These numbers suggest that the risk of finding a 'bad' partition is smaller when HcC is used, specially for the Entropy impurity.

Table 3 presents a comparison between the running time of the 4 methods for each configuration of n and k. The numbers are relative to the running time of LCA, which is the fastest of them. Among the other three methods, PCext ob-

Table 3. Each entry is the ratio between the average running time of the method and the average running time of LCA, which is the fastest of them.

Methods	k n	3	5	7	9
HcC	12	1.6	6.8	30.8	138.4
PCext		3.5	3.3	3.2	3.5
SliqExt		5.4	5.5	6.3	7.2
HcC	25	1.7	8.9	41.2	174.7
PCext		5.1	5.6	6	6.1
SliqExt		21.6	24.5	28	30
HcC	50	2.2	10.8	45.4	181.5
PCext		8.6	10.1	10	10.2
SLIQext		90.6	101.2	104.7	107.6

Table 4. The table at the top (bottom) shows the number of datasets in which LCA (HcC) had statistically better/worse accuracy than the other methods. In each entry they are represented by the numbers following the plus/minus signs, respectively.

	PCExt	HcC	SliqExt
Gini	+3 -3	+1 -3	+8 -3
Entropy	+4 -0	+4 -0	+7 -2

	PCExt	LCA	SliqExt
Gini	+2 -2	+3 -1	+8 -3
Entropy	+2 -1	+0 -4	+7 -2

tained the best results. As expected, HcC is very competitive for small values of k and it becomes less competitive when k grows. For the slowest configuration, $n = 50$ and $k = 9$, HcC took, in average, 0.38 seconds. We can also observe that SLIQext becomes less competitive as n grows. All the experiments were executed on a PC Intel i7-6500U CPU with 2.5GHz and 8GB of RAM. The algorithms were implemented in Python 3 using numpy and and are available in https://github.com/felipeamp/icml-2018.

5.2. Decision tree induction

We also carried out a set of experiments to evaluate how the methods behave when they are used in decision tree induction. Here we just present a summary of these experiments – a more complete complete description can be found in the supplementary material.

We employed 11 datasets in total. Eight of them are from the UCI repository: Mushroom, KDD98, Adult, Nursery, Covertype, Cars, Contraceptive and Poker (Lichman, 2013). Two others are available in Kaggle: San Francisco Crime and Shelter Animal Outcome (SF-OpenData; Austin-Animal-Center). The last dataset was created by translating texts from the Reuters database (Lichman, 2013) into phonemes, using the CMU pronouncing dictionary (CMU). We shall note that these datasets were also used in (Laber & de A. Mello Pereira, 2018) where methods for splitting nominal attributes that do not rely on impurity measures are proposed.

We chose these datasets because they have at least 1000 samples and they either contain multi-valued attributes or attributes that can be naturally aggregated to produce multi-valued attributes. For datasets Cars, CoverType, Nursery and Contraceptive we add new nominal attributes that were obtained by aggregating some of the original ones. Additional details are given in the supplementary material.

In this second set of experiments we build decision trees with depth at most 16. We employed a 95% one-tailed paired t-student test to compare the accuracy attained by the methods over 20 3-fold stratified cross-validations. Table 4 shows how LCA and HcC compare with each of the other methods with regards to the number of datasets in which they had statistically better/worse accuracy. As an example, the entry associated with Entropy/PCExt in the top Table 4 shows that out of the 11 datasets, for the Entropy impurity, LCA was statistically better in 4 datasets while PCExt was better in none.

Given the results on the previous section, we were not expecting a strong performance from LCA. However, to our surprise, LCA was quite competitive, performing better than some of the other methods in these datasets, specially for the Entropy impurity measure. HcC, as expected, had a good performance. These results suggest that LCA is also an interesting alternative, specially when speed is an issue.

6. Final Remarks

In this paper we proved that the 2-*PMWIP* is NP-Hard and we devised algorithms with constant approximation guarantee for it. Furthermore, we reported experiments that suggest that our methods proposed are good candidates to be used in splitting nominal attributes with many values during decision tree/random forest induction. HcC has the advantage of generating partitions with lower impurity than other available methods while LCA has the advantage of being very fast.

Some interesting questions remain open. The main one concerns the existence of a FPTAS for 2-*PMWIP*, that is, an algorithm that for every $\epsilon > 0$ obtains an approximation $(1+\epsilon)$ with running time polynomial on n and $1/\epsilon$. Another interesting question regards the existence of algorithms with provably approximation for the *L-PMWIP*, the most general problem where the values of an attribute have to be partitioned into at most L groups.

References

Austin-Animal-Center. Shelter animal outcomes dataset. kaggle.com/c/shelter-animal-outcomes.

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

Burshtein, D., Pietra, V. D., Kanevsky, D., and Nadas, A. Minimum impurity partitions. *Ann. Statist.*, 1992.

Chou, P. A. Optimal partitioning for classification and regression trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4), 1991.

CMU. Cmu pronouncing dictionary. www.speech.cs.cmu.edu/cgi-bin/cmudict.

Coppersmith, Don, Hong, Se June, and Hosking, Jonathan R. M. Partitioning nominal attributes in decision trees. *Data Min. Knowl. Discov*, 3(2):197–217, 1999.

Cormen, T.H., Leiserson, C.E., and Rivest, R.L. *Introduction to Algoritms*. McGraw-Hill Book Company, 1998.

Elomaa, Tapio and Rousu, Juho. Efficient multisplitting revisited: Optima-preserving elimination of partition candidates. *Data Min. Knowl. Discov*, 8(2):97–126, 2004.

Gülcü, Talha Cihad, 0005, Min Ye, and Barg, Alexander. Construction of polar codes for arbitrary discrete memoryless channels. In *ISIT*, pp. 51–55. IEEE, 2016.

Hothorn, Torsten, Hornik, Kurt, and Zeileis, Achim. Unbiased recursive partitioning. *Journal of Computational and Graphical Statistics*, 15(3):651–674, September 2006. ISSN 1061-8600 (print), 1537-2715 (electronic).

Kartowsky, Assaf and Tal, Ido. Greedy-merge degrading has optimal power-law. *CoRR*, abs/1703.04923, 2017. URL http://arxiv.org/abs/1703.04923.

Kurkoski, Brian M. and Yagi, Hideki. Quantization of binary-input discrete memoryless channels. *IEEE Trans. Information Theory*, 60(8):4544–4552, 2014.

Laber, E. S. and de A. Mello Pereira, F. Splitting criteria for classification problems with multi-valued attributes and large number of classes. *Pattern Letters Recognition*, 2018.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Loh. Improving the precision of classification trees. *The Annals of Applied Statistics*, 2009.

Mehta, M., Agrawal, R., and Rissanen, J. SLIQ: A fast scalable classifier for data mining. *Lecture Notes in Computer Science: Proc. 5th Int. Conf. on Extending Database Technology*, 1996.

Nadas, A., Nahamoo, D., Picheny, M. A., and Powell, J. An iterative ip-op approximation of the most informative split in the construction of decision trees. *ieeeassp*, pp. 565–568, 1991.

Nazer, Bobak, Ordentlich, Or, and Polyanskiy, Yury. Information-distilling quantizers. In *ISIT*, pp. 96–100. IEEE, 2017.

Nowozin, Sebastian. Improved information gain estimates for decision tree induction. In *ICML*, 2012.

Pereg, Uzi and Tal, Ido. Channel upgradation for non-binary input alphabets and macs. *IEEE Trans. Information Theory*, 63(3):1410–1424, 2017.

Quinlan, J. Ross. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.

SF-OpenData. San francisco crime dataset. kaggle.com/c/sf-crime.

Tal, Ido and Vardy, Alexander. How to construct polar codes. *IEEE Trans. Information Theory*, 59(10):6562–6582, 2013.