Binary Partitions with Approximate Minimum Impurity

Eduardo S. Laber
laber@inf.puc-rio.br
PUC-RIO

Marco Molinaro
mmolinaro@inf.puc-rio.br
PUC-RIO

Felipe de A. Mello Pereira
felipeamp@gmail.com
PUC-RIO

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. The first one, `Hypercube Cover`, is closely related with the well-established `Twoing` method [Breiman et al. 84]. We prove that `Hypercube Cover` provides a 2-approximation of the minimum impurity for a broad class of impurity measures that includes both Gini and Entropy. In addition, we propose a very simple and efficient procedure that provides a constant approximation for the same class of impurity measures.

We complement our study with a number of experiments that provide evidence that our 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 [10, 20]. Here we consider the latter, which is a well-studied problem [2, 17, 4, 3, 6, 8]. 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

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

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 [2], 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 [22]. 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. [2] 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 $O(2^n)$ 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 [17, 4, 16, 6, 15]. 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 [6] and will be used to formalize our problem.

Consider a nominal attribute A that may take n possible values $a_1, \dots, a_n$. 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 $V_1, \dots, V_\ell$ so as to minimize the sum of the weighted impurities

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

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 that has to do with its curvature. The formal definition of this class is postponed to Section 2.2.

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 [3, 6] 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 [2]. 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 [6]. 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 [2, 4, 3, 6, 12]. As mentioned above, for the 2-class problem, Breiman et. al. [2] 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 [2], which is generalized for $k > 2$ classes and multiway partitions in [4, 3, 6]. 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 [2, 17, 16, 6, 15]. For none of them approximation guarantees are available. The conclusion of the experiments reported in [6] is that `PCext`, one of the methods proposed in that paper, overcomes `Flip Flop` [17] and `SLIQ` [16] 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 [24]), 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 [24, 12, 11, 21, 19]. Kurkoski and Yagi [12] 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 [9, 19, 21, 11], 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.

2.1 Concave Functions

We will need the following properties of one-dimensional concave functions.

Lemma 2.1. *Let $f : [0, 1] \to \mathbb{R}$ be a continuous and concave function with $f(0) = 0$. Then:*

1. *(Subadditivity) For all $x, y \geq 0$ we have $f(x + y) \leq f(x) + f(y)$.*
2. *(Sublinearity) For all $x \geq 0$ and $\alpha \geq 1$, $f(\alpha x) \leq \alpha f(x)$. In particular, for $0 < x \leq y$ we have $\frac{f(x)}{x} \geq \frac{f(y)}{y}$ (just set $\alpha = \frac{y}{x}$)*
3. *If $f(1) = 0$, then for all $0 \leq x \leq y < 1$*
$$\frac{f(x)}{1-x} \leq \frac{f(y)}{1-y}.$$

Proof. The definition of concave functions assures
$$\frac{x}{x+y} f(x+y) + \frac{y}{x+y} f(0) = \frac{x}{x+y} f(x+y) \leq f(x)$$
and
$$\frac{y}{x+y} f(x+y) + \frac{x}{x+y} f(0) = \frac{y}{x+y} f(x+y) \leq f(y).$$
By adding the inequalities we establish the first item. The second item follows because the definition of concave functions assures that $f(x) \geq (1/\alpha)f(\alpha x) + (1 - 1/\alpha)f(0) = (1/\alpha)f(\alpha x)$. For the third one, we can apply the result given by the second item to the function $g(x) = f(1 - x)$ (which is concave and has $g(0) = 0$), obtaining that $\frac{g(x)}{x} = \frac{f(1-x)}{x}$ is non-increasing in $(0, 1]$, or equivalently, that $\frac{f(x)}{1-x}$ is non-decreasing in (0,1]. This concludes the proof. ☐

The following inequality will be useful to simplify our calculations.

Corollary 2.2. *Let $f : [0, 1] \to \mathbb{R}$ be a concave function with $f(0) = f(1) = 0$ and let x_n, x_d, y_n, y_d be such that $0 \leq x_n < x_d$ and $0 < y_n < y_d$. Then, we have*
$$x_d f(x_n/x_d) \leq C\, y_d f(y_n/y_d),$$
where
$$C = \max\left\{\frac{x_d - x_n}{y_d - y_n}, \frac{x_n}{y_n}\right\}.$$

Proof. If $x_n/x_d \leq y_n/y_d$ we use item 3 of Lemma 2.1, otherwise we use item 2. ☐

2.2 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 $0 < p \leq q \leq 1$

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

Impurity measures satisfying the conditions (P0)-(P2) are called *frequency-weighted impurity measures with concave functions* [6]. These impurities measures are superadditive, as shown in [6].

Lemma 2.3 (Lemma 1 in [6])**.** *If I satisfies (P0)-(P2) then for every vectors $\mathbf{u}_L$ and $\mathbf{u}_R$ in $\mathbb{R}_+^k$, we have $I(\mathbf{u}_L + \mathbf{u}_R) \geq I(\mathbf{u}_L) + I(\mathbf{u}_R)$.*

Although property (P3) is not particularly intuitive it can be shown that if a simple constraint ($xf''(x)$ is non-increasing in the interval [0,1]) is imposed on the second derivative f'' of f then (P3) is also satisfied. This result can be found in the appendix. Another interesting observation is that $f(x) = x(\ln^2(x) - 2\ln(x))$ is an example of a function that satisfies both (P1) and (P2) but it does not satisfy (P3).

The next lemma shows that both Gini and Entropy belong to the class $\mathcal{C}$.

Lemma 2.4. *The Gini measure I_{Gini} and the Entropy measure I_{Entr} belong to $\mathcal{C}$.*

Proof. The measure I_{Gini} is obtained using the function $f_{Gini}(x) = x(1-x)$, and I_{Entr} is obtained using the function $f_{Entr}(x) = x \log \frac{1}{x}$. Clearly both functions satisfy property (P1), and it is known they also satisfy (P2) [6]. So it remains to be shown that they satisfy property (P3).

For f_{Gini}, (P3) becomes

$$p(1-p) \leq p(1-q) + p\left(1 - \frac{p}{q}\right) \qquad \forall q \in [p, 1]$$

which after canceling the p's out and rearranging, is equivalent to $p \geq q + \frac{p}{q} - 1$ for all $q \in [p, 1]$, or $p \geq \max_{q \in [p,1]}(q + \frac{p}{q} - 1)$. But the function in the max is convex in q, and hence its maximum is attained at one of the endpoints $q = p$ and $q = 1$; for these endpoints the inequality holds at equality, which then proves the desired property.

For f_{Entr}, a simple inspection shows that (P3) holds at equality. $\square$

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

$$I(\mathbf{u}) \leq I\Big((\mathbf{u}\cdot\mathbf{d}, \mathbf{u}\cdot(\mathbf{1}-\mathbf{d}))\Big) + I(\mathbf{u}\circ\mathbf{d}) + I(\mathbf{u}\circ(\mathbf{1}-\mathbf{d})) \tag{2.1}$$

Proof. When $\mathbf{u}\cdot\mathbf{d} = 0$ or $\mathbf{u}\cdot(\mathbf{1}-\mathbf{d}) = 0$, the result trivially holds. Thus, we assume that both $\mathbf{u}\cdot\mathbf{d} \neq 0$ and $\mathbf{u}\cdot(\mathbf{1}-\mathbf{d}) \neq 0$. Note that by the definition of I, inequality (2.1) is invariant to scaling $\mathbf{u}$; thus, to simplify the notation, we assume without loss of generality that $\|\mathbf{u}\|_1 = 1$. The left-hand side of inequality (2.1) is $\sum_i f(u_i)$, and the first term in the right-hand side is

$$f(\mathbf{d}\cdot\mathbf{u}) + f((\mathbf{1}-\mathbf{d})\cdot\mathbf{u}) = \sum_{i=1}^{k}\left(\frac{u_i d_i}{\mathbf{d}\cdot\mathbf{u}}\right) f(\mathbf{d}\cdot\mathbf{u}) + \sum_{i=1}^{k}\left(\frac{u_i(1-d_i)}{(\mathbf{1}-\mathbf{d})\cdot\mathbf{u}}\right) f((\mathbf{1}-\mathbf{d})\cdot\mathbf{u}).$$

Thus, we need to prove

$$\sum_{i=1}^{k} f(u_i) \leq \sum_{i=1}^{k}\left(\frac{u_i d_i}{\mathbf{d}\cdot\mathbf{u}}\right) f(\mathbf{d}\cdot\mathbf{u}) + (\mathbf{d}\cdot\mathbf{u})\sum_{i=1}^{k} f\left(\frac{u_i d_i}{\mathbf{d}\cdot\mathbf{u}}\right) +$$
$$\sum_{i=1}^{k}\left(\frac{u_i(1-d_i)}{(\mathbf{1}-\mathbf{d})\cdot\mathbf{u}}\right) f((\mathbf{1}-\mathbf{d})\cdot\mathbf{u}) + ((\mathbf{1}-\mathbf{d})\cdot\mathbf{u})\sum_{i=1}^{k} f\left(\frac{u_i(1-d_i)}{(\mathbf{1}-\mathbf{d})\cdot\mathbf{u}}\right)$$

Employing Property (P3) of f, with $p = d_i u_i$ and $q = \mathbf{d}\cdot\mathbf{u}$, we get

$$f(d_i u_i) \leq \frac{d_i u_i}{\mathbf{d}\cdot\mathbf{u}} f(\mathbf{d}\cdot\mathbf{u}) + (\mathbf{d}\cdot\mathbf{u}) f\left(\frac{d_i u_i}{\mathbf{d}\cdot\mathbf{u}}\right)$$

Moreover, using property (P3) with $p = (1-d_i)u_i$ and $q = (\mathbf{1}-\mathbf{d})\cdot\mathbf{u}$, we get

$$f(u_i(1-d_i)) \leq \left(\frac{(1-d_i)u_i}{(\mathbf{1}-\mathbf{d})\cdot\mathbf{u}}\right) f((\mathbf{1}-\mathbf{d})\cdot\mathbf{u}) + ((\mathbf{1}-\mathbf{d})\cdot\mathbf{u}) f\left(\frac{(1-d_i)u_i}{(\mathbf{1}-\mathbf{d})\cdot\mathbf{u}}\right)$$

The subadditivity of f implies $f(u_i) \leq f(d_i u_i) + f((1-d_i)u_i)$. Thus, by adding the previous inequalities we obtain

$$f(u_i) \leq f(u_i d_i) + f(u_i(1-d_i)) \leq \frac{d_i u_i}{\mathbf{d}\cdot\mathbf{u}} f(\mathbf{d}\cdot\mathbf{u}) + (\mathbf{d}\cdot\mathbf{u}) f\left(\frac{d_i u_i}{\mathbf{d}\cdot\mathbf{u}}\right) +$$
$$\frac{(1-d_i)u_i}{(\mathbf{1}-\mathbf{d})\cdot\mathbf{u}} f((\mathbf{1}-\mathbf{d})\cdot\mathbf{u}) + ((\mathbf{1}-\mathbf{d})\cdot\mathbf{u}) f\left(\frac{(1-d_i)u_i}{(\mathbf{1}-\mathbf{d})\cdot\mathbf{u}}\right).$$

Adding the previous inequality over all i's gives the result. □

2.3 Necessary conditions for optimal partitions

We end this section presenting two results that give necessary conditions for optimal partitions. The first one, proved in [2], 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 [3, 6]. Both results are stated using our notation.

Theorem 2.6 (Theorem 4.5 of [2])**.** *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 instance (V_2, I).*

Lemma 2.7 (Hyperplanes Lemma [3, 6])**.** *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 2.7) 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^*.

Proof. Let $\alpha = \max_i |d_i^*|$. Define

$$\mathbf{d} = \frac{\mathbf{1} - (\mathbf{d}^*/\alpha)}{\|\mathbf{1} - (\mathbf{d}^*/\alpha)\|_\infty}$$

and

$$C = \frac{1}{\|\mathbf{1} - (\mathbf{d}^*/\alpha)\|_\infty}$$

To verify that $\mathbf{d} \in [0,1]^k$, it is enough to observe that $\mathbf{d}^*/\alpha$ lies in $[-1,1]^k$ and $(\mathbf{1} - (\mathbf{d}^*/\alpha))$ lies in $\mathbb{R}_+^k$.

If $\mathbf{d}^* \cdot \pi(\mathbf{u}) > 0$ then

$$\mathbf{d} \cdot \pi(\mathbf{u}) = \frac{(\mathbf{1} - (\mathbf{d}^*/\alpha)) \cdot \pi(\mathbf{u})}{\|\mathbf{1} - (\mathbf{d}^*/\alpha))\|_\infty} = \frac{1 - (\mathbf{d}^*\pi(\mathbf{u}))/\alpha}{\|\mathbf{1} - (\mathbf{d}^*/\alpha)\|_\infty} < \frac{1}{\|\mathbf{1} - (\mathbf{d}^*/\alpha)\|_\infty} = C$$

On the other hand, if $\mathbf{d}^* \cdot \pi(\mathbf{u}) < 0$ then

$$\mathbf{d} \cdot \pi(\mathbf{u}) = \frac{(\mathbf{1} - (\mathbf{d}^*/\alpha)) \cdot \pi(\mathbf{u})}{\|\mathbf{1} - (\mathbf{d}^*/\alpha))\|_\infty} = \frac{1 - (\mathbf{d}^*\pi(\mathbf{u}))/\alpha}{\|\mathbf{1} - (\mathbf{d}^*/\alpha)\|_\infty} > \frac{1}{\|\mathbf{1} - (\mathbf{d}^*/\alpha)\|_\infty} = C$$

□

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)

```
1: For each $\mathbf{v}$ in $V$ let $r(\mathbf{v}) = (\mathbf{d} \cdot \mathbf{v})/\|\mathbf{v}\|_1$
2: Rank the vectors in $V$ according to $r(\mathbf{v})$
3: for $j = 1, \ldots, n-1$ do
4:     $P_j \leftarrow$ subset of $V$ containing the $j$ vectors with the largest value of $r(\cdot)$
5:     Evaluate the impurity of partition $(P_j, V \setminus P_j)$
6: end for
7: 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*

$$\frac{I(\mathcal{B}_{\mathbf{d}})}{\mathsf{opt}(V)} \leq 1 + \frac{I(\mathbf{u} \circ \mathbf{d}) + I(\mathbf{u} \circ (\mathbf{1} - \mathbf{d}))}{\min_{\mathbf{d}' \in \{0,1\}^k} \{I(\mathbf{u} \circ \mathbf{d}') + I(\mathbf{u} \circ (\mathbf{1} - \mathbf{d}'))\}} \tag{3.2}$$

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 $\mathit{collapse}_{\mathbf{d}} : \mathbb{R}^k_+ \to \mathbb{R}^2_+$ that maps $\mathbf{v} \mapsto (\mathbf{v} \cdot \mathbf{d}, \mathbf{v} \cdot (1 - \mathbf{d}))$. Moreover, for a set of vectors S, define $\mathit{collapse}_{\mathbf{d}}(S)$ as the set obtained applying $\mathit{collapse}_{\mathbf{d}}()$ 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 $(\mathit{collapse}_{\mathbf{d}}(V), I)$.

The main motivation for looking at 2-class instances is that we know from Theorem 2.6 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 1 of $\mathcal{B}_{\mathbf{d}}$ so that the impurity of the binary partition $(\mathit{collapse}_{\mathbf{d}}(P_j), \mathit{collapse}_{\mathbf{d}}(V \setminus P_j))$ 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) $I(\mathcal{B}_{\mathbf{d}}) \leq I(\mathcal{A}_{\mathbf{d}})$ and (ii) $(\mathit{collapse}_{\mathbf{d}}(L), \mathit{collapse}_{\mathbf{d}}(R))$ is an optimal partition for the 2-class instance $(\mathit{collapse}_{\mathbf{d}}(V), I)$.*

Proof. The first item follows because both algorithms consider the same partitions and $\mathcal{B}_{\mathbf{d}}$ returns the one with minimum impurity.

The second item follows from Theorem 2.6 applied to the instance $(\mathit{collapse}_{\mathbf{d}}(V), I)$ because: (i) the ranking function $r(\mathbf{v}) = (\mathbf{d} \cdot \mathbf{v})/\|\mathbf{v}\|_1 = \mathit{collapse}_{\mathbf{d}}(\mathbf{v})_1/\|\mathbf{v}\|_1$, used in algorithm $\mathcal{A}_{\mathbf{d}}$, is precisely the one from Theorem 2.6. Hence, the $n-1$ partitions considered in Theorem 2.6 match the collapsed version of those considered in Line 1 of $\mathcal{A}_{\mathbf{d}}$ and (ii) by construction $\mathcal{A}_{\mathbf{d}}$ returns the partition (L, R) for which its collapsed version $(\mathit{collapse}_{\mathbf{d}}(L), \mathit{collapse}_{\mathbf{d}}(R))$ has the smallest impurity. □

Given the first item of the above proposition, 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 2.5) we get

$$\begin{aligned} I(\mathcal{A}_{\mathbf{d}}) \leq{} & I\left((\mathbf{d} \cdot \mathbf{u}_L, \bar{\mathbf{d}} \cdot \mathbf{u}_L)\right) + I(\mathbf{d} \circ \mathbf{u}_L) + I(\bar{\mathbf{d}} \circ \mathbf{u}_L) \\ & + I\left((\mathbf{d} \cdot \mathbf{u}_R, \bar{\mathbf{d}} \cdot \mathbf{u}_R)\right) + I(\mathbf{d} \circ \mathbf{u}_R) + I(\bar{\mathbf{d}} \circ \mathbf{u}_R) \end{aligned} \tag{3.3}$$

$$= \mathsf{opt}(\mathit{collapse}_{\mathbf{d}}(V)) + I(\mathbf{d} \circ \mathbf{u}_L) + I(\bar{\mathbf{d}} \circ \mathbf{u}_L) + I(\mathbf{d} \circ \mathbf{u}_R) + I(\bar{\mathbf{d}} \circ \mathbf{u}_R), \tag{3.4}$$

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

Now we need to upper bound the last four terms in the RHS of the equation (3.4). Using the superadditivity of I (Lemma 2.3) we have

$$I(\mathcal{A}_{\mathbf{d}}) \le \mathsf{opt}(\mathit{collapse}_{\mathbf{d}}(V)) + I(\mathbf{d} \circ \mathbf{u}) + I(\bar{\mathbf{d}} \circ \mathbf{u}) \tag{3.5}$$

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 $\mathsf{opt}(V) \ge \mathsf{opt}(\mathit{collapse}_{\mathbf{d}}(V))$.*

Proof. For any $\mathbf{u} \in \mathbb{R}^k_+$ we have by definition

$$I(\mathbf{u}) = \|\mathbf{u}\|_1 \left(\sum_{i|d_i=1}^{k} f\left(\frac{d_i u_i}{\|\mathbf{u}\|_1}\right) + \sum_{i|d_i=0}^{k} f\left(\frac{(1-d_i)u_i}{\|\mathbf{u}\|_1}\right) \right) \overset{\text{subadd.}}{\ge}$$
$$\|\mathbf{u}\|_1 \left(f\left(\frac{\mathbf{u}\cdot\mathbf{d}}{\|\mathbf{u}\|_1}\right) + f\left(\frac{\mathbf{u}\cdot\bar{\mathbf{d}}}{\|\mathbf{u}\|_1}\right) \right) = I(\mathit{collapse}_{\mathbf{d}}(\mathbf{u})), \tag{3.6}$$

where the last identity follows from the definition of $\mathit{collapse}_{\mathbf{d}}(\mathbf{u})$ and the fact $\|\mathit{collapse}_{\mathbf{d}}(\mathbf{u})\|_1 = \|\mathbf{u}\|_1$.

Let (L^*, R^*) be an optimal partition of V. Moreover, let $\mathbf{u}_{L^*} = \sum_{\mathbf{v}\in L^*} \mathbf{v}$ and $\mathbf{u}_{R^*} = \sum_{\mathbf{v}\in R^*} \mathbf{v}$. Thus,

$$\mathsf{opt}(V) = I(\mathbf{u}_{L^*}) + I(\mathbf{u}_{R^*}) \ge I(\mathit{collapse}_{\mathbf{d}}(\mathbf{u}_{L^*})) + I(\mathit{collapse}_{\mathbf{d}}(\mathbf{u}_{R^*})) \ge \mathsf{opt}(\mathit{collapse}_{\mathbf{d}}(V)),$$

where the first inequality follows from inequality (3.6) and the last inequality follows because $\mathit{collapse}_{\mathbf{d}}(L^*)$ and $\mathit{collapse}_{\mathbf{d}}(R^*))$ form a partition for $\mathit{collapse}_{\mathbf{d}}(V)$. □

For our second lower bound, we consider the relaxed problem where each example corresponds to a distinct attribute value and the 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 2.7 that, in the optimal partition, 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

$$\mathsf{opt}(V) \ge \mathsf{opt}(V') = \min_{\mathbf{d}'\in\{0,1\}^k} \left\{ I(\mathbf{u} \circ \mathbf{d}') + I(\mathbf{u} \circ (\mathbf{1} - \mathbf{d}')) \right\}. \tag{3.7}$$

Thus, by using the upper bound given by equation (3.5), the lower bound given by Lemma 3.4 and the previous inequality we obtain that the RHS of inequality (3.2) 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 is easy to obtain the following bound on the approximation of `HcC`.

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

Proof. Let $\mathbf{d}^* \in \{0,1\}^k$ be such that

$$I(\mathbf{u} \circ \mathbf{d}^*) + I(\mathbf{u} \circ (1 - \mathbf{d}^*)) = \min_{\mathbf{d} \in \{0,1\}^k} \left(I(\mathbf{u} \circ \mathbf{d}) + I(\mathbf{u} \circ (1 - \mathbf{d})) \right).$$

Moreover, let $\mathbf{d}'$ be the direction employed by `HcC`.

Then,

$$\frac{I(\mathcal{B}_{\mathbf{d}'})}{\mathsf{opt}(V)} \leq \frac{I(\mathcal{B}_{\mathbf{d}^*})}{\mathsf{opt}(V)} \leq 1 + \frac{I(\mathbf{u} \circ \mathbf{d}^*) + I(\mathbf{u} \circ (1 - \mathbf{d}^*))}{I(\mathbf{u} \circ \mathbf{d}^*) + I(\mathbf{u} \circ (1 - \mathbf{d}^*))} = 2,$$

where the last inequality follows from Theorem 3.2 □

We shall mention that `HcC` is closely related with the `Twoing` method proposed in [2]. 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 [2] 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 $\sum_{\mathbf{v} \in V} \mathbf{v}$. In what follows we assume w.l.o.g. that class 1 is the class with the largest number of examples, that is, $u_1 \geq u_i$, for $i = 2, \ldots, k$. Let $\mathbf{e}_1$ be the direction in which the first coordinate, corresponding to class 1, has value 1 and the other coordinates have value 0. The next theorem shows that algorithm $\mathcal{B}_{\mathbf{e}_1}$, denoted here by `LargestClassAlone` (`LCA` for short), has a constant approximation for our class.

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

Proof. It follows from Theorem 3.2 that

$$\frac{I(\mathtt{LCA})}{\mathsf{opt}(V)} \leq \frac{I(\mathcal{A}_{\mathbf{e}_1})}{\mathsf{opt}(V)} \leq 1 + \frac{I(\mathbf{u} \circ \mathbf{e}_1) + I(\mathbf{u} \circ (\mathbf{1} - \mathbf{e}_1))}{\min_{\mathbf{d}' \in \{0,1\}^k} \{ I(\mathbf{u} \circ \mathbf{d}') + I(\mathbf{u} \circ (\mathbf{1} - \mathbf{d}')) \}}. \tag{3.8}$$

[2] The theorem holds under the weaker assumption that the subsystem property holds rather than property (P3).

Thus, to establish the theorem it is enough to prove that, for every direction $\mathbf{d}$ in the hypercube $\{0,1\}^k$,

$$\frac{I(\mathbf{u}\circ\mathbf{e}_1)+I(\mathbf{u}\circ(\mathbf{1}-\mathbf{e}_1))}{I(\mathbf{u}\circ\mathbf{d})+I(\mathbf{u}\circ(\mathbf{1}-\mathbf{d}))}\leq(2+\sqrt{3}). \tag{3.9}$$

If either $\mathbf{d}=\mathbf{0}$ or $\mathbf{d}=\mathbf{1}$ the result follows from Lemma 2.3. Moreover, if either $\mathbf{d}=\mathbf{e}_1$ or $\mathbf{d}=\mathbf{1}-\mathbf{e}_1$ the result trivially holds. Thus, we assume $\mathbf{d}\notin\{\mathbf{0},\mathbf{1},\mathbf{e}_1,\mathbf{1}-\mathbf{e}_1\}$

In what follows, to simplify the notation we use $S=\mathbf{u}\cdot\mathbf{d}$ and $T=\mathbf{u}\cdot(\mathbf{1}-\mathbf{d})$. Moreover, we use $S'=S-u_1d_1$ and $T'=T-u_1(1-d_1)$. We can assume w.l.o.g that $S'\geq T'$ due to the symmetry of the LHS of inequality (3.9) with respect to $\mathbf{d}$. We split the proof into two cases according to whether $d_1=1$ or $d_1=0$.

Case 1 $d_1=1$. In this case $T=T'$ and $S'=S-u_1$. The upper bound is given by

$$I(\mathbf{u}\circ\mathbf{e}_1)+I(\mathbf{u}\circ(\mathbf{1}-\mathbf{e}_1))=I(\mathbf{u}\circ(\mathbf{1}-\mathbf{e}_1))=\sum_{i>1|d_i=1}(S'+T)f\left(\frac{u_i}{S'+T}\right)+\sum_{i|d_i=0}(S'+T)f\left(\frac{u_i}{S'+T}\right)\leq$$
$$\sum_{i>1|d_i=1}(S'+T)f\left(\frac{u_i}{S'+T}\right)+(S'+T)f\left(\frac{T}{S'+T}\right)+\sum_{i|d_i=0}Tf\left(\frac{u_i}{T}\right), \tag{3.10}$$

where the inequality follows from property (P3) of class of functions $f(\cdot)$, using $p=u_i/(S'+T)$ and $q=u_i/T$. Note that $T>0$ since $T=\mathbf{u}\cdot(\mathbf{1}-\mathbf{d})$ and $\mathbf{d}\neq\mathbf{1}$.

Let B be a subset of $[k]$ with the following properties: (i) $B\subseteq\{i|i>1 \text{ and } d_i=1\}$; (ii) $\sum_{i\in B}u_i\geq S'/2$; (iii) if a set B' satisfies (i) and (ii) then $\sum_{i\in B}u_i\leq\sum_{i\in B'}u_i$.

In addition, let $\bar{B}=\{i|i>1 \text{ and } d_i=1\}\setminus B$. We have that

$$I(\mathbf{u}\circ\mathbf{d})+I(\mathbf{u}\circ(\mathbf{1}-\mathbf{d}))=\sum_{i|d_i=1}(S'+u_1)f\left(\frac{u_i}{S'+u_1}\right)+\sum_{i|d_i=0}T\cdot f\left(\frac{u_i}{T}\right)>$$
$$\frac{2}{3}\sum_{i>1|d_i=1}(S'+u_1)f\left(\frac{u_i}{S'+u_1}\right)+\frac{1}{3}(S'+u_1)\sum_{i\in B\cup\bar{B}\cup\{1\}}f\left(\frac{u_i}{S'+u_1}\right)+\sum_{i|d_i=0}T\cdot f\left(\frac{u_i}{T}\right)\geq$$
$$\frac{2}{3}\sum_{i>1|d_i=1}(S'+u_1)f\left(\frac{u_i}{S'+u_1}\right)+\frac{1}{3}(S'+u_1)\left(f\left(\frac{\sum_{i\in B}u_i}{S'+u_1}\right)+f\left(\frac{u_1+\sum_{i\in\bar{B}}u_i}{S'+u_1}\right)\right)+\sum_{i|d_i=0}T\cdot f\left(\frac{u_i}{T}\right)$$

where the last inequality follows from the subadditivity of f.

Thus, putting together (3.10) and the previous inequality, we conclude that the left term of (3.9) is at most

$$\max\left\{\frac{\sum_{i>1|d_i=1}(S'+T)f\left(\frac{u_i}{S'+T}\right)}{\frac{2}{3}\sum_{i>1|d_i=1}(S'+u_1)f\left(\frac{u_i}{S'+u_1}\right)},\frac{(S'+T)f\left(\frac{T}{S'+T}\right)}{\frac{1}{3}(S'+u_1)\left(f\left(\frac{\sum_{i\in B}u_i}{S'+u_1}\right)+f\left(\frac{u_1+\sum_{i\in\bar{B}}u_i}{S'+u_1}\right)\right)},\frac{\sum_{i|d_i=0}T\cdot f\left(\frac{u_i}{T}\right)}{\sum_{i|d_i=0}T\cdot f\left(\frac{u_i}{T}\right)}\right\}\leq 3,$$

where the inequality can be proved by applying Corollary 2.2 to the 2 first terms in the max expression above. In fact, for the first term we have

$$\frac{\sum_{i>1|d_i=1}(S'+T)f\left(\frac{u_i}{S'+T}\right)}{\frac{2}{3}\sum_{i>1|d_i=1}(S'+u_1)f\left(\frac{u_i}{S'+u_1}\right)}\leq\frac{3}{2}\max_{i>1|d_i=1}\left\{\frac{S'+T-u_i}{S'+u_1-u_i}\right\}\leq\frac{3}{2}\left(\frac{S'+T}{S'}\right)\leq 3,$$

where the second inequality follows because $u_1 \geq u_i$ and the last one because $S' \geq T$. For the second term we have

$$\frac{(S'+T)f\left(\frac{T}{S'+T}\right)}{\frac{1}{3}(S'+u_1)\left(f\left(\frac{\sum_{i\in B}u_i}{S'+u_1}\right)+f\left(\frac{u_1+\sum_{i\in\bar{B}}u_i}{S'+u_1}\right)\right)} = \frac{\frac{1}{2}(S'+T)f\left(\frac{T}{S'+T}\right)+\frac{1}{2}(S'+T)f\left(\frac{T}{S'+T}\right)}{\frac{1}{3}(S'+u_1)\left(f\left(\frac{\sum_{i\in B}u_i}{S'+u_1}\right)+f\left(\frac{u_1+\sum_{i\in\bar{B}}u_i}{S'+u_1}\right)\right)} \leq$$

$$\frac{3}{2}\max\left\{\frac{T}{\sum_{i\in B}u_i}, \frac{S'}{S'+u_1-\sum_{i\in B}u_i}, \frac{T}{u_1+\sum_{i\in\bar{B}}u_i}, \frac{S'}{S'-\sum_{i\in\bar{B}}u_i}\right\} =$$
$$\frac{3}{2}\max\left\{\frac{T}{\sum_{i\in B}u_i}, \frac{S'}{S'+u_1-\sum_{i\in B}u_i}, \frac{T}{S'+u_1-\sum_{i\in B}u_i}, \frac{S'}{\sum_{i\in B}u_i}\right\} \leq 3$$

where the last inequality holds because: (a) $S' \geq T = T'$; (b) the choice of B assures that $\sum_{i\in B}u_i \geq S'/2$; (c) the minimality of B assures that $\sum_{i\in B}u_i - u_1 \leq S'/2$ so that $S'/2 \leq S' + u_1 - \sum_{i\in B}u_i$

Case 2) $d_1 = 0$.

In this case $S = S'$. Let c a constant larger than 1 that will be defined later in the analysis. In addition, let $D^- = \{i | d_i = 1 \text{ and } u_i < S/c\}$ and $D^+ = \{i | d_i = 1 \text{ and } u_i \geq S/c\}$. We handle in different ways the case where there is a large class and the case where there is not.

Subcase 2.1) $u_1 \geq S/c$

We have that

$$I(\mathbf{u}\circ\mathbf{e}_1)+I(\mathbf{u}\circ(\mathbf{1}-\mathbf{e}_1)) = \sum_{i\in D^-}(S+T')f\left(\frac{u_i}{S+T'}\right)+\sum_{i\in D^+}(S+T')f\left(\frac{u_i}{S+T'}\right)+\sum_{i>1|d_i=0}(S+T')f\left(\frac{u_i}{S+T'}\right) \leq$$

$$\sum_{i\in D^-}(S+T')f\left(\frac{u_i}{S+T'}\right)+\sum_{i\in D^+}S\cdot f\left(\frac{u_i}{S}\right)+(S+T')f\left(\frac{S}{S+T'}\right)+\sum_{i>1|d_i=0}(S+T')f\left(\frac{u_i}{S+T'}\right),$$

where the inequality follows from property (P3) of the class of functions f, using $p = u_i/(S+T')$ and $q = u_i/S$. Note that $S > 0$ since $S = \mathbf{u}\cdot\mathbf{d}$ and $\mathbf{d} \neq \mathbf{0}$.

On the other hand,

$$I(\mathbf{u}\circ\mathbf{d})+I(\mathbf{u}\circ(\mathbf{1}-\mathbf{d})) = \sum_{i\in D^-}S\cdot f\left(\frac{u_i}{S}\right)+\sum_{i\in D^+}S\cdot f\left(\frac{u_i}{S}\right)+(u_1+T')\cdot f\left(\frac{u_1}{u_1+T'}\right)+\sum_{i>1|d_i=0}(u_1+T')\cdot f\left(\frac{u_i}{u_1+T'}\right)$$

Thus, by comparing the 4 summands in the upper bound with the 4 summands in the lower bound we conclude that the left term of (3.9) is at most

$$\max\left\{\frac{\sum_{i\in D^-}(S+T')f\left(\frac{u_i}{S+T'}\right)}{\sum_{i\in D^-}S\cdot f\left(\frac{u_i}{S}\right)}, \frac{\sum_{i\in D^+}S\cdot f\left(\frac{u_i}{S}\right)}{\sum_{i\in D^+}S\cdot f\left(\frac{u_i}{S}\right)}, \frac{(S+T')f\left(\frac{S}{S+T'}\right)}{(u_1+T')\cdot f\left(\frac{u_1}{u_1+T'}\right)}, \frac{\sum_{i>1|d_i=0}(S+T')f\left(\frac{u_i}{S+T'}\right)}{\sum_{i>1|d_i=0}(u_1+T')\cdot f\left(\frac{u_i}{u_1+T'}\right)}\right\} \leq$$

$$\max\left\{\frac{2-1/c}{1-1/c}, c\right\}, \tag{3.11}$$

where the inequality can be proved by applying Corollary 2.2 to the terms in the max expression above. In fact, for the first term we have

$$\frac{\sum_{i\in D^-}(S+T')f\left(\frac{u_i}{S+T'}\right)}{\sum_{i\in D^-} S\cdot f\left(\frac{u_i}{S}\right)} \le \max_{i\in D^-}\left\{\frac{S+T'-u_i}{S-u_i}\right\} \le \frac{2S-S/c}{S-S/c} = \frac{2-1/c}{1-1/c},$$

where the last inequality holds because $u_i < S/c$ since $i \in D^-$.

The second term of the max expression is equal to 1. For the third term, we have

$$\frac{(S+T')f\left(\frac{S}{S+T'}\right)}{(u_1+T')\cdot f\left(\frac{u_1}{u_1+T'}\right)} \le \max\left\{\frac{S}{u_1},1\right\} \le c$$

Finally, for the fourth term we have

$$\frac{\sum_{i>1|d_i=0}(S+T')f\left(\frac{u_i}{S+T'}\right)}{\sum_{i>1|d_i=0}(u_1+T')\cdot f\left(\frac{u_i}{u_1+T'}\right)} \le \max_{i>1|d_i=0}\left\{\frac{S+T'-u_i}{u_1+T'-u_i}\right\} \le \max\left\{\frac{S}{u_1},1\right\} \le c$$

Subcase 2.2) $u_1 < S/c$

We have that

$$I(\mathbf{u}\circ\mathbf{e}_1)+I(\mathbf{u}\circ(\mathbf{1}-\mathbf{e}_1)) = \sum_{i|d_i=1}(S+T')f\left(\frac{u_i}{S+T'}\right)+\sum_{i>1|d_i=0}(S+T')f\left(\frac{u_i}{S+T'}\right) \le$$

$$\sum_{i|d_i=1}(S+T')f\left(\frac{u_i}{S+T'}\right)+\sum_{i>1|d_i=0}T'\cdot f\left(\frac{u_i}{T'}\right)+(S+T')f\left(\frac{T'}{S+T'}\right),$$

where the inequality follows from property (P3) of the class of functions f, using $p = u_i/(S+T')$ and $q = u_i/T'$. Note that $T' > 0$ since $T' = \mathbf{u}\cdot(\mathbf{1}-\mathbf{d}) - u_1$ and $\mathbf{d} \neq (\mathbf{1}-\mathbf{e}_1)$.

Let B be a subset of $\{j|d_j = 1\}$ for which $\left|\sum_{i\in B} u_i - S/2\right|$ is minimum. It follows that

$$S/2-(S/2c) \le \sum_{i\in B} u_i \le S/2+(S/2c). \tag{3.12}$$

Indeed, if $\sum_{i\in B} u_i > S/2+(S/2c)$, then the subset $B' = B - \{j\}$, where j is an arbitrary element in B is such that

$$|S/2| < \left|\sum_{i\in B'} u_i - S/2\right| < \left|\sum_{i\in B} u_i - S/2\right|,$$

which contradicts the minimality of B. If, $\sum_{i\in B} u_i < S/2-(S/2c)$, we can reach a contradiction via an analogous argument.

Let $\bar{B} = \{j|d_j = 1\} \setminus B$. Let α be a positive real number smaller than 1. We have that

$$I(\mathbf{u}\circ\mathbf{d})+I(\mathbf{u}\circ(\mathbf{1}-\mathbf{d})) = \alpha\sum_{i|d_i=1} S\cdot f\left(\frac{u_i}{S}\right)+\sum_{i|d_i=0}(T'+u_1)\cdot f\left(\frac{u_i}{u_1+T'}\right)+(1-\alpha)\sum_{i\in B\cup\bar{B}} S\cdot f\left(\frac{u_i}{S}\right) \ge$$

$$\alpha \sum_{i|d_i=1} S \cdot f\left(\frac{u_i}{S}\right) + \sum_{i>1|d_i=0} (T' + u_1) \cdot f\left(\frac{u_i}{u_1 + T'}\right) + (1-\alpha) S \cdot \left(f\left(\frac{\sum_{i \in B} u_i}{S}\right) + f\left(\frac{\sum_{i \in \bar{B}} u_i}{S}\right)\right),$$

where the inequality follows from the subadditivity of $f()$.

Thus, the left term of (3.9) is at most

$$\max\left\{ \frac{\sum_{i|d_i=1}(S+T')f\left(\frac{u_i}{S+T'}\right)}{\alpha \sum_{i|d_i=1} S \cdot f\left(\frac{u_i}{S}\right)}, \frac{\sum_{i>1|d_i=0} T' \cdot f\left(\frac{u_i}{T'}\right)}{\sum_{i>1|d_i=0}(T'+u_1) \cdot f\left(\frac{u_i}{u_1+T'}\right)}, \frac{(S+T')f\left(\frac{T'}{S+T'}\right)}{(1-\alpha)S \cdot \left(f\left(\frac{\sum_{i\in B} u_i}{S}\right) + f\left(\frac{\sum_{i \in \bar{B}} u_i}{S}\right)\right)} \right\} \leq$$

$$\max\left\{\frac{2-1/c}{(1-1/c)\alpha}, \frac{c}{(c-1)(1-\alpha)}\right\}, \tag{3.13}$$

where, again, the inequality can be proved by applying Corollary 2.2 to the terms in the max expression above. In fact, for the first term of max we have

$$\frac{\sum_{i|d_i=1}(S+T')f\left(\frac{u_i}{S+T'}\right)}{\alpha \sum_{i|d_i=1} S \cdot f\left(\frac{u_i}{S}\right)} \leq \left(\frac{1}{\alpha}\right) \max_{i|d_i=1}\left\{\frac{S+T'-u_i}{S-u_i}\right\} \leq \frac{2-1/c}{(1-1/c)\alpha},$$

where the last inequality holds because $T' \leq S$ and $u_i \leq S/c$.

For the second term we have

$$\frac{\sum_{i>1|d_i=0} T' \cdot f\left(\frac{u_i}{T'}\right)}{\sum_{i>1|d_i=0}(T'+u_1) \cdot f\left(\frac{u_i}{u_1+T'}\right)} \leq \max\left\{1, \frac{T'-u_i}{T'+u_1-u_i}\right\} = 1$$

Finally, for the last term we have

$$\frac{(S+T')f\left(\frac{T'}{S+T'}\right)}{(1-\alpha)S \cdot \left(f\left(\frac{\sum_{i\in B} u_i}{S}\right) + f\left(\frac{\sum_{i\in\bar{B}} u_i}{S}\right)\right)} = \frac{\frac{S+T'}{2} f\left(\frac{T'}{S+T'}\right) + \frac{S+T'}{2} f\left(\frac{T'}{S+T'}\right)}{(1-\alpha)S \cdot \left(f\left(\frac{\sum_{i\in B} u_i}{S}\right) + f\left(\frac{\sum_{i\in\bar{B}} u_i}{S}\right)\right)} \leq \tag{3.14}$$

$$\frac{1}{2(1-\alpha)} \max\left\{\frac{T'}{\sum_{i\in B} u_i}, \frac{S}{S - \sum_{i\in B} u_i}\right\} \leq \frac{c}{(c-1)(1-\alpha)}, \tag{3.15}$$

where the last inequality is a consequence of $S \geq T'$ and inequalities 3.12.

The ratio $3+\sqrt{3}$ is obtained by considering inequalities (3.11) and (3.13), setting $c = 2+\sqrt{3}$ and $\alpha = (3-\sqrt{3})/2$. $\square$

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 $I = I_{Gini}$, $\mathbf{e}_1$ 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 (3.2).

Lemma 4.1. *The direction* $\mathbf{e}_1$ *satisfies*

$$I_{Gini}(\mathbf{u} \circ \mathbf{e}_1) + I_{Gini}(\mathbf{u} \circ (\mathbf{1} - \mathbf{e}_1)) = \min_{\mathbf{d} \in \{0,1\}^k} \{I_{Gini}(\mathbf{u} \circ \mathbf{d}) + I_{Gini}(\mathbf{u} \circ (\mathbf{1} - \mathbf{d}))\}.$$

Proof. We have to prove that

$$I_{Gini}(\mathbf{u} \circ \mathbf{e}_1) + I_{Gini}(\mathbf{u} \circ (\mathbf{1} - \mathbf{e}_1)) \leq I_{Gini}(\mathbf{u} \circ \mathbf{d}) + I_{Gini}(\mathbf{u} \circ (\mathbf{1} - \mathbf{d})),$$

for every $\mathbf{d} \in \{0,1\}^k$. For $\mathbf{d} = \mathbf{0}$ and $\mathbf{d} = \mathbf{1}$ the inequality follows from Lemma 2.3. Thus, we assume that both $\mathbf{d} \neq \mathbf{0}$ and $\mathbf{d} \neq \mathbf{1}$.

It follows from the definition of $I_{Gini}(\cdot)$ that

$$I_{Gini}(\mathbf{u}\circ\mathbf{d})+I_{Gini}(\mathbf{u}\circ(\mathbf{1}-\mathbf{d})) = (\mathbf{u}\cdot\mathbf{d}) \left(\frac{(\mathbf{u} \cdot \mathbf{d})^2 - \sum_{i|d_i=1}(u_i)^2}{(\mathbf{u}\mathbf{d})^2} \right) + (\mathbf{u}(\mathbf{1}-\mathbf{d})) \left(\frac{(\mathbf{u}(\mathbf{1} - \mathbf{d}))^2 - \sum_{i|d_i=0}(u_i)^2}{(\mathbf{u}(\mathbf{1} - \mathbf{d}))^2} \right) =$$

$$\|\mathbf{u}\|_1 - \left(\frac{\sum_{i|d_i=1}(u_i)^2}{\mathbf{u} \cdot \mathbf{d}} \right) - \left(\frac{\sum_{i|d_i=0}(u_i)^2}{\mathbf{u}(\mathbf{1} - \mathbf{d})} \right)$$

Define $g(\mathbf{d})$ as the sum of two last terms of the above expression, that is,

$$g(\mathbf{d}) = \left(\frac{\sum_{i|\mathbf{d}_i=1}(u_i)^2}{\mathbf{u} \cdot \mathbf{d}} \right) + \left(\frac{\sum_{i|\mathbf{d}_i=0}(u_i)^2}{\mathbf{u}(\mathbf{1} - \mathbf{d})} \right)$$

It is enough to prove that $g(\mathbf{e}_1) \geq g(\mathbf{d})$ for an arbitrary $\mathbf{d}$. For that, we assume w.l.o.g. that $d_1 = 1$ due to the symmetry of $g(\mathbf{d})$ with respect to $\mathbf{d}$.

Let

$$\alpha = \frac{\sum_{i>1|d_i=1}(u_i)^2}{\sum_{i>1|d_i=1} u_i} \qquad \text{and} \qquad \beta = \frac{\sum_{i|d_i=0}(u_i)^2}{\sum_{i|d_i=0} u_i}$$

Thus,

$$g(\mathbf{d}) = \frac{(u_1)^2 + \alpha(\mathbf{u} \cdot \mathbf{d} - u_1)}{u_1 + (\mathbf{u} \cdot \mathbf{d} - u_1)} + \beta$$

Moreover, we can write $g(\mathbf{e}_1)$ as a function of $\mathbf{d}$

$$g(\mathbf{e}_1) = u_1 + \frac{\alpha(\mathbf{u} \cdot \mathbf{d} - u_1) + \beta\mathbf{u}(\mathbf{1} - \mathbf{d})}{(\mathbf{u} \cdot \mathbf{d} - u_1) + \mathbf{u}(\mathbf{1} - \mathbf{d})}$$

The following inequalities will be useful: $\alpha, \beta \leq u_1$ since $u_1 \geq u_i$ for all i, $(\mathbf{u} \cdot \mathbf{d} - u_1) \geq \alpha$ and $\mathbf{u}(\mathbf{1} - \mathbf{d}) \geq \beta$.

We need to prove that

$$g(\mathbf{e}_1) = u_1 + \frac{\alpha(\mathbf{u} \cdot \mathbf{d} - u_1) + \beta(\mathbf{u}(\mathbf{1} - \mathbf{d}))}{(\mathbf{u} \cdot \mathbf{d} - u_1) + \mathbf{u}(\mathbf{1} - \mathbf{d})} \geq \frac{(u_1)^2 + \alpha(\mathbf{u} \cdot \mathbf{d} - u_1)}{u_1 + (\mathbf{u} \cdot \mathbf{d} - u_1)} + \beta = g(\mathbf{d}),$$

or equivalently,

$$\frac{u_1(\mathbf{u}\cdot\mathbf{d}-u_1)}{u_1+(\mathbf{u}\cdot\mathbf{d}-u_1)}-\frac{\alpha(\mathbf{u}\cdot\mathbf{d}-u_1)}{u_1+(\mathbf{u}\cdot\mathbf{d}-u_1)}\geq\frac{\beta(\mathbf{u}\cdot\mathbf{d}-u_1)}{\mathbf{u}\cdot(\mathbf{1}-\mathbf{d})+(\mathbf{u}\cdot\mathbf{d}-u_1)}-\frac{\alpha(\mathbf{u}\cdot\mathbf{d}-u_1)}{\mathbf{u}\cdot(\mathbf{1}-\mathbf{d})+(\mathbf{u}\cdot\mathbf{d}-u_1)}$$

Simplifying the terms we need to prove

$$(\beta-\alpha)[(\mathbf{u}\cdot\mathbf{d}-u_1)+u_1]\leq(u_1-\alpha)[(\mathbf{u}\cdot\mathbf{d}-u_1)+\mathbf{u}\cdot(\mathbf{1}-\mathbf{d})]$$

which is equivalent to

$$\beta u_1-\alpha u_1\leq(u_1-\beta)(\mathbf{u}\cdot\mathbf{d}-u_1)+(u_1-\alpha)\mathbf{u}\cdot(\mathbf{1}-\mathbf{d}), \tag{4.16}$$

However, because $\alpha,\beta\leq u_1$, $(\mathbf{u}\cdot\mathbf{d}-u_1)\geq\alpha$ and $\mathbf{u}\cdot(\mathbf{1}-\mathbf{d})\geq\beta$, we have

$$(u_1-\beta)\alpha+(u_1-\alpha)\beta\leq(u_1-\beta)(\mathbf{u}\cdot\mathbf{d}-u_1)+(u_1-\alpha)\mathbf{u}\cdot(\mathbf{1}-\mathbf{d}).$$

Thus, to establish inequality (4.16), it is enough to prove that

$$\beta u_1-\alpha u_1\leq(u_1-\beta)\alpha+(u_1-\alpha)\beta,$$

or, equivalently,

$$\alpha\beta\leq\alpha u_1.$$

The last inequality holds because $u_1\geq\beta$. □

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 following example shows that this is the case for algorithm $\mathcal{A}_{\mathbf{e}_1}$.

Let $x>c>0$. We consider the instance $V=\{(x,0,0),(0,x,0),(c,0,c)\}$. Algorithm $\mathcal{A}_{\mathbf{e}_1}$ solves the 2-class problem for $collapse_1(V)=\{(x,0),(0,x),(c,c)\}$. One optimal solution for this problem is the partition $L=\{(x,0)\}$ and $R=\{(0,x),(c,c)\}$. By evaluating the corresponding non-collapsed partition $\widetilde{L}=\{(x,0,0)\}$ and $\widetilde{R}=\{(0,x,0),(c,0,c)\}$ for the original problem with 3 classes we obtain

$$I_{Gini}(\widetilde{L})+I_{Gini}(\widetilde{R})=\frac{4xc+2c^2}{x+2c}$$

On the other hand, consider the partition (L^*,R^*) of V, where $L^*=\{(x,0,0),(c,0,c)\}$ and $R^*=\{(0,x,0)\}$. We have

$$I_{Gini}(L^*)+I_{Gini}(R^*)=\frac{2xc+2c^2}{x+2c}$$

Thus, the ratio between the cost of the partition provided by $\mathcal{A}_{\mathbf{e}_1}$ and the partition (L^*,R^*) is $(2x+c)/(x+c)$, which goes to 2 when x/c goes to ∞.

For `LCA` (which is $\mathcal{B}_{\mathbf{e}_1}$) 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 shows that the most balanced direction with respect to $\mathbf{u}$ is the one that minimizes the the denominator in the upper bound on the approximation ratio given by inequality 3.2.

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

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

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

Proof. We have that

$$I_{Ent}(\mathbf{u}\circ\mathbf{d}) + I_{Ent}(\mathbf{u}\circ(\mathbf{1}-\mathbf{d})) = \mathbf{u}\cdot\mathbf{d}\log(\mathbf{u}\cdot\mathbf{d}) + \mathbf{u}\cdot(\mathbf{1}-\mathbf{d})\log(\mathbf{u}\cdot(\mathbf{1}-\mathbf{d})) + \sum_{i=1}^{k} u_i\log(1/u_i) =$$

$$\mathbf{u}\cdot\mathbf{d}\log(\mathbf{u}\cdot\mathbf{d}) + (\|\mathbf{u}\|_1 - \mathbf{u}\cdot\mathbf{d})\log(\|\mathbf{u}\|_1 - \mathbf{u}\cdot\mathbf{d}) + \sum_{i=1}^{k} u_i\log(1/u_i)$$

Let $p = \mathbf{u}\cdot\mathbf{d}/\|\mathbf{u}\|_1$. The above expression can be rewritten as

$$\|\mathbf{u}\|_1(p\log p + (1-p)\log(1-p)) + \|\mathbf{u}\|_1\log\|\mathbf{u}\|_1 + \sum_{i=1}^{k} u_i\log(1/u_i)$$

The result follows because the above expression as a function of p is unimodal and symmetric around $p = 1/2$, where it achieves its minimum value. □

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 [7].

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.*

The next theorem shows that the approximation of `LCA` is at most 3.

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

Proof. It follows from Theorem 3.2 and Lemma 4.3 that

$$\frac{I_{Ent}(\texttt{LCA})}{\mathsf{opt}(V)} \le 1 + \frac{I_{Ent}(\mathbf{u}\circ\mathbf{e}_1) + I_{Ent}(\mathbf{u}\circ(\mathbf{1}-\mathbf{e}_1))}{I_{Ent}(\mathbf{u}\circ\mathbf{d}^*) + I_{Ent}(\mathbf{u}\circ(\mathbf{1}-\mathbf{d}^*))} \le 1 + \frac{I_{Ent}(\mathbf{u}\circ(\mathbf{1}-\mathbf{e}_1))}{I_{Ent}(\mathbf{u}\circ\mathbf{d}^*) + I_{Ent}(\mathbf{u}\circ(\mathbf{1}-\mathbf{d}^*))}, \quad (4.17)$$

where $\mathbf{d}^*$ is the most balanced direction in $\{0,1\}^k$ with respect to vector $\mathbf{u}$.

We can assume w.l.o.g. that $d_1^* = 0$ due to the symmetry between $\mathbf{d}^*$ and $(\mathbf{1}-\mathbf{d}^*)$ in the above equation. We also assume $\mathbf{d}^* \neq (\mathbf{1}-\mathbf{e}_1)$ for otherwise `LCA` has approximation ratio equals to 2. This last assumption implies that $d_i^* = 0$ for some $i > 1$. In addition, we have $d_i^* = 1$ for some i for otherwise $\mathbf{d}^*$ is not the most balanced direction in $\{0,1\}^k$.

To simplify the notation let $S = \mathbf{u} \cdot \mathbf{d}^*$, $T = \mathbf{u} \cdot (\mathbf{1} - \mathbf{d}^*)$ and $T' = T - u_1$. Hence, we have

$$\frac{I_{Ent}(\texttt{LCA})}{\mathsf{opt}(V)} \leq 1 + \frac{\sum_{i=2}^{k} u_i \log\left(\frac{S+T'}{u_i}\right)}{\sum_{i|d_i^*=1} u_i \log\left(\frac{S}{u_i}\right) + \sum_{i|d_i^*=0} u_i \log\left(\frac{T'+u_1}{u_i}\right)}$$

$$= 1 + \frac{\sum_{i|d_i^*=1} u_i \log\left(\frac{S+T'}{u_i}\right) + \sum_{i>1|d_i^*=0} u_i \log\left(\frac{S+T'}{u_i}\right)}{\sum_{i|d_i^*=1} u_i \log\left(\frac{S}{u_i}\right) + \sum_{i|d_i^*=0} u_i \log\left(\frac{T'+u_1}{u_i}\right)} \tag{4.18}$$

The following properties will be useful for the analysis:

(i) $S \geq T'$;

(ii) $S \geq u_1$;

(iii) $S \leq T' + 2u_1$.

If property (i) does not hold we would have a direction more balanced than $\mathbf{d}^*$ by setting $d_1^* = 1$. Similarly, if property (ii) does not hold we could obtain a direction more balanced than $\mathbf{d}^*$ by setting $d_i^* = 1$ for some $i > 1$ such that $d_i^* = 0$. To see that property (iii) holds, let i with $d_i^* = 1$. We must have $S - u_i \leq T' + u_1$ for otherwise we could obtain a direction more balanced than $\mathbf{d}^*$ by setting $d_i^* = 0$. Thus, $S \leq T' + u_1 + u_i \leq T' + 2u_1$.

To obtain a bound on the rightmost side of inequality (4.18), we analyze separately each $i > 1$. For any i with $d_i^* = 1$ and $u_i \leq S/2$, we have that

$$\frac{u_i \log(\frac{S+T'}{u_i})}{u_i \log(\frac{S}{u_i})} \leq \frac{u_i \log(\frac{2S}{u_i})}{u_i \log(\frac{S}{u_i})} = \frac{u_i \left(1 + \log(\frac{S}{u_i})\right)}{u_i \log(\frac{S}{u_i})} \leq 2,$$

where the first inequality follows form property (i) and the last inequality holds because $\log(S/u_i) > 1$ since $u_i \leq S/2$.

For i with $d_i^* = 1$ and $u_i > S/2$ we have that

$$\frac{u_i \log(\frac{S+T'}{u_i})}{u_1 \log(\frac{T'+u_1}{u_1}) + u_i \log(\frac{S}{u_i})} \leq \frac{u_i \log(\frac{T'+S}{u_i})}{u_i \log(\frac{T'+u_1}{u_1}) + u_i \log(\frac{S}{u_i})} = \frac{\log(\frac{T'+S}{u_i})}{\log\left(\frac{(T'+u_1)S}{u_1 u_i}\right)} \leq 1$$

where the last inequality uses property (ii), $S \geq u_1$, to ensure that $(T' + S)/u_i \leq (T' + u_1)S/(u_1 u_i)$.

For any $i > 1$, with $d_i^* = 0$, we have that

$$\frac{u_i \log(\frac{S+T'}{u_i})}{u_i \log(\frac{T'+u_1}{u_i})} \leq \frac{u_i \log(\frac{2T'+2u_1}{u_i})}{u_i \log(\frac{T'+u_1}{u_i})} = \frac{u_i \left(\log(\frac{T'+u_1}{u_i}) + 1\right)}{u_i \log(\frac{T'+u_1}{u_i})} \leq 2,$$

where the first inequality follows from property (iii) and the last inequality holds because $T' + u_1 \geq 2u_i$. $\square$

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.*

Proof. The idea is to show a reduction from the NP-Complete problem PARTITION using the fact that for some specific instances of our problem, the optimal partition is the most balanced one.

Consider an instance of PARTITION given by a multiset $U = \{u_1, \dots, u_k\}$ of integers (recall in this problem we want to decide whether this multiset can be partitioned in two sub-multisets with the same sum of elements). Create an instance (V, I_{Entr}) of our problem as follows: for each number $u_i \in U$ add the scaled canonical vector $\mathbf{v}_i = u_i \mathbf{e}_i$ to V.

Let (L, R) be a binary partition of V and let $\mathbf{u} = \sum_{\mathbf{v} \in V} \mathbf{v} = (u_1, \dots, u_k)$. If $\mathbf{d} \in \{0,1\}^k$ is the direction corresponding to partition (L, R), that is, $d_i = 1$ iff $\mathbf{v}_i \in L$ then the impurity of (L, R) is given by

$$I_{Entr}(\mathbf{u} \circ \mathbf{d}) + I_{Entr}(\mathbf{u} \circ (\mathbf{1} - \mathbf{d}))$$

It follows from Lemma 4.3 that the above expression is minimized when

$$\min\{\mathbf{u} \cdot \mathbf{d}, \mathbf{u} \cdot (\mathbf{1} - \mathbf{d})\} = \min \left\{ \sum_{i | \mathbf{v}_i \in L} u_i, \sum_{i | \mathbf{v}_i \in R} u_i \right\}$$

is maximized.

Thus, if we can find in polytime such minimizing partition for V, we can decide whether U can be partitioned into multisets with the same sum. This concludes the reduction. ☐

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 experiments are very similar to those in [6] 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 [6], if all the entries corresponding to a value or a class are zero, we re-generate the contingency table, 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 [6], of the `SLIQ` method proposed in [16]. 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 [6] 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 [6] `PCext` and `SLIQext` outperformed other available methods, such as the `Flip Flop` method [17], 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`		**97.3**	**99.2**	**99.9**	**100.0**
`PCext`	12	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`		**73.9**	**65.8**	**73.3**	**85.3**
`PCext`	25	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`		51.4	33.1	31.0	33.9
`PCext`	50	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`		**98.3**	**99.4**	**100**	**100**
`PCext`	12	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`		**83.3**	**76.9**	**81.0**	**87.7**
`PCext`	25	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`		**70.0**	**57.4**	**53.5**	**52.5**
`PCext`	50	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`,

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

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. We note that most of these results were attained with $n = 12$ and $k = 3$.

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. Among the other three methods, `PCext` obtained 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. They are available in `https://github.com/felipeamp/icml-2018`.

Although `LCA` does not seem to be competitive with the other heuristics in terms of the impurity of the partitions generated, it might be used when both n and k are large and speed is an issue. In addition, `LCA` could be used together with any method, incurring a negligible overhead, to guarantee that the ratio between the impurity of the partition found and the optimal one is bounded.

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.

We employed 11 datasets in total. Eight of them are from the UCI repository: Mushroom, KDD98, Adult, Nursery, Covertype, Cars, Contraceptive and Poker [14]. Two others are available in Kaggle: San Francisco Crime and Shelter Animal Outcome [23, 1]. The last dataset was created by translating texts from the Reuters database [14] into phonemes, using the CMU pronouncing dictionary [5]. We shall note that these datasets were also used in [13] 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. From the KDD98 dataset we derived a new dataset KDD98-9 that contains only the positive samples (people that donate money) of KDD98 and the target attribute, Target_D, is split into 9 classes, where the i-th class correspond to the i-th quantile in terms of amount of money donated. For the Reuters Phonemes

Table 4: Information about the employed datasets after data cleaning and attributes aggregation. Column k is the number of classes and `Reg` stands for Regression; columns m_{nom} and m_{nom}^{ext} are the number of nominal attributes in the original and extended datasets (those with the extra attributes), respectively; column m_{num} is the number of numeric attributes.

Dataset	Samples	k	m_{nom}	m_{nom}^{ext}	m_{num}
`Mushroom`	5644	2	22	N/A	0
`Adult`	30162	2	8	N/A	6
`KDD98`	4843	`Reg`	65	N/A	314
`Nursery`	12960	5	8	11	0
`CoverType`	581012	7	44	46	10
`Car`	1728	4	6	8	0
`Contracep`	1473	3	7	9	2
`Poker`	25010	10	10	N/A	0
`Shelter`	26711	22	5	N/A	1
`S.F. Crime`	878049	39	3	N/A	2
`Phonemes`	10000	15	3	N/A	0

dataset, we extracted 10000 samples containing the 15 most common phonemes as class and tried to predict when they are about to happen given the 3 preceding phonemes. This dataset is motivated by Spoken Language Recognition problems, where phonotactic models are used as an important part of the classification system [18]. For the San Francisco Crime dataset, we give the month, day of the week, police department district and latitute/longitude and try to predict the crime category. Lastly, for the Shelter Animal Outcomes dataset, we converted the age into a numeric field containing the number of days old and separated the breed into two categorical fields, repeating the breed in both in case there was only one originally. We also removed the AnimalID, Name and the DateTime. For this dataset we try to predict the outcome type and subtype (concatenated into a single categorical field). For both San Francisco Crime and Shelter Animal Outcomes datasets we created a version of them (`S.F. Crime-15` and `Shelter-15`), containing only 15 classes, instead of the 39 and 22 original ones, respectively. This was done by grouping the rarest classes into a single one.

For datasets `Cars`, `CoverType`, `Nursery` and `Contraceptive` we added new nominal attributes that were obtained by aggregating some of the original ones as we describe below. The goal was to obtain natural attributes with a larger number of values. In our experiments we consider the datasets with these extra attributes rather then the original ones and we removed samples with missing values from all datasets. Table 4 provides some statistics.

- `Nursery-Ext`. This dataset is obtained by adding three new attributes to dataset Nursery. The first attribute has 15 distinct values and it is constructed through the aggregation of 2 attributes from group EMPLOY, one with 5 values and the other with 3 values. The second attribute has 72 distinct values corresponding to the aggregation of attributes from the attributes in group `STRUCT_FINAN`. The third attribute, with 9 distinct values, is the combination of the attributes in group `SOC_HEALTH`.

- `Covertype-Ext`. We combined 40 binary attributes related with the soil type into a new attribute with 40 distinct values. The same approach was employed to combine the 4 binary attributes related with the wilderness area into a new attribute with 4 distinct values. This is an interesting case because, apparently, the 40 (soil type) binary attributes as well as the 4 (wilderness area) binary attributes were derived from a binarization of two attributes, one with 40 distinct value and the other with 4 distinct values.

Table 5: Average accuracy for Decision Trees built using Gini impurity over 20 3-fold stratified cross-validations. The best results are bold faced.

	`SliqExt`	`PCext`	`HcC`	`LCA`
`Adult`	83.11	**83.28**	83.25	83.25
`Mushroom`	99.9	**100**	**100**	**100**
`KDD98-9`	37.92	**39.88**	38.96	38.8
`Nursery-Ext`	94.13	**95.83**	**95.83**	**95.83**
`Cars-Ext`	91.23	96.46	**96.5**	**96.5**
`CoverType-Ext`	88.31	88.32	**89.39**	88.72
`Cotraceptive-Ext`	48.86	48.52	**49.31**	48.97
`PokerHand`	**52.54**	52.09	52.09	51.97
`San Francisco Crime`	**27.51**	27.13	27.13	27.16
`Shelter Animal`	**55.08**	53.82	53.59	53.6
`Reuters Phonemes`	36.11	**37.95**	37.89	37.8
`Average`	64.97	65.75	65.81	65.69

- `Cars-Ext`. To obtain this dataset, the 2 attributes related with the concept `PRICE`, `buying` and `maint`, were combined into an attribute with 16 distinct values. Moreover, the 3 attributes related with concept `CONFORT` were combined into an an attribute with 36 distinct values.
- `Contraceptive-Ext`. The 2 attributes related with the couple's education were combined into an attribute with 16 distinct values. Moreover, the 3 attributes related with the couple's occupations and standard of living were aggregated into a new attribute with 32 distinct values.

In this second set of experiments we build decision trees with depth at most 16. To prevent the selection of non-informative nominal attributes, we used a χ^2-test for each attribute at every node of the tree: if the χ^2-test on the contingency table of attribute A has p-value larger than 10% at a node ν, then A is not used in ν. Furthermore, attributes with less than 15 samples associated with its second most frequent value are also not considered for splitting. This helps avoid data overfitting.

Table 5 and 6 present the average accuracy achieved over 20 3-fold stratified cross-validations using Gini and Entropy impurities, respectively. Moreover, we used a 95% one-tailed paired t-student test to compare the accuracy attained by the different methods. Tables 7 and 8 show, respectively, how `LCA` and `HcC` compare with each of the other methods with regards to the number of datasets in which the had statistically better/worse accuracy. As an example, the entry associated with Entropy/`PCExt` in Table 7 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 discussed in 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.

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 the methods proposed in this paper 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.

Table 6: Average accuracy for Decision Trees built using Entropy impurity over 20 3-fold stratified cross-validations. The best results are bold faced.

	`SliqExt`	`PCext`	`HcC`	`LCA`
`Adult`	83.6	**83.74**	**83.74**	**83.74**
`Mushroom`	99.99	**100**	**100**	**100**
`KDD98-9`	36.6	**38.77**	38.67	38.75
`Nursery-Ext`	94.13	**95.88**	**95.88**	**95.88**
`Cars-Ext`	92.42	96.3	96.28	**96.54**
`Contraceptive-Ext`	**48.99**	48.53	48.76	48.93
`CoverType-Ext`	88.26	88.35	88.83	**88.98**
`Poker Hand`	51.16	51.37	51.59	**51.97**
`San Francisco Crime`	**27.56**	27.31	27.31	27.33
`Shelter Animal`	**55.12**	54.18	54.2	54.17
`Reuters Phonemes`	35.27	37.25	36.91	**37.48**
`Average`	64.83	65.61	65.65	65.8

Table 7: Number of datasets in which `LCA` 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

Table 8: Number of datasets in which `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`		`LCA`		`SliqExt`	
Gini	+2	-2	+3	-1	+8	-3
Entropy	+2	-1	+0	-4	+7	-2

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 $1/\epsilon$. We believe that an algorithm with the flavor of `HcC` might have this property. The reason is that `HcC` considers a set of directions that covers the space where the optimal direction lies, in the sense that for every direction in this space there exists one in the `HcC`'s set that is 'close' to it. Thus, considering a ϵ-net of directions, one of them should generate a binary partition with impurity similar to the optimal one.

Another interesting contribution would be an analysis of the approximation ratio of `PCext` since, according to our experiments and those reported in [6], it provides a good trade-off between the running time and the impurity of the generated partitions, specially for the Gini impurity.

Finally, 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

[1] Austin-Animal-Center. Shelter animal outcomes dataset. `kaggle.com/c/shelter-animal-outcomes`.

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

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

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

[5] CMU. Cmu pronouncing dictionary. `www.speech.cs.cmu.edu/cgi-bin/cmudict`.

[6] D. Coppersmith, S. J. Hong, and J. R. M. Hosking. Partitioning nominal attributes in decision trees. *Data Min. Knowl. Discov*, 3(2):197–217, 1999.

[7] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algoritms.* McGraw-Hill Book Company, 1998.

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

[9] T. C. Gülcü, M. Y. 0005, and A. Barg. Construction of polar codes for arbitrary discrete memoryless channels. In *ISIT*, pages 51–55. IEEE, 2016.

[10] T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning. *Journal of Computational and Graphical Statistics*, 15(3):651–674, Sept. 2006.

[11] A. Kartowsky and I. Tal. Greedy-merge degrading has optimal power-law. *CoRR*, abs/1703.04923, 2017.

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

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

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

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

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

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

[18] J. Navrátil. Recent advances in phonotactic language recognition using binary-decision trees. In *INTERSPEECH*. ISCA, 2006.

[19] B. Nazer, O. Ordentlich, and Y. Polyanskiy. Information-distilling quantizers. In *ISIT*, pages 96–100. IEEE, 2017.

[20] S. Nowozin. Improved information gain estimates for decision tree induction. In *ICML*, 2012.

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

[22] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1992.

[23] SF-OpenData. San francisco crime dataset. `kaggle.com/c/sf-crime`.

[24] I. Tal and A. Vardy. How to construct polar codes. *IEEE Trans. Information Theory*, 59(10):6562–6582, 2013.

A A result about property (P3)

Proposition A.1. *If f satisfies properties $P(1)$ and $P(2)$ and f is a smooth function such that $xf''(x)$ is non-increasing in the interval [0,1] then f also satisfies property (P3).*

Proof. Let $f'(x)$ be the derivative of f on point x. The proof consists of showing two steps

(i) if $f'(x) - f(x)/x$ is non-increasing in $(0, 1]$ then f satisfies (P3).

(ii) if $xf''(x)$ is non-increasing in $[0, 1]$ then $f'(x) - f(x)/x$ is non-increasing in $(0, 1]$.

We first show step (i). We can assume $q < 1$ because for $q = 1$ (P3) is trivial. The condition on f implies $f'(p) - f(p)/p \geq f'(q) - f(q)/q$ for $p < q$, which is equivalent to $g'(p)p \geq g'(q)q$, where $g(x) = f(x)/x$. Thus, we make use of this alternative condition. We have that

$$g(p/q) - g(p) = \int_p^{p/q} g'(x)dx = \frac{p}{q}\int_q^1 g'\left(\frac{px}{q}\right)dx \geq \frac{p}{q}\int_q^1 \left(\frac{q}{p}\right) g'(x)dx = -g(q),$$

where the last inequality follows from the condition on g. Thus, $g(p) \leq g(p/q) + g(q)$. Multiplying both sides by p we get $p \cdot g(p) \leq p \cdot g(p/q) + p \cdot g(q)$. By using $g(p/q) = f(p/q)(q/p)$ and $g(q) = f(q)/q$ we get that

$$f(p) \leq qf(p/q) + p/qf(q).$$

Now we prove step (ii). We have that $f'(x)-f(x)/x$ is not increasing in $(0, 1]$ if and only if its derivative is smaller than or equal to 0 in the interval (0,1]. Thus, we must have

$$\frac{x^2 f''(x) - xf'(x) + f(x)}{x^2} \leq 0. \tag{A.19}$$

First we note that

$$\lim_{x\to 0} \frac{x^2 f''(x) - xf'(x) + f(x)}{x^2} \leq \lim_{x\to 0} \frac{x^2 f''(x) - xf'(x) + xf'(0)}{x^2} = f''(0) - f''(0) = 0$$

Hence, if the derivative of $h(x) = x^2 f''(x) - xf'(x) + f(x)$ is smaller than or equal to 0 in [0,1] then inequality (A.19) holds. We have that $h'(x) = x^2 f'''(x) + xf''(x)$. Thus, (A.19) holds if $xf'''(x) + f''(x) \leq 0$ for every $x \in [0, 1]$. Let $m(x) = xf''(x)$. We have that $m'(x) = xf'''(x) + f''(x)$. Since $m(x)$ is non-increasing in [0,1] it follows that $m'(x) \leq 0$ for every $x \in [0, 1]$, and thus inequality (A.19) holds. □